Smart Chaos Leveraging Generative AI in Chaos Engineering

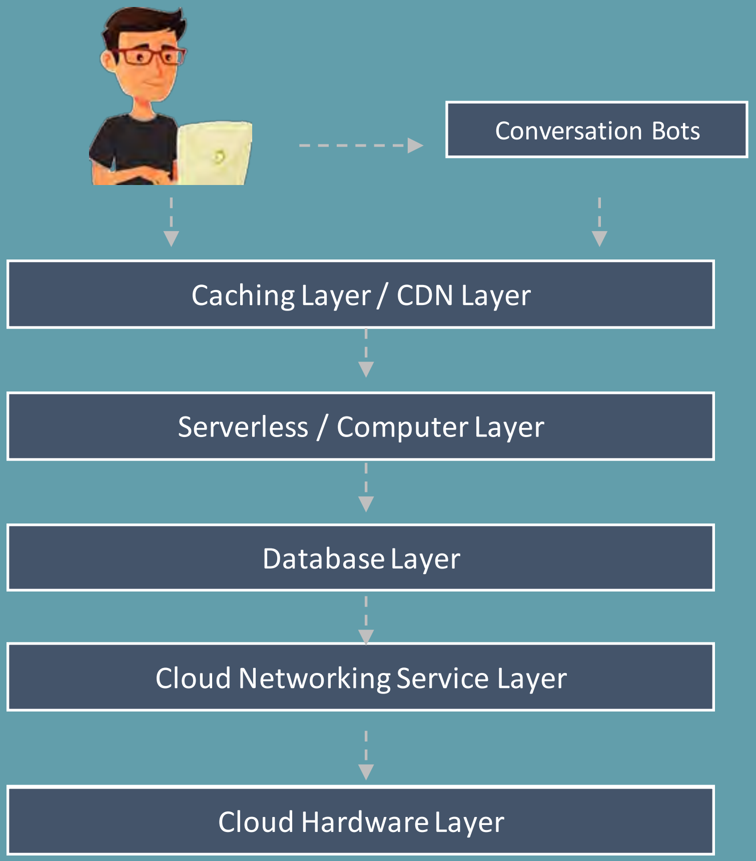
Indika Wimalasuriya

Chaos Engineering 2024

Agenda

- **Significance of Resilience in Distributed Systems**
- **Introduction to Chaos Engineering and GenAI**
- **Chaos Engineering - Methodoloy**
- **Optimizing Chaos Engineering Stages with GenAI**
- **Exploration of Specific GenAI Use Cases in Chaos Engineering**
- **Best practices and potential pitfalls - just my two cents**

Modern enterprise are complex and increasingly hard to manage



Conversation Bots

Caching Layer / CDN Layer

Serverless / Computer Layer

Database Layer

Cloud Networking Service Layer

Cloud Hardware Layer

- **Failures can occur at all logical layers.**
- **Failures can happen at any time, sometimes in combination with other errors.**
- **Bugs may manifest long after they have been deployed to a system.**
- **Bugs have the potential to propagate across the entire system.**
- **Problems tend to exacerbate at higher levels of the system due to recursion.**

Next major outage - when?

*“2021, Facebook and its properties, Instagram and WhatsApp, were down for more than **five hours** due to configuration changes on routers in Facebook's data centers. A five-hour outage is an eternity in our always-on digital economy, costing the company an estimated $65 million and 4.8% in stock valuation.”*

Source : https://www.forbes.com/sites/forbestechcouncil/2021/11/29/observability-and-aiops-why-convergence-is-the-future-to-improving-uptime/?sh=7d013bdb55f1

*“2021, Fastly (leading CDN) had an outage that lasted 1 hour causing major down time for Amazon, eBay, Reddit, Spotify, Twitch, The Guardian, The New York Times, and even UK government websites. This was due to a **bug introduced part of software** development, later got triggered by a configuration change pushed”*

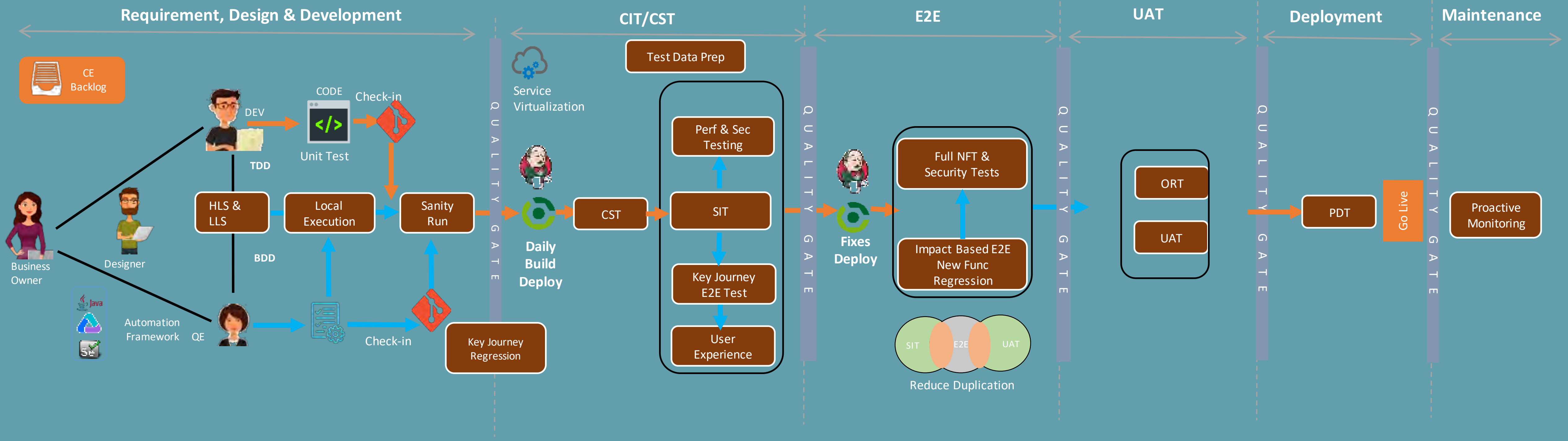
https://www.bmc.com/blogs/network-outages/

*“Datadog experienced a **substantial outage** on March 8, 2023, affecting multiple regions due to a network stack issue triggered by an automatic security update. The root cause was identified as systemd deleting routes during a restart, leading to a prolonged disruption in web access, APIs, monitors, and data ingestion.”*

Source : https://www.datadoghq.com/blog/2023-03-08-multiregion-infrastructure-connectivity-issue/

Why Traditional Testing Falls Short?

Testing verifies the known, but in the dance with failures, the steps are often unrehearsed and spontaneous.



Requirement, Design & Development

CE Backlog

Business Owner

Designer

DEV

CODE

Unit Test

Check-in

TDD

HLS & LLS

Local Execution

Sanity Run

BDD

Automation Framework

QE

Check-in

Key Journey Regression

QUALITY GATE

CIT/CST

Service Virtualization

Daily Build Deploy

Test Data Prep

CST

Perf & Sec Testing

SIT

Key Journey E2E Test

User Experience

QUALITY GATE

Fixes Deploy

E2E

Full NFT & Security Tests

Impact Based E2E New Func Regression

SIT

E2E

UAT

Reduce Duplication

QUALITY GATE

UAT

ORT

UAT

QUALITY GATE

Deployment

PDT

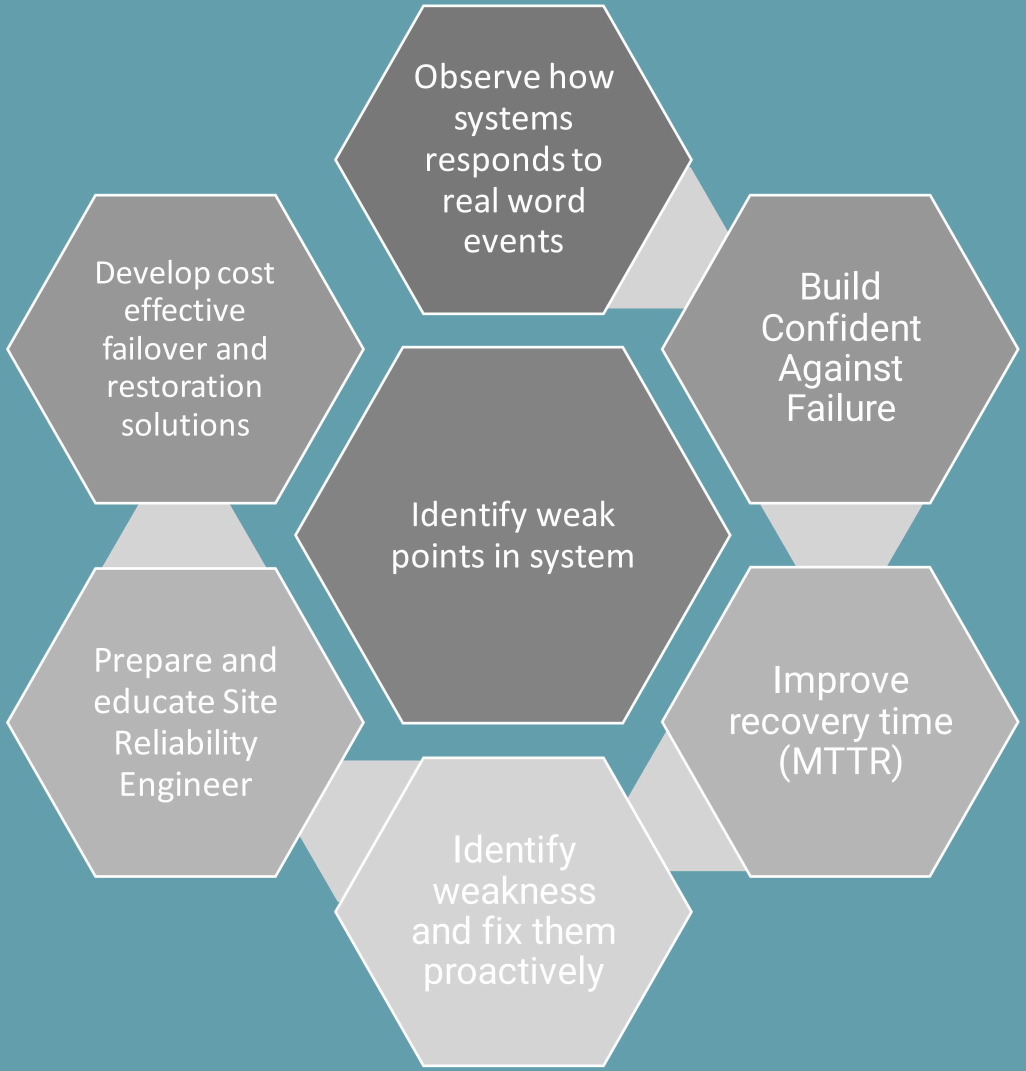
Go Live

QUALITY GATE

Maintenance

Proactive Monitoring

Why Chaos Engineering



Identify weak points in system

Observe how systems responds to real word events

Build Confident Against Failure

Improve recovery time (MTTR)

Identify weakness and fix them proactively

Prepare and educate Site Reliability Engineer

Develop cost effective failover and restoration solutions

- ⬆ **Increase Reliability and Resiliency of Systems**
- ⬆ **Increase End User Satisfaction (Achieve SLOs)**
- ⬆ **Improve team collaboration and skill**
- ⬇ **Reduce Mean Time to Resolve (MTTR)**
- ⬆ **Improve Observability and Monitoring**

What is GenAI(Generative AI)

Generative AI refers to machine learning models that can create new, original content like text, images, audio, and more

What is difference been predictive or analytical AI vs generative AI?

- Unlike predictive or analytical AI, generative AI models are not focused on classifying data or finding patterns
- Instead, they are trained on massive datasets to learn how to generate brand new outputs that are similar to the training data

Prominent examples of generative AI models:

- GPT-3: Generates human-like text and can converse in natural language
- DALL-E: Creates original images from text descriptions
- AlphaFold: Predicts 3D protein structure from amino acid sequence

Key features of generative AI models:

- These models are trained via machine learning approaches like deep learning and neural networks
- They continue to learn and improve as they are exposed to more training data over time
- The goal is for the AI to generate novel, high-quality content that is realistic and useful for human applications

Applications of GenAI

Text Generation:

- Articles & stories - GPT-3 can generate news articles and fiction stories
- Code - GitHub Copilot suggests context-relevant code for developers

Video Generation:

- AI models can generate or edit video content

Other Creative Applications:

- Generating logos, recipes, fashion designs, architectural drawings
- Personalizing content like customized ads or product recommendations

Image Generation:

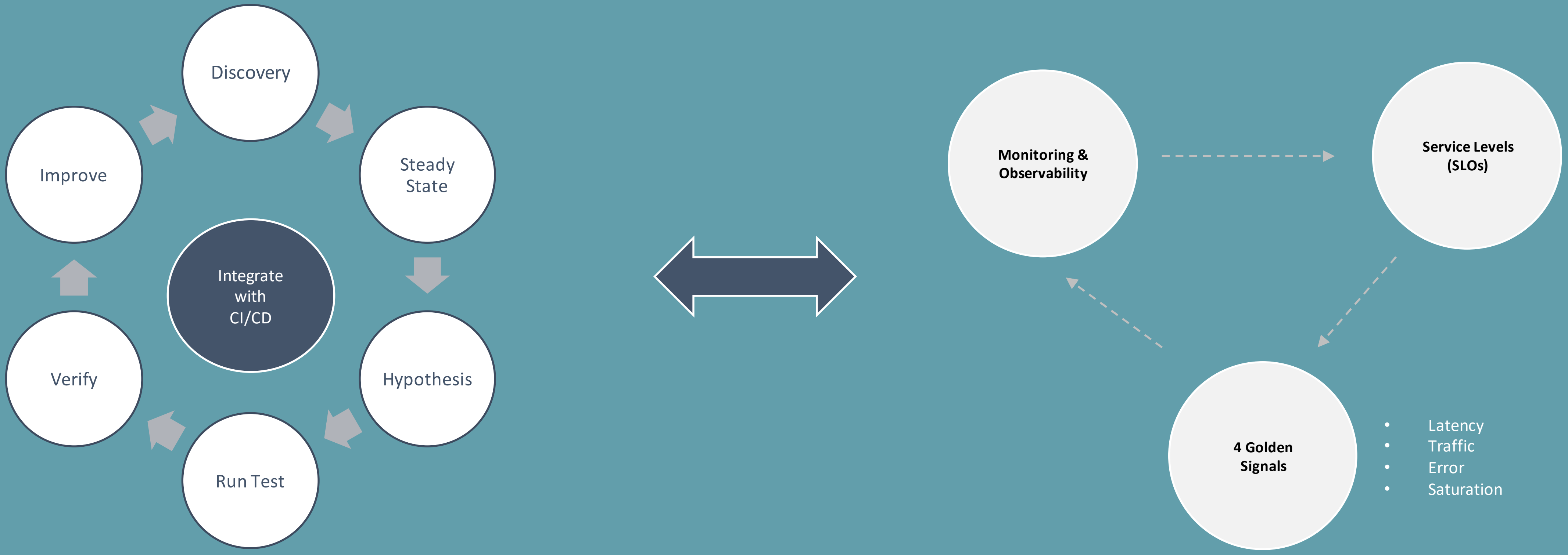
- Art - DALL-E 2 creates original digital art from text prompts
- Photos - Generative models can edit or enhance photos

Drug Discovery:

- Models can analyze chemical compounds and suggest new drug candidates

Chaos Engineering – "Methodology"

Chaos Engineering is the discipline of experimenting on a distributed system to build confidence in its ability to withstand turbulent conditions in production.



- Latency
- Traffic
- Error
- Saturation

Th Process

Measure Everything

Enhancing Chaos Engineering with Generative AI: A Comprehensive Framework

01

Discovery Phase

Leverage GenAI for anomaly detection in historical data, uncovering potential weaknesses or areas of interest

02

Dependency Analysis

Utilize GenAI to analyze system dependencies, identifying points of failure or vulnerabilities in the architecture.

03

Steady State Definition

Train GenAI models to predict performance metrics, aiding in defining the system's steady state.

04

Hypothesis Creation

Implement GenAI for automated hypothesis generation

05

Experiment Design

Use GenAI for scenario simulation, assisting in designing controlled experiments and optimizing testing efficiency.

06

Blast Radius Definition

Utilize GenAI to assess the potential blast radius of chaos experiments, considering dependencies and system topology.

07

Rumsfeld Matrix - "Known Knowns, Known Unknowns, and Unknown Unknowns“

Leverage GenAI for data analysis to categorize knowns and unknowns, aiding in identifying potential unknown unknowns.

08

Monitoring and Analysis

Integrate GenAI models into real-time monitoring for anomaly detection during chaos experiments and root cause analysis.

09

Documentation and Reporting

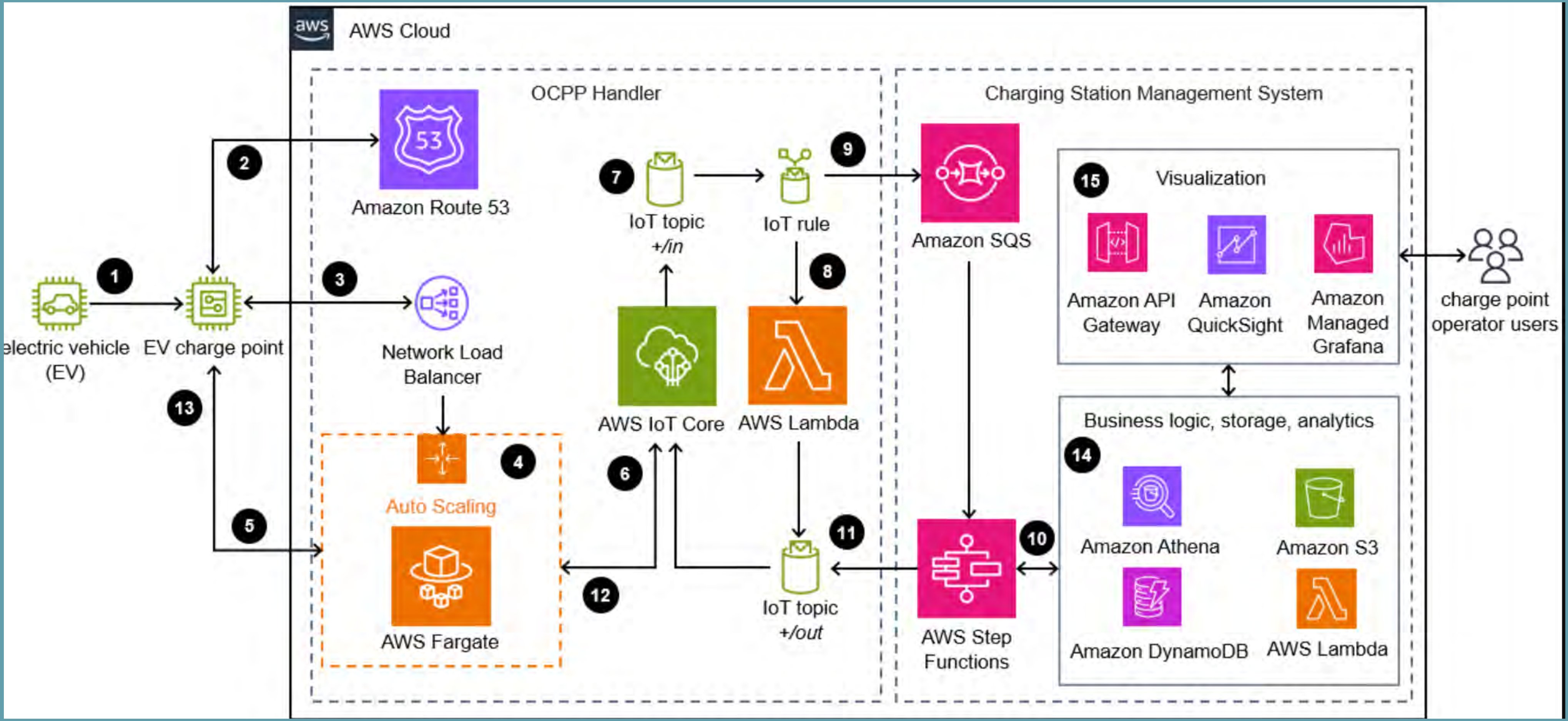
Employ GenAI for automated reporting, summarizing chaos experiment outcomes and consolidating insights.

10

Iterative Improvement

Establish a continuous feedback loop with GenAI, allowing for adaptive experimentation and improved chaos engineering approaches.

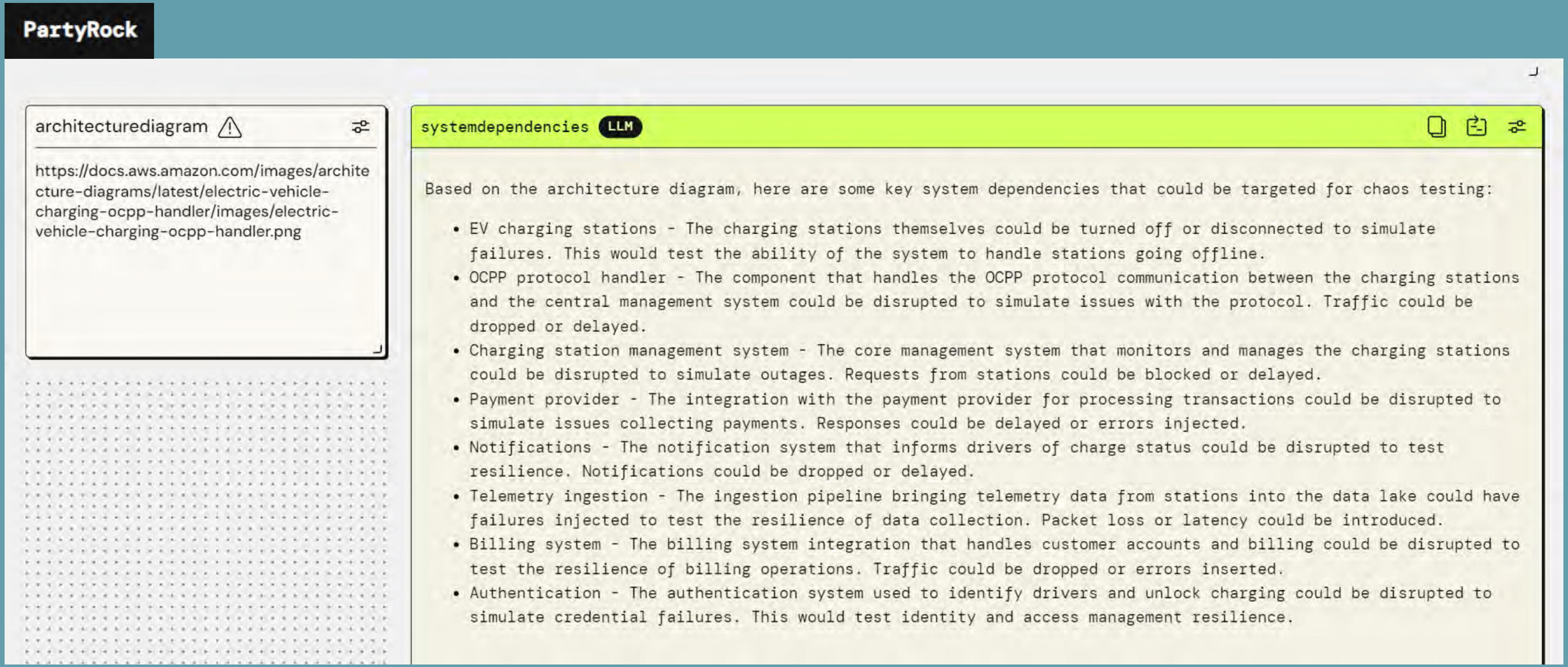
Generative AI in Action: Illuminating Possibilities via Architecture Diagrams



https://docs.aws.amazon.com/images/architecture-diagrams/latest/electric-vehicle-charging-ocpp-handler/images/electric-vehicle-charging-ocpp-handler.png

Dependency Analysis with GenAI

Supply the architecture diagram link to the Generative AI model (Claude) and let it automatically generate crucial system dependencies.

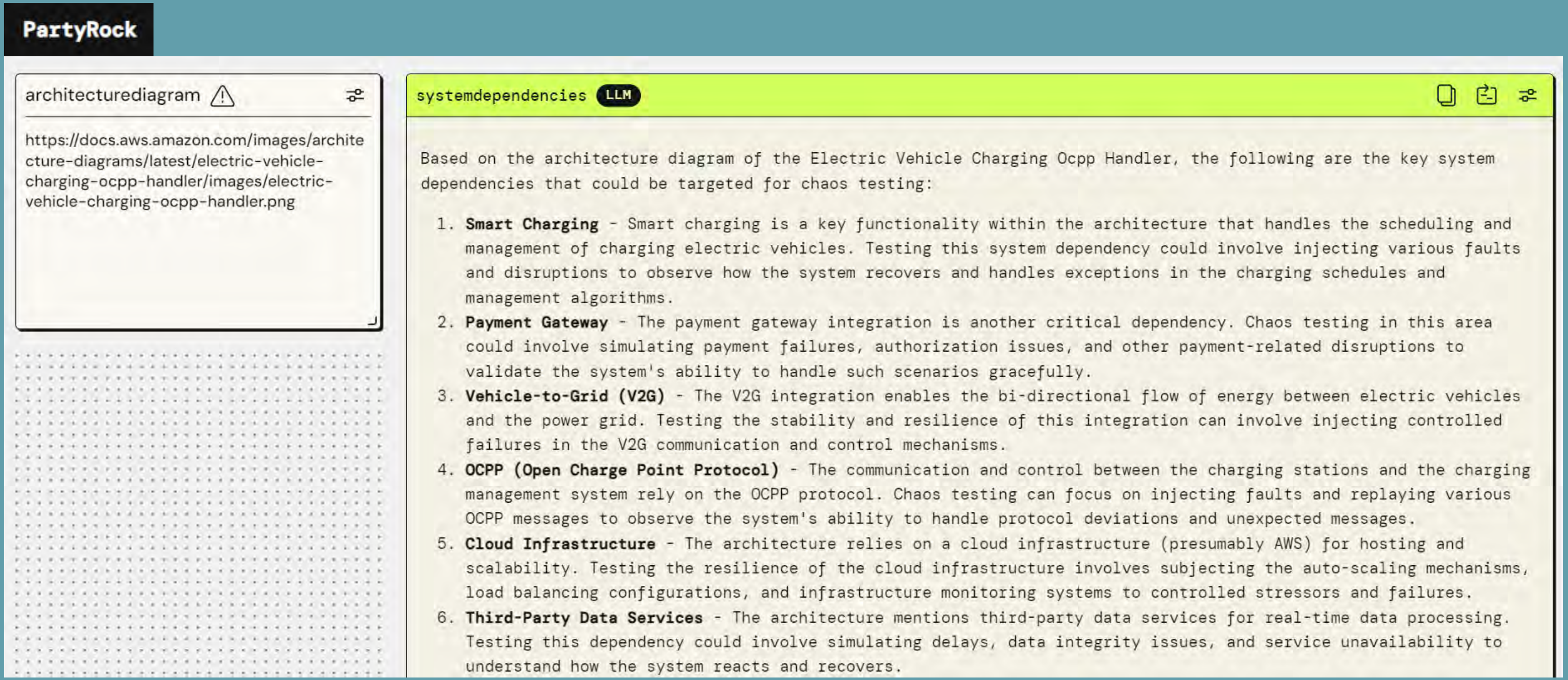


https://docs.aws.amazon.com/images/architecture-diagrams/latest/electric-vehicle-charging-ocpp-handler/images/electric-vehicle-charging-ocpp-handler.png

Based on the architecture diagram, here are some key system dependencies that could be targeted for chaos testing:

- EV charging stations - The charging stations themselves could be turned off or disconnected to simulate failures. This would test the ability of the system to handle stations going offline.
- OCPP protocol handler - The component that handles the OCPP protocol communication between the charging stations and the central management system could be disrupted to simulate issues with the protocol. Traffic could be dropped or delayed.
- Charging station management system - The core management system that monitors and manages the charging stations could be disrupted to simulate outages. Requests from stations could be blocked or delayed.
- Payment provider - The integration with the payment provider for processing transactions could be disrupted to simulate issues collecting payments. Responses could be delayed or errors injected.
- Notifications - The notification system that informs drivers of charge status could be disrupted to test resilience. Notifications could be dropped or delayed.
- Telemetry ingestion - The ingestion pipeline bringing telemetry data from stations into the data lake could have failures injected to test the resilience of data collection. Packet loss or latency could be introduced.
- Billing system - The billing system integration that handles customer accounts and billing could be disrupted to test the resilience of billing operations. Traffic could be dropped or errors inserted.
- Authentication - The authentication system used to identify drivers and unlock charging could be disrupted to simulate credential failures. This would test identity and access management resilience.

Dependency Analysis with GenAI



https://docs.aws.amazon.com/images/architecture-diagrams/latest/electric-vehicle-charging-ocpp-handler/images/electric-vehicle-charging-ocpp-handler.png

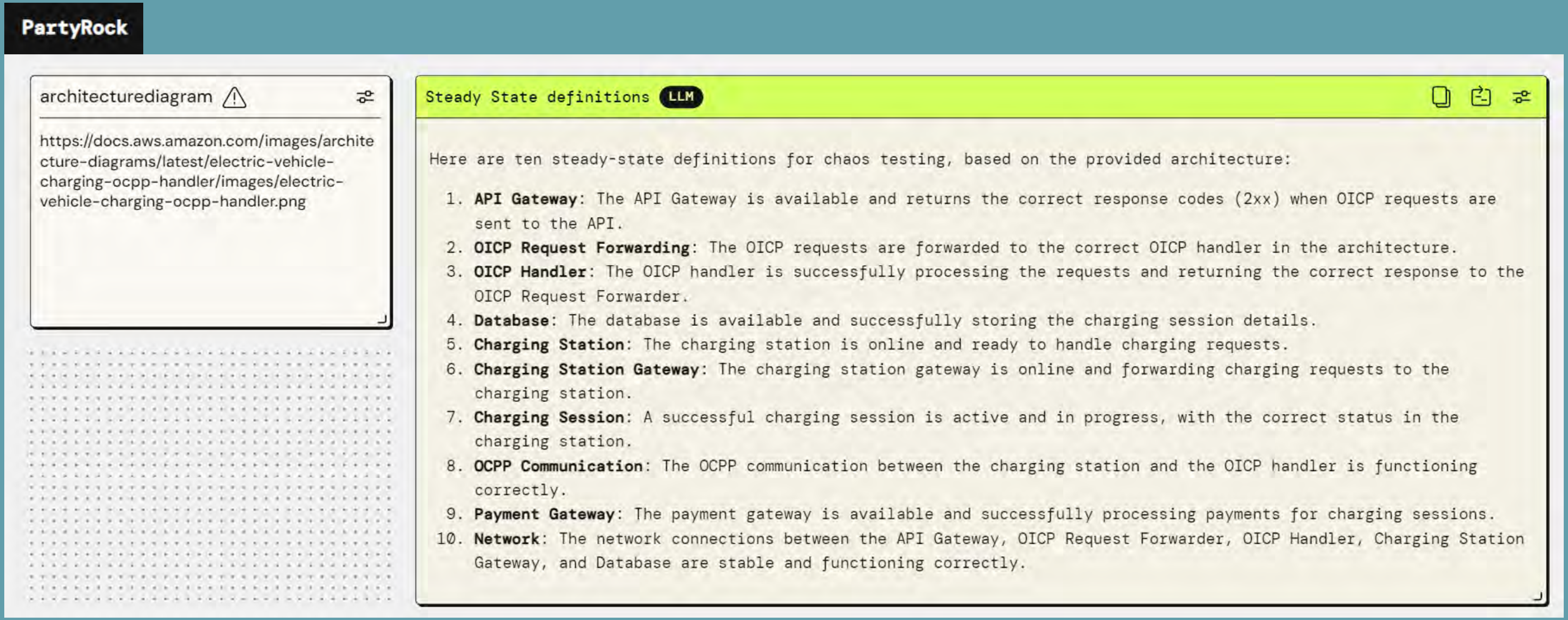
Based on the architecture diagram of the Electric Vehicle Charging Ocpp Handler, the following are the key system dependencies that could be targeted for chaos testing:

1. **Smart Charging** - Smart charging is a key functionality within the architecture that handles the scheduling and management of charging electric vehicles. Testing this system dependency could involve injecting various faults and disruptions to observe how the system recovers and handles exceptions in the charging schedules and management algorithms.
2. **Payment Gateway** - The payment gateway integration is another critical dependency. Chaos testing in this area could involve simulating payment failures, authorization issues, and other payment-related disruptions to validate the system's ability to handle such scenarios gracefully.
3. **Vehicle-to-Grid (V2G)** - The V2G integration enables the bi-directional flow of energy between electric vehicles and the power grid. Testing the stability and resilience of this integration can involve injecting controlled failures in the V2G communication and control mechanisms.
4. **OCPP (Open Charge Point Protocol)** - The communication and control between the charging stations and the charging management system rely on the OCPP protocol. Chaos testing can focus on injecting faults and replaying various OCPP messages to observe the system's ability to handle protocol deviations and unexpected messages.
5. **Cloud Infrastructure** - The architecture relies on a cloud infrastructure (presumably AWS) for hosting and scalability. Testing the resilience of the cloud infrastructure involves subjecting the auto-scaling mechanisms, load balancing configurations, and infrastructure monitoring systems to controlled stressors and failures.
6. **Third-Party Data Services** - The architecture mentions third-party data services for real-time data processing. Testing this dependency could involve simulating delays, data integrity issues, and service unavailability to understand how the system reacts and recovers.

Pro-Tip : Experiment with different models to achieve more precise results (in this instance, we utilized the command model)

Steady State Defining with GenAI

Supply the architecture diagram link to the Generative AI model (Claude) and let it automatically generate the steady-state definitions for the system



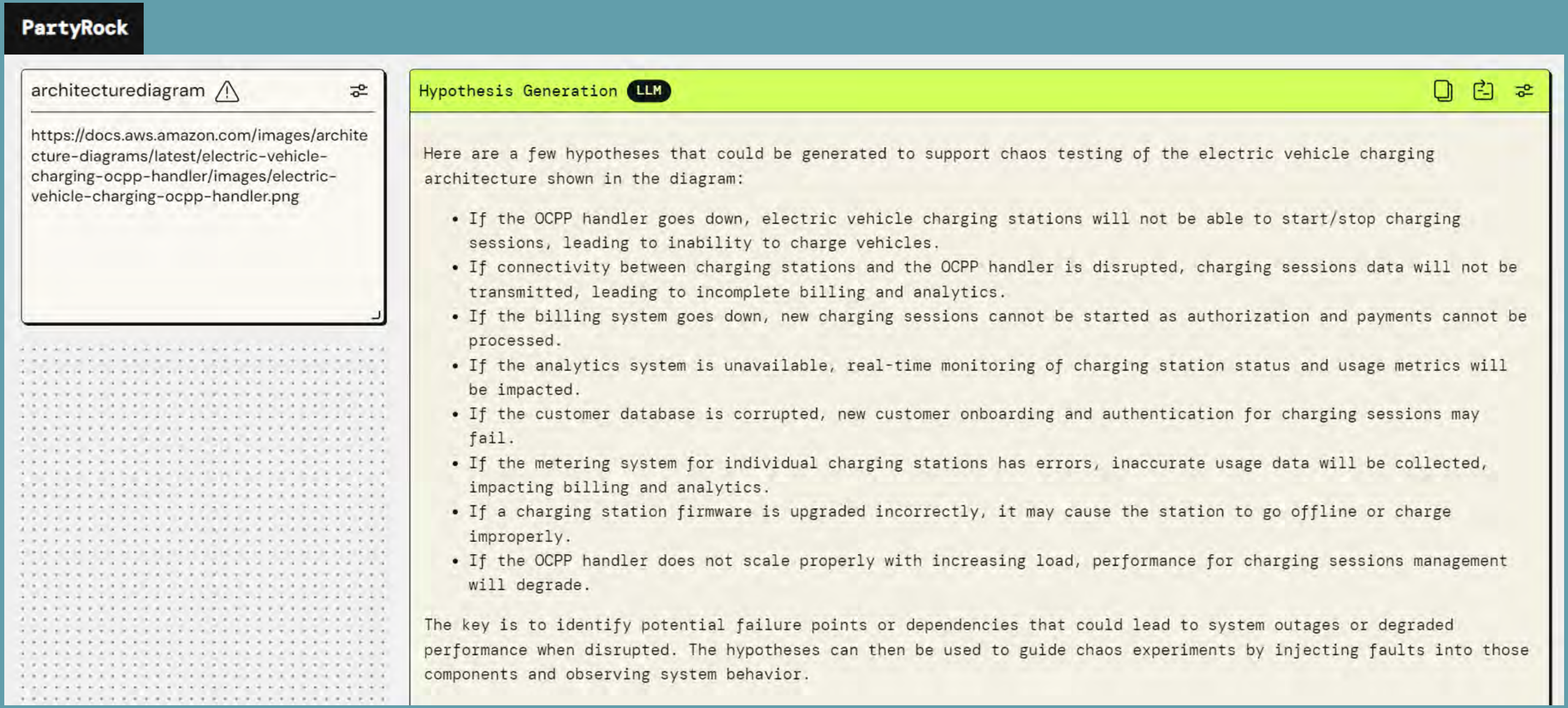
https://docs.aws.amazon.com/images/architecture-diagrams/latest/electric-vehicle-charging-ocpp-handler/images/electric-vehicle-charging-ocpp-handler.png

Here are ten steady-state definitions for chaos testing, based on the provided architecture:

1. **API Gateway**: The API Gateway is available and returns the correct response codes (2xx) when OICP requests are sent to the API.
2. **OICP Request Forwarding**: The OICP requests are forwarded to the correct OICP handler in the architecture.
3. **OICP Handler**: The OICP handler is successfully processing the requests and returning the correct response to the OICP Request Forwarder.
4. **Database**: The database is available and successfully storing the charging session details.
5. **Charging Station**: The charging station is online and ready to handle charging requests.
6. **Charging Station Gateway**: The charging station gateway is online and forwarding charging requests to the charging station.
7. **Charging Session**: A successful charging session is active and in progress, with the correct status in the charging station.
8. **OCPP Communication**: The OCPP communication between the charging station and the OICP handler is functioning correctly.
9. **Payment Gateway**: The payment gateway is available and successfully processing payments for charging sessions.
10. **Network**: The network connections between the API Gateway, OICP Request Forwarder, OICP Handler, Charging Station Gateway, and Database are stable and functioning correctly.

Hypothesis Creation with GenAI

Supply the architecture diagram link to the Generative AI model (Claude) and let it automatically generate hypothesis



https://docs.aws.amazon.com/images/architecture-diagrams/latest/electric-vehicle-charging-ocpp-handler/images/electric-vehicle-charging-ocpp-handler.png

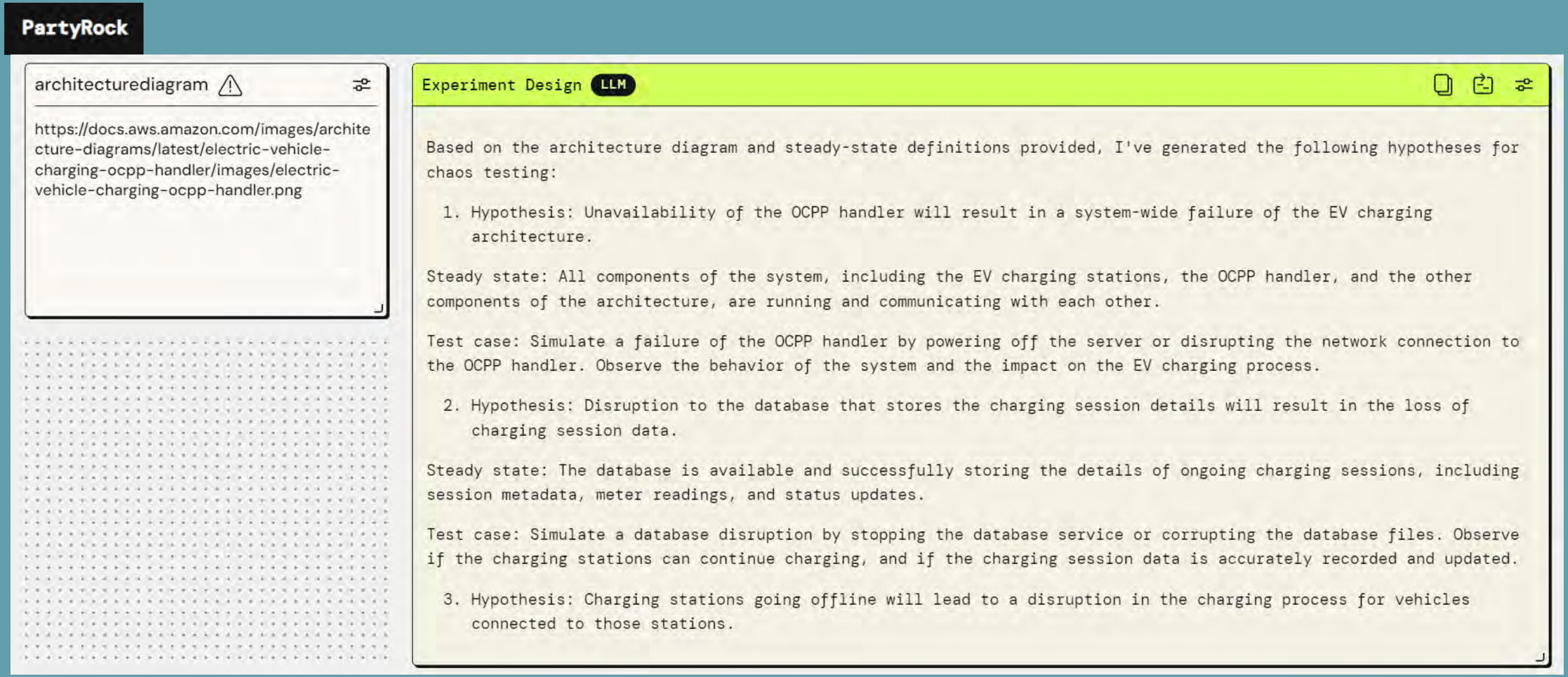
Here are a few hypotheses that could be generated to support chaos testing of the electric vehicle charging architecture shown in the diagram:

- If the OCPP handler goes down, electric vehicle charging stations will not be able to start/stop charging sessions, leading to inability to charge vehicles.
- If connectivity between charging stations and the OCPP handler is disrupted, charging sessions data will not be transmitted, leading to incomplete billing and analytics.
- If the billing system goes down, new charging sessions cannot be started as authorization and payments cannot be processed.
- If the analytics system is unavailable, real-time monitoring of charging station status and usage metrics will be impacted.
- If the customer database is corrupted, new customer onboarding and authentication for charging sessions may fail.
- If the metering system for individual charging stations has errors, inaccurate usage data will be collected, impacting billing and analytics.
- If a charging station firmware is upgraded incorrectly, it may cause the station to go offline or charge improperly.
- If the OCPP handler does not scale properly with increasing load, performance for charging sessions management will degrade.

The key is to identify potential failure points or dependencies that could lead to system outages or degraded performance when disrupted. The hypotheses can then be used to guide chaos experiments by injecting faults into those components and observing system behavior.

Experiment Design with GenAI

Supply the architecture diagram link to the Generative AI model (Command) and let it automatically generate Experiment and Test cases



https://docs.aws.amazon.com/images/architecture-diagrams/latest/electric-vehicle-charging-ocpp-handler/images/electric-vehicle-charging-ocpp-handler.png

Based on the architecture diagram and steady-state definitions provided, I've generated the following hypotheses for chaos testing:

1. Hypothesis: Unavailability of the OCPP handler will result in a system-wide failure of the EV charging architecture.

Steady state: All components of the system, including the EV charging stations, the OCPP handler, and the other components of the architecture, are running and communicating with each other.

Test case: Simulate a failure of the OCPP handler by powering off the server or disrupting the network connection to the OCPP handler. Observe the behavior of the system and the impact on the EV charging process.

2. Hypothesis: Disruption to the database that stores the charging session details will result in the loss of charging session data.

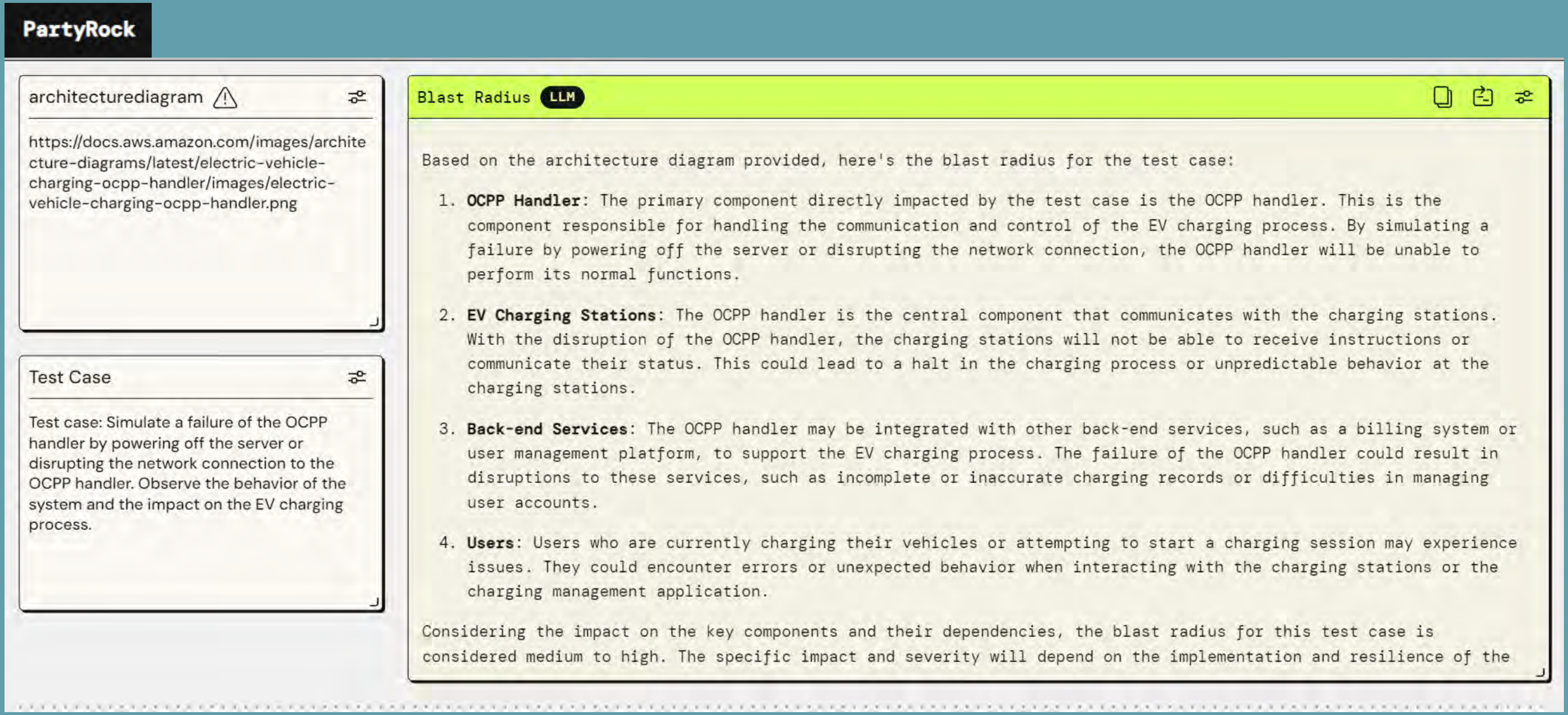
Steady state: The database is available and successfully storing the details of ongoing charging sessions, including session metadata, meter readings, and status updates.

Test case: Simulate a database disruption by stopping the database service or corrupting the database files. Observe if the charging stations can continue charging, and if the charging session data is accurately recorded and updated.

3. Hypothesis: Charging stations going offline will lead to a disruption in the charging process for vehicles connected to those stations.

Blast Radius Defining with GenAI

Supply the Test case and link to Architecture diagram to Generative AI model (Command) and let it automatically generate blast radius for the partiucler test



https://docs.aws.amazon.com/images/architecture-diagrams/latest/electric-vehicle-charging-ocpp-handler/images/electric-vehicle-charging-ocpp-handler.png

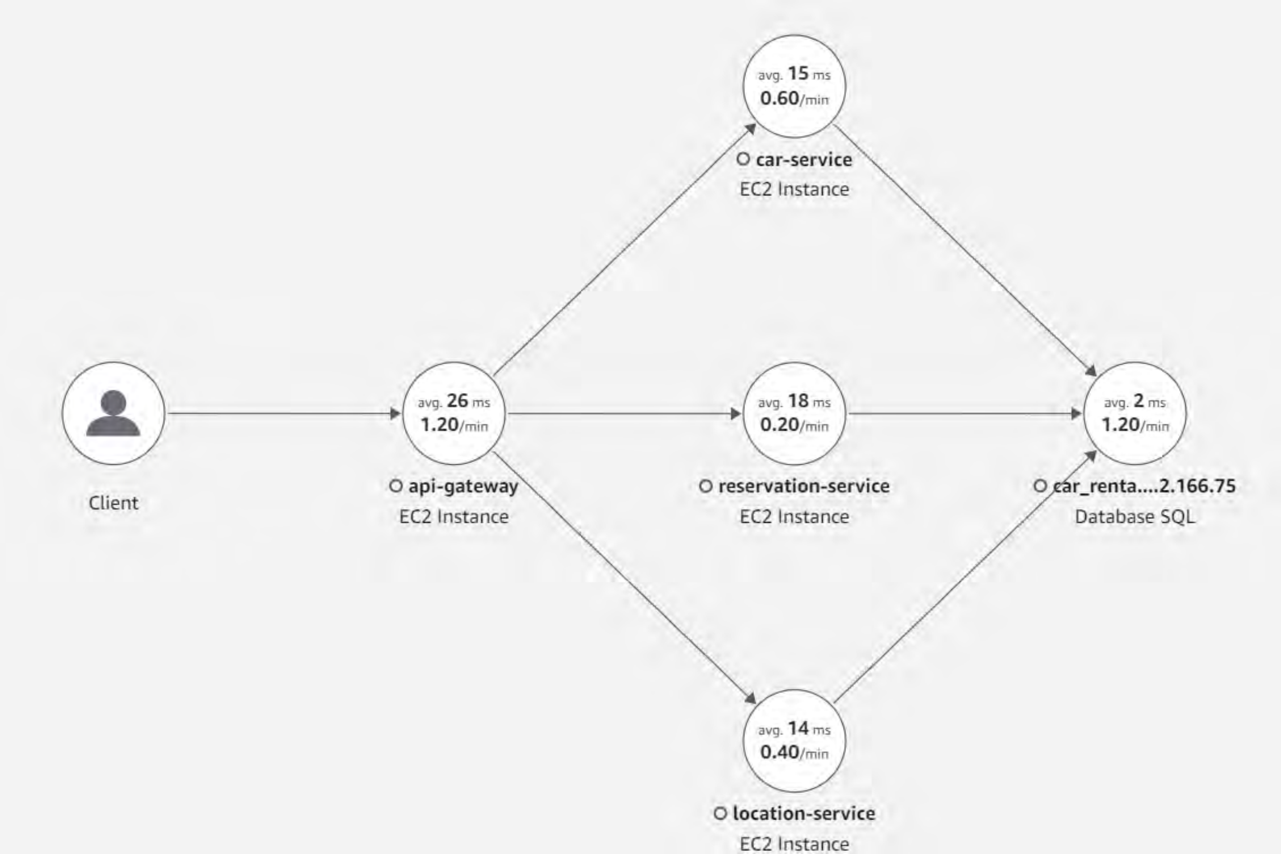
Test case: Simulate a failure of the OCPP handler by powering off the server or disrupting the network connection to the OCPP handler. Observe the behavior of the system and the impact on the EV charging process.

Based on the architecture diagram provided, here's the blast radius for the test case:

1. **OCPP Handler**: The primary component directly impacted by the test case is the OCPP handler. This is the component responsible for handling the communication and control of the EV charging process. By simulating a failure by powering off the server or disrupting the network connection, the OCPP handler will be unable to perform its normal functions.
2. **EV Charging Stations**: The OCPP handler is the central component that communicates with the charging stations. With the disruption of the OCPP handler, the charging stations will not be able to receive instructions or communicate their status. This could lead to a halt in the charging process or unpredictable behavior at the charging stations.
3. **Back-end Services**: The OCPP handler may be integrated with other back-end services, such as a billing system or user management platform, to support the EV charging process. The failure of the OCPP handler could result in disruptions to these services, such as incomplete or inaccurate charging records or difficulties in managing user accounts.
4. **Users**: Users who are currently charging their vehicles or attempting to start a charging session may experience issues. They could encounter errors or unexpected behavior when interacting with the charging stations or the charging management application.

Considering the impact on the key components and their dependencies, the blast radius for this test case is considered medium to high. The specific impact and severity will depend on the implementation and resilience of the

Generative AI in Action: Illuminating Possibilities via Auto generated Service Maps



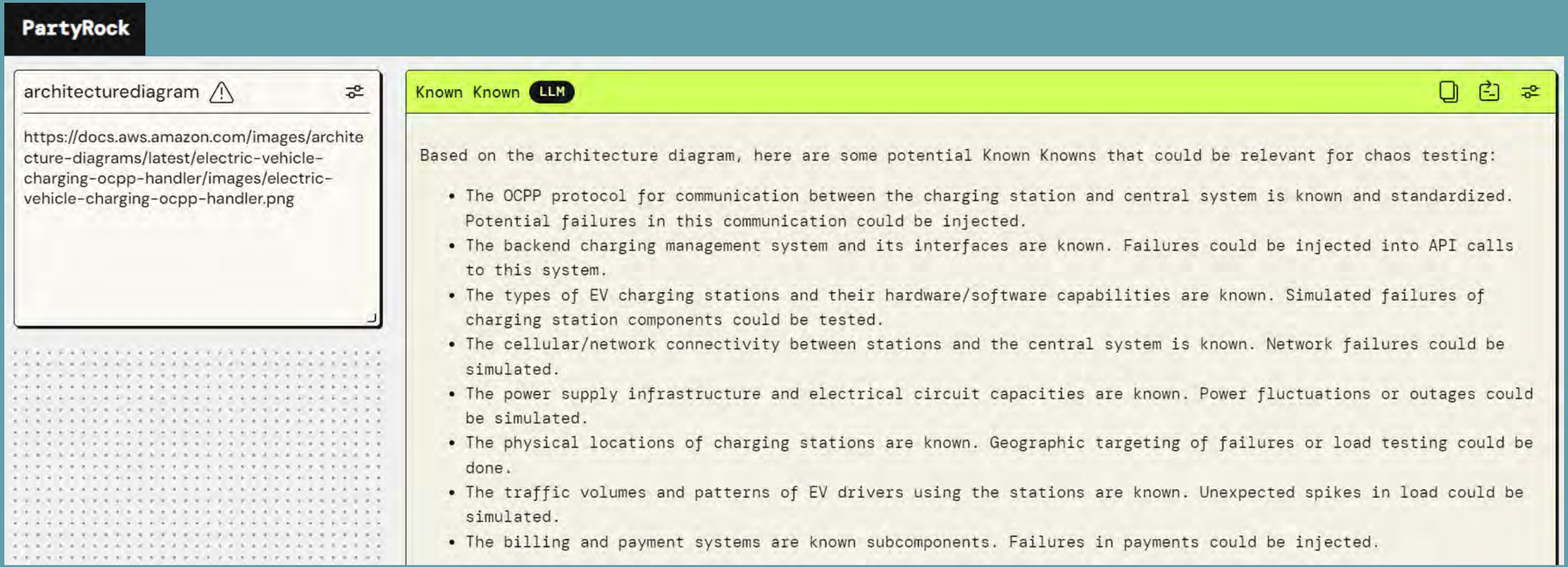
Service Map generated by X-Ray

The Rumsfeld Matrix - "Known Knowns, Known Unknowns, and Unknown Unknowns"

	Known	Unknown
Known	Evaluating components of your systems that are familiar and thoroughly understood, such as system architecture, infrastructure, identified failure points, and CI/CD tests.	Investigating potential issues and vulnerabilities in your system that are known but haven't undergone rigorous testing or validation, such as theoretical vulnerabilities or unverified failure scenarios.
Unknown	Reviewing issues that you are cognizant of but may have been forgotten or overlooked with the passage of time, such as adherence to best practices, documented procedures, or insights from historical incidents.	Conducting comprehensive chaos tests to discover unforeseen and unanticipated vulnerabilities that may emerge unexpectedly, leading to surprises, often of an unpleasant nature.

Rumsfeld Matrix – Known Known with GenAI

Generative AI can assist in analyzing historical data and known patterns, providing insights into well-understood aspects of the system.



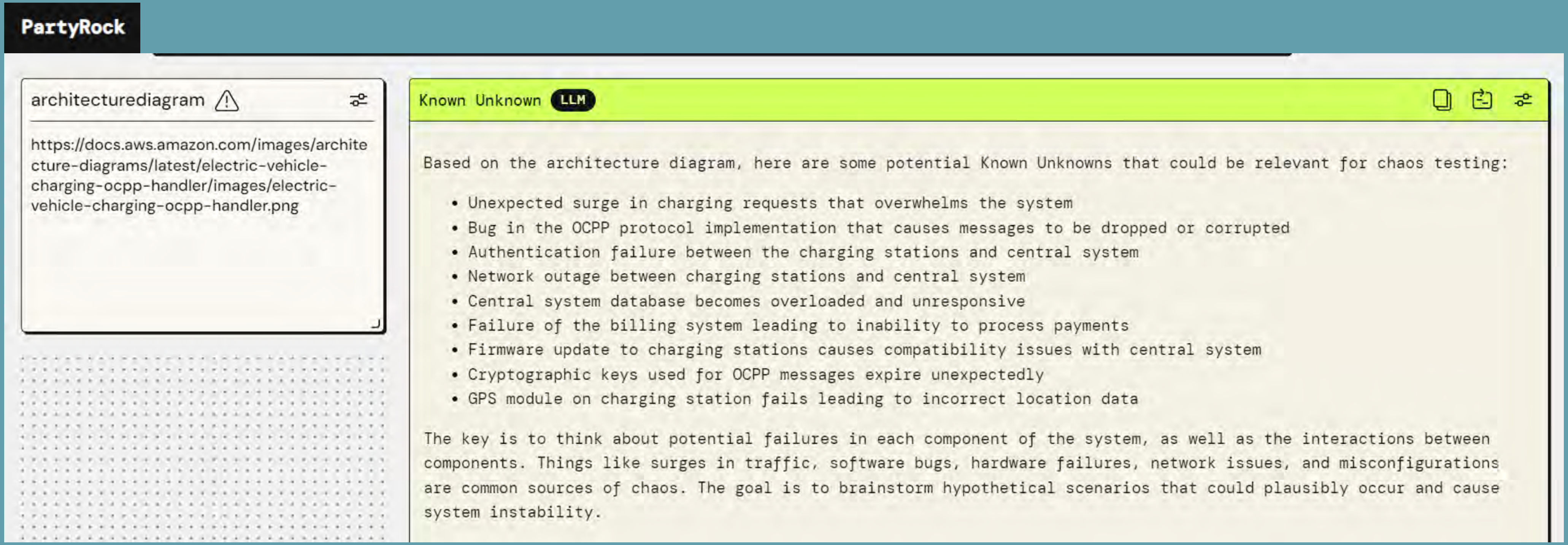
https://docs.aws.amazon.com/images/architecture-diagrams/latest/electric-vehicle-charging-ocpp-handler/images/electric-vehicle-charging-ocpp-handler.png

Based on the architecture diagram, here are some potential Known Knowns that could be relevant for chaos testing:

- The OCPP protocol for communication between the charging station and central system is known and standardized. Potential failures in this communication could be injected.
- The backend charging management system and its interfaces are known. Failures could be injected into API calls to this system.
- The types of EV charging stations and their hardware/software capabilities are known. Simulated failures of charging station components could be tested.
- The cellular/network connectivity between stations and the central system is known. Network failures could be simulated.
- The power supply infrastructure and electrical circuit capacities are known. Power fluctuations or outages could be simulated.
- The physical locations of charging stations are known. Geographic targeting of failures or load testing could be done.
- The traffic volumes and patterns of EV drivers using the stations are known. Unexpected spikes in load could be simulated.
- The billing and payment systems are known subcomponents. Failures in payments could be injected.

Rumsfeld Matrix – Known Unknown with GenAI

By leveraging Generative AI for scenario simulation, it can help identify potential scenarios or conditions that are recognized as uncertainties but have known parameters.



https://docs.aws.amazon.com/images/architecture-diagrams/latest/electric-vehicle-charging-ocpp-handler/images/electric-vehicle-charging-ocpp-handler.png

Based on the architecture diagram, here are some potential Known Unknowns that could be relevant for chaos testing:

- Unexpected surge in charging requests that overwhelms the system
- Bug in the OCPP protocol implementation that causes messages to be dropped or corrupted
- Authentication failure between the charging stations and central system
- Network outage between charging stations and central system
- Central system database becomes overloaded and unresponsive
- Failure of the billing system leading to inability to process payments
- Firmware update to charging stations causes compatibility issues with central system
- Cryptographic keys used for OCPP messages expire unexpectedly
- GPS module on charging station fails leading to incorrect location data

The key is to think about potential failures in each component of the system, as well as the interactions between components. Things like surges in traffic, software bugs, hardware failures, network issues, and misconfigurations are common sources of chaos. The goal is to brainstorm hypothetical scenarios that could plausibly occur and cause system instability.

Rumsfeld Matrix

Unknown Knowns:

This category implies information that is known but not acknowledged. Generative AI may not directly address this category, as it relies on existing data and patterns.

Unknown Unknowns:

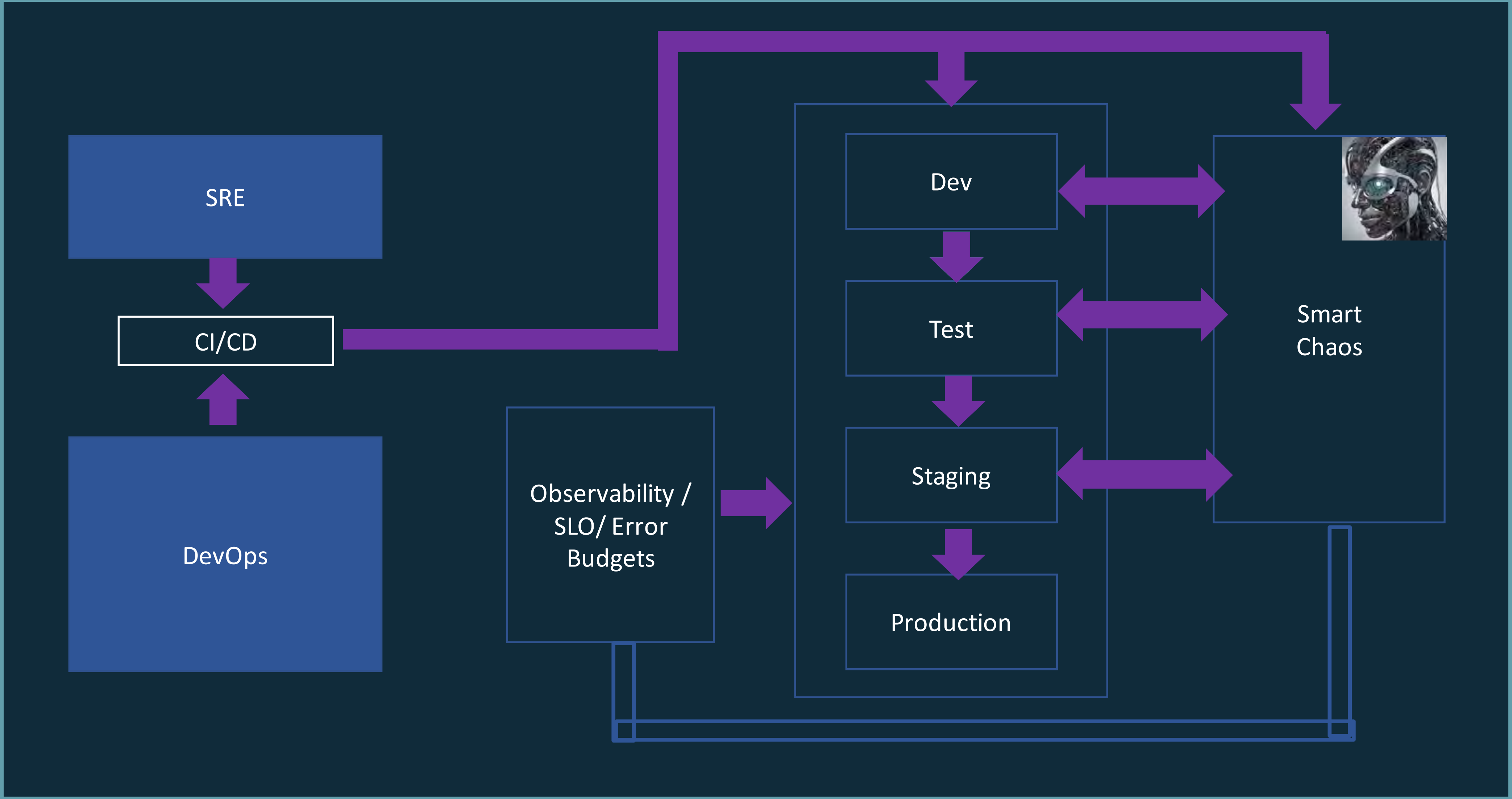
Generative AI's ability to discover hidden patterns in data might contribute to uncovering unknown unknowns to some extent, but complete identification remains challenging.

Smart Chaos : Automating entire Chaos Engineering workflow using GenAI

Source	Build	Test	Deploy	Observability	
AWS CodeCommit	AWS CodeBuild	AWS CodeBuild	AWS CodeDeploy	• CloudWatch Metrics - Collect metrics on utilization, traffic, errors, and more. Create alarms and dashboards to monitor application health. • CloudWatch Logs - Aggregate and monitor application and system logs in one place. • X-Ray - Trace requests to identify performance bottlenecks and errors.	**Chaos Engineering Pipeline**

Build training dataset	Create chaos experiment templates	Generate chaos experiments	Execute experiments via AWS FIS	Evaluate Outcomes	Expand scope iteratively
• Collect telemetry data to establish baseline normal behavior • Log failures, dependencies etc to train models	• API failure, instance termination, induce latency etc • GenAI will combine these building blocks	• Use CloudWatch Service Map to understand dependencies • Leverage GenAI to analyze telemetry data and detect critical flows • Auto-generate experiment combinations using templates to target critical flows	• Inject failure scenarios into non-critical paths first • Slowly increase blast radius to critical components	• Analyze metrics, logs, traces to ensure proper recovery • Feed results back into GenAI to improve chaos targeting	• Increase magnitude and duration of failures • Add more failure scenarios

Smart Chaos : Target end state


SRE

CI/CD

DevOps

Observability / SLO/ Error Budgets

Dev

Test

Staging

Production

Smart Chaos

Best practices and potential pitfalls

Best Practices:

- Quality Data: Ensure diverse, high-quality training data for GenAI.
- Flexible Templates: Design adaptable chaos experiment templates.
- Gradual Expansion: Increase chaos from non-critical to critical components gradually.
- Feedback Loop: Feed results back to GenAI for continuous improvement.
- Operational Collaboration: Collaborate with operations teams for real-world alignment.

Pitfalls to Avoid:

- Watch for Bias: Be cautious of biased training data.
- Balance Flows: Consider vulnerabilities in both critical and non-critical paths.
- Controlled Expansion: Avoid rapid chaos expansion to critical components.
- Thorough Analysis: Analyze metrics, logs, and traces post-experiment, including recovery.
- Human Expertise Matters: GenAI is a tool; include human judgment for a holistic approach.

Thank you!