AI-DRIVEN CHAOS ENGINEERING:

AUTOMATING RESILIENCE TESTING WITH PREDICTIVE INSIGHTS

SRIMAAN YARRAM

INTRODUCTION TO CHAOS ENGINEERING

Definition: Intentional introduction of failures into a system to test its resilience.

Purpose: Identify weaknesses before they manifest in production.

Key Principle: “Embrace failure to build robust systems.”

CHALLENGES IN PREDICTING REAL-TIME FAILURES

Unpredictable Failures: Despite extensive testing, unforeseen issues arise in live environments.

Complexity: Modern systems intricacies make it hard to anticipate all failure modes.

Resource Intensive: Continuous testing can be time-consuming and costly.

EVOLVING FROM REACTIVE TO PROACTIVE APPROACHES

Traditional Approach

- Manual Fault Injection
- Knowledge-Based Scenarios
- Limited Real-World Insights
- Partial Automation
- Static Experimentation
- Reactive Approach

AI-Driven Approach

- Automated Fault Injection
- Adaptive & Dynamic Tests
- Real-World Data-Driven Testing
- Full AI Automation
- Continuous Learning & Improvemen
- Predictive & Proactive

INTEGRATING AI TO PREDICT AND AUTOMATE FAILURES

AI Analysis: Utilize machine learning to analyze historical logs.

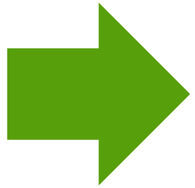
Predictive Modeling: Forecast potential failure zones.

Automated Injection: Deploy targeted failures based on AI insights.

KEY OBSERVABILITY TOOLS AND LOGS

- New Relic: Application performance monitoring.
- Splunk: Log aggregation and analysis.
- AWS CloudWatch: Infrastructure and application monitoring.
- Purpose: Collect comprehensive data for AI-driven insights.

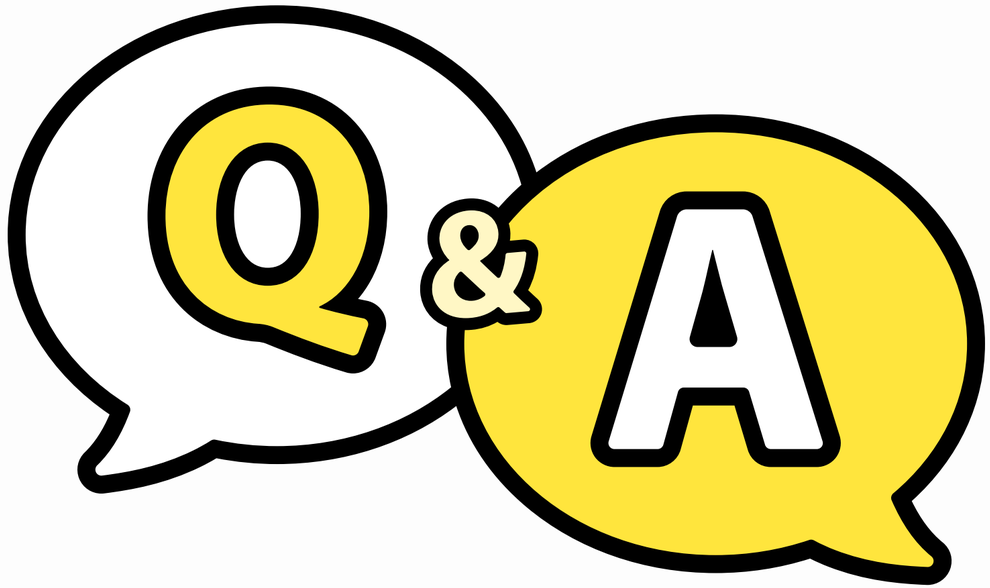
```python
# Initialize S3 client
s3_client = boto3.client('s3')
bucket_name, log_file_key = 'your-s3-bucket', 'logs/system_logs.csv'

# Fetch log file from S3
obj = s3_client.get_object(Bucket=bucket_name, Key=log_file_key)
df = pd.read_csv(obj['Body'])
df['timestamp'] = pd.to_datetime(df['timestamp'])

# Define feature columns for anomaly detection
features = ["cpu_usage", "memory_usage", "response_time"]

# Apply Isolation Forest model to detect anomalies
df["anomaly"] = IsolationForest(contamination=0.05, random_state=42).fit_predict(df[features])
df["anomaly"] = df["anomaly"].map({1: "Normal", -1: "Anomaly"})

# Filter detected anomalies
detected_anomalies = df[df["anomaly"] == "Anomaly"]

# Create a fault injection experiment based on detected anomalies
experiment = {
    "description": "AI-driven fault injection",
    "targets": [{
        "resource": "EC2", "action": "cpu_stress", "duration": "5m", "intensity": "high"
    } for _ in range(len(detected_anomalies))],
    "conditions": {"trigger": "when anomaly detected in logs"}
}

# Save experiment configuration to a JSON file
with open("fault_injection_experiment.json", "w") as f:
    json.dump(experiment, f, indent=4)

# Plot anomalies in response time
plt.figure(figsize=(10, 5))
plt.scatter(df["timestamp"], df["response_time"], c=(df["anomaly"] == "Anomaly"), cmap="coolwarm", label="Anomalies")
plt.xlabel("Timestamp"), plt.ylabel("Response Time (ms)")
plt.title("AI-Driven Log Analysis: Detecting Anomalies")
plt.legend(), plt.xticks(rotation=45), plt.show()

# Display the detected anomalies in a user-friendly format
import ace_tools as tools
tools.display_dataframe_to_user(name="Detected Log Anomalies", dataframe=detected_anomalies)
```



```json
{
  "description": "AWS FIS experiment to stop and start an EC2 instance",
  "targets": {
    "TargetInstances": {
      "resourceType": "aws:ec2:instance",
      "selectionMode": "ALL",
      "filters": [
        {
          "path": "tags.Name",
          "values": ["test-instance"]
        }
      ]
    }
  },
  "actions": {
    "StopInstance": {
      "actionId": "aws:ec2:stop-instances",
      "parameters": {
        "force": "true"
      },
      "targets": {
        "Instances": "TargetInstances"
      }
    },
    "StartInstance": {
      "actionId": "aws:ec2:start-instances",
      "targets": {
        "Instances": "TargetInstances"
      }
    }
  },
  "schedule": {
    "StopInstance": {
      "after": "60s"
    }
  },
```

SUMMARIZING THE BENEFITS OF AI-DRIVEN CHAOS ENGINEERING

- Proactive Resilience: Anticipate and mitigate potential failures.
- Efficiency: Streamlined testing through automation.
- Continuous Improvement: Leverage AI insights for ongoing system enhancement.


Q&A