Chaos Engineering for Serverless Architectures

Jason Barto

Principal Solutions Architect, AWS
@ jasbarto

Outline

- Common Faults
- Fault Injection Techniques
  - Source Code
  - Environment
  - Network
  - Configuration
- Demonstration

The Nature of Failure and its simulation

Common Faults

Resource Exhaustion

- CPU Stress
- Memory
- Disk Space
- Disk Bandwidth

Network Dependency Disruption

- Latency
- Bandwidth
- Packet loss
- Failure to connect
- 4XX / 5XX HTTP Response

Not all failures are binary

Fault Injection Types

Code Manipulation

Environment Manipulation

Network Manipulation

Application

Compute Environment

Network

Fault Injection Techniques

	Code Manipulation	Environment Manipulation	Network Manipulation
COTS on EC2	No	Yes	Yes
COTS on Serverless	No	No	No
Application on EC2	Yes	Yes	Yes
Application on Serverless	Yes	No	Yes

Fault Injection Techniques

	Code Manipulation	Environment Manipulation	Network Manipulation
COTS on EC2	No	Yes	Yes
COTS on Serverless	No	No	No
Application on EC2	Yes	Yes	Yes
Application on Serverless	Yes	No	Yes

Fault Injection Techniques

	Code Manipulation	Environment Manipulation	Network Manipulation
COTS on EC2	No	Yes	Yes
COTS on Serverless	No	No	No
Application on EC2	Yes	Yes	Yes
Application on Serverless	Yes	No	Yes

We need configuration manipulation

Source Code Manipulation

Common Faults

- 4XX / 5XX API Responses
- Disk exhaustion
- Message corruption
- Network Latency

Common Tools

- AWS Java SDK Request Handlers
- Failure Lambda
- AWS Lambda Chaos Injection

Environment Manipulation

Common Faults

- CPU Stress
- Memory exhaustion
- Open file exhaustion
- Disk space exhaustion
- Disk Bandwidth throttling
- Network throttling

Common Tools

- Stress-NG
- fallocate / dd
- Traffic Control
- CPUStres
- TestLimit

Network Manipulation

Common Faults

- TCP Packet Loss
- Bandwidth Limitation
- Network Latency
- No Connectivity

Common Tools

- Security Groups
- Network Access Control Lists
- Network Firewall
- HTTP Proxy
- NAT Instance

Configuration Manipulation

Common Faults

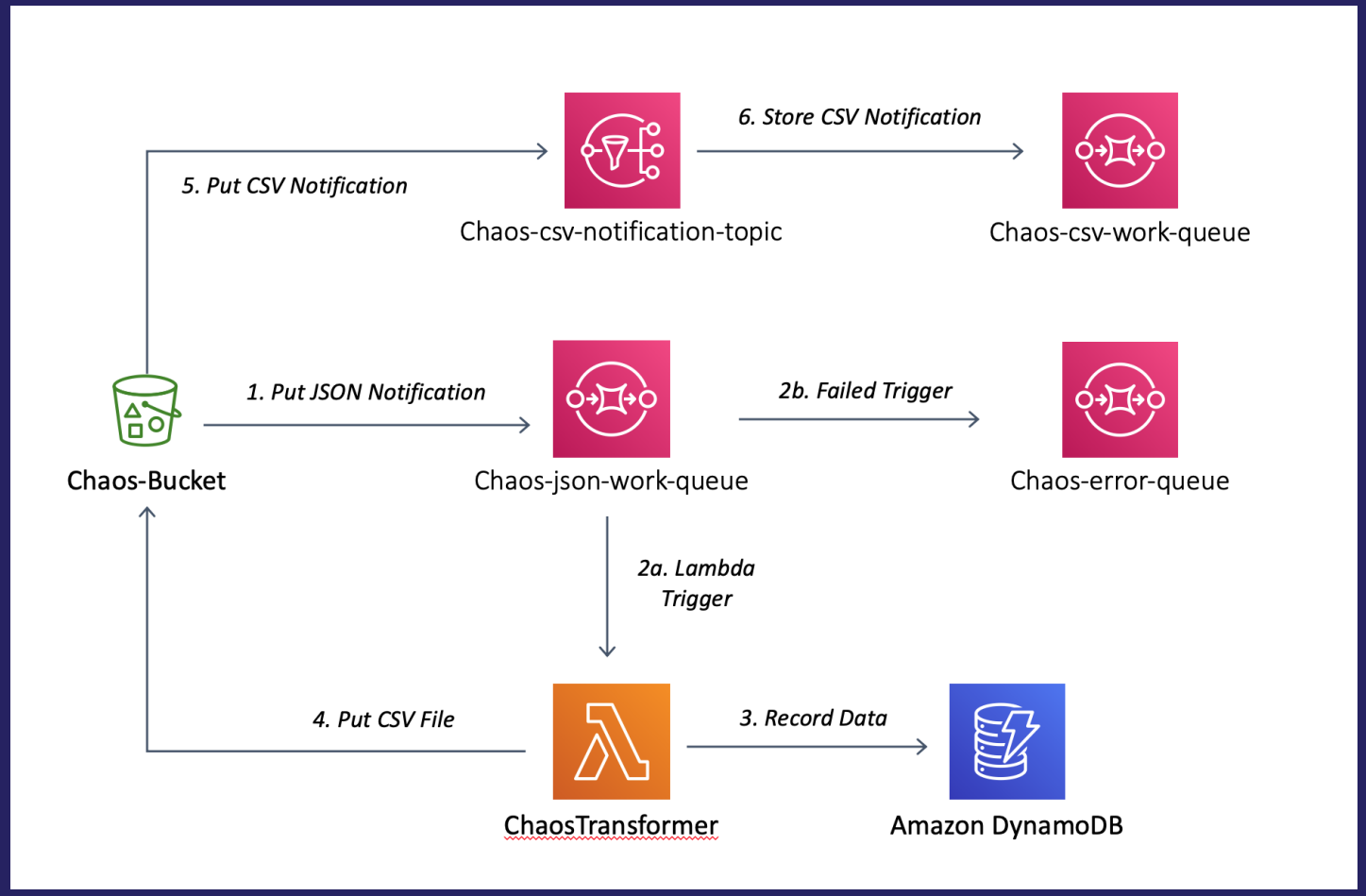
- SQS cannot call AWS Lambda
- Kinesis is not able to fulfill a request from CloudWatch

Common Tools

- Resource Policies
- IAM Policies
- VPC Attachment

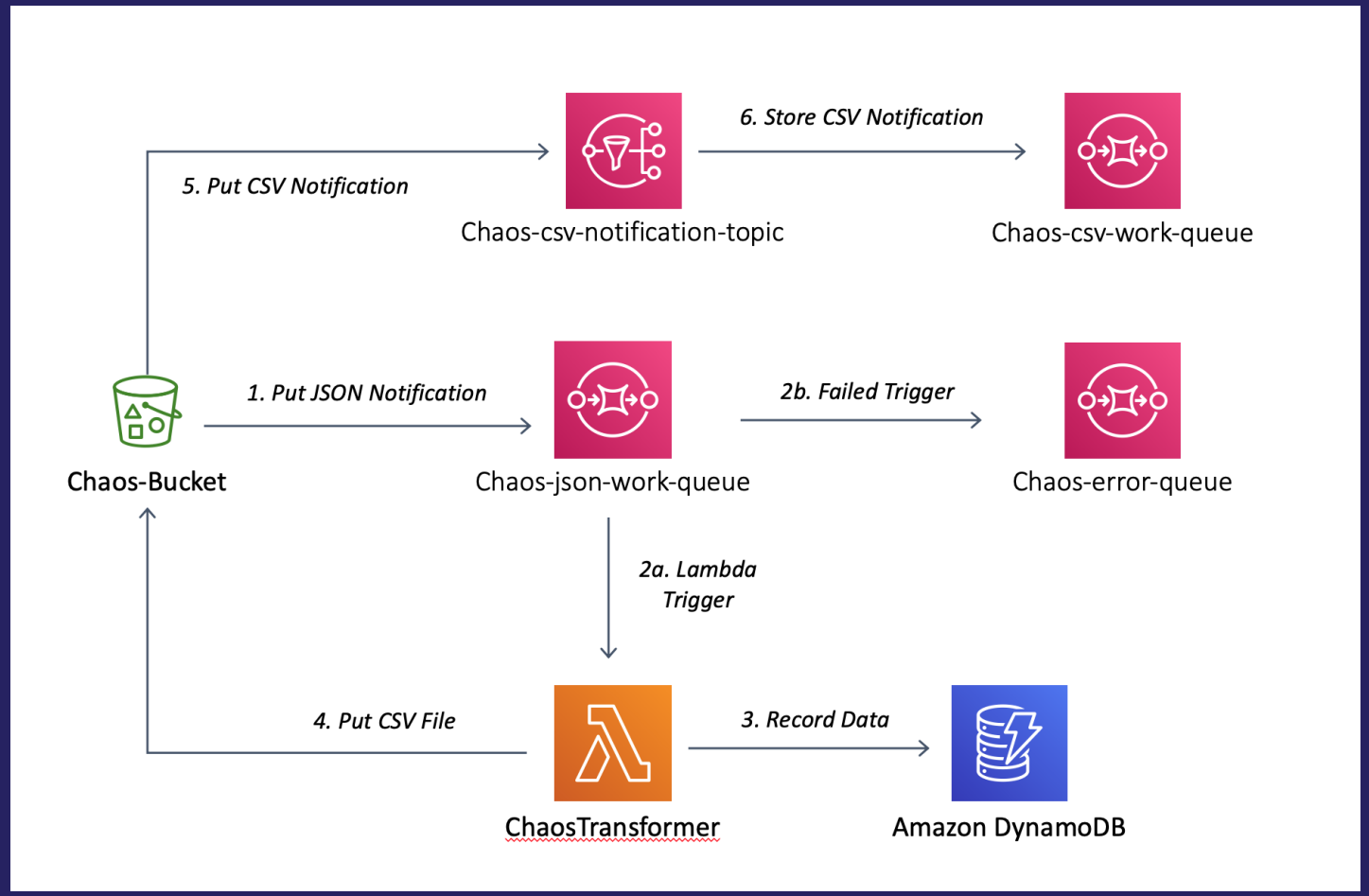
Time to Experiment

Serverless Data Transformation



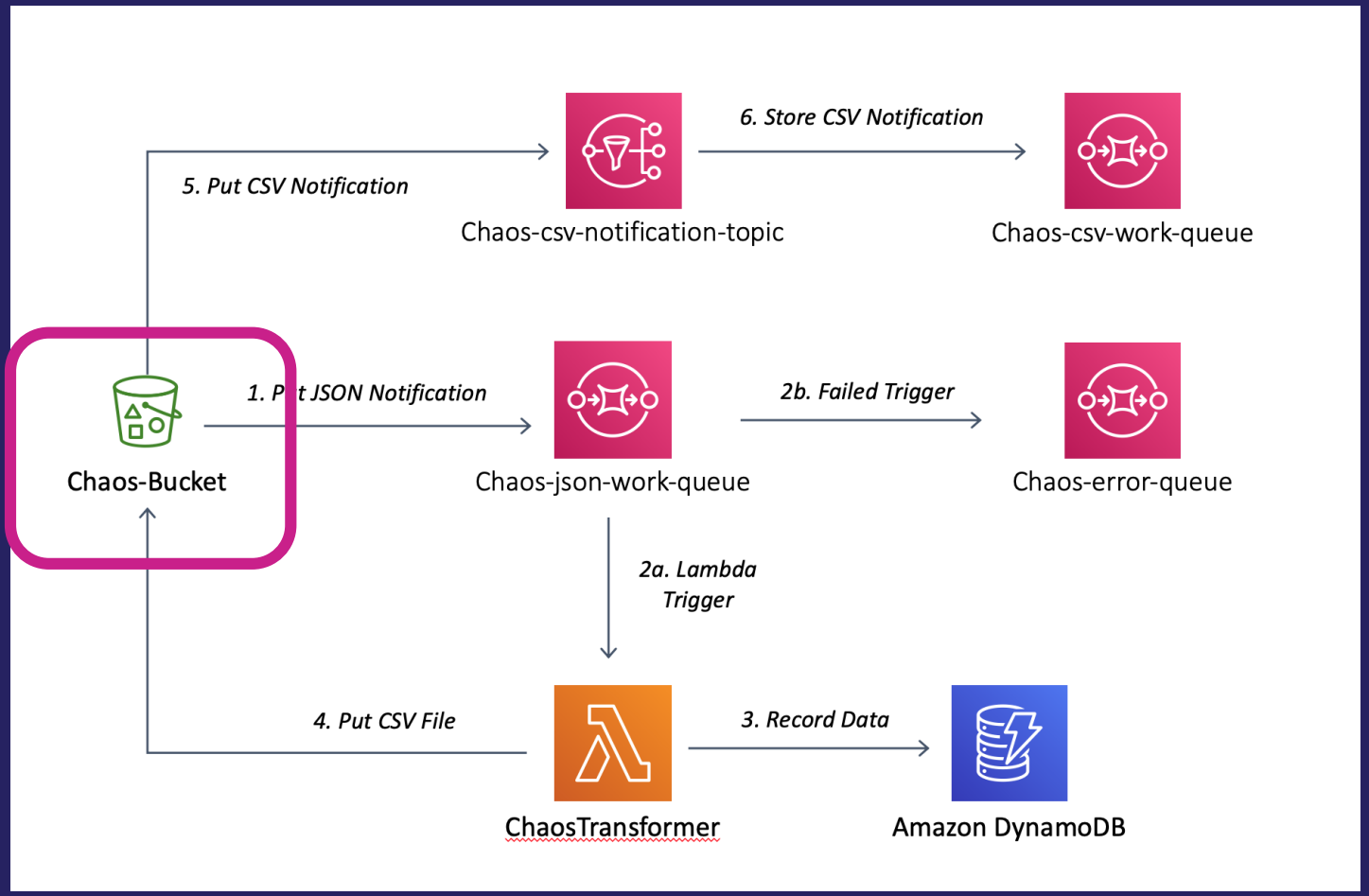
Serverless Data Transformation

- Convert JSON to CSV



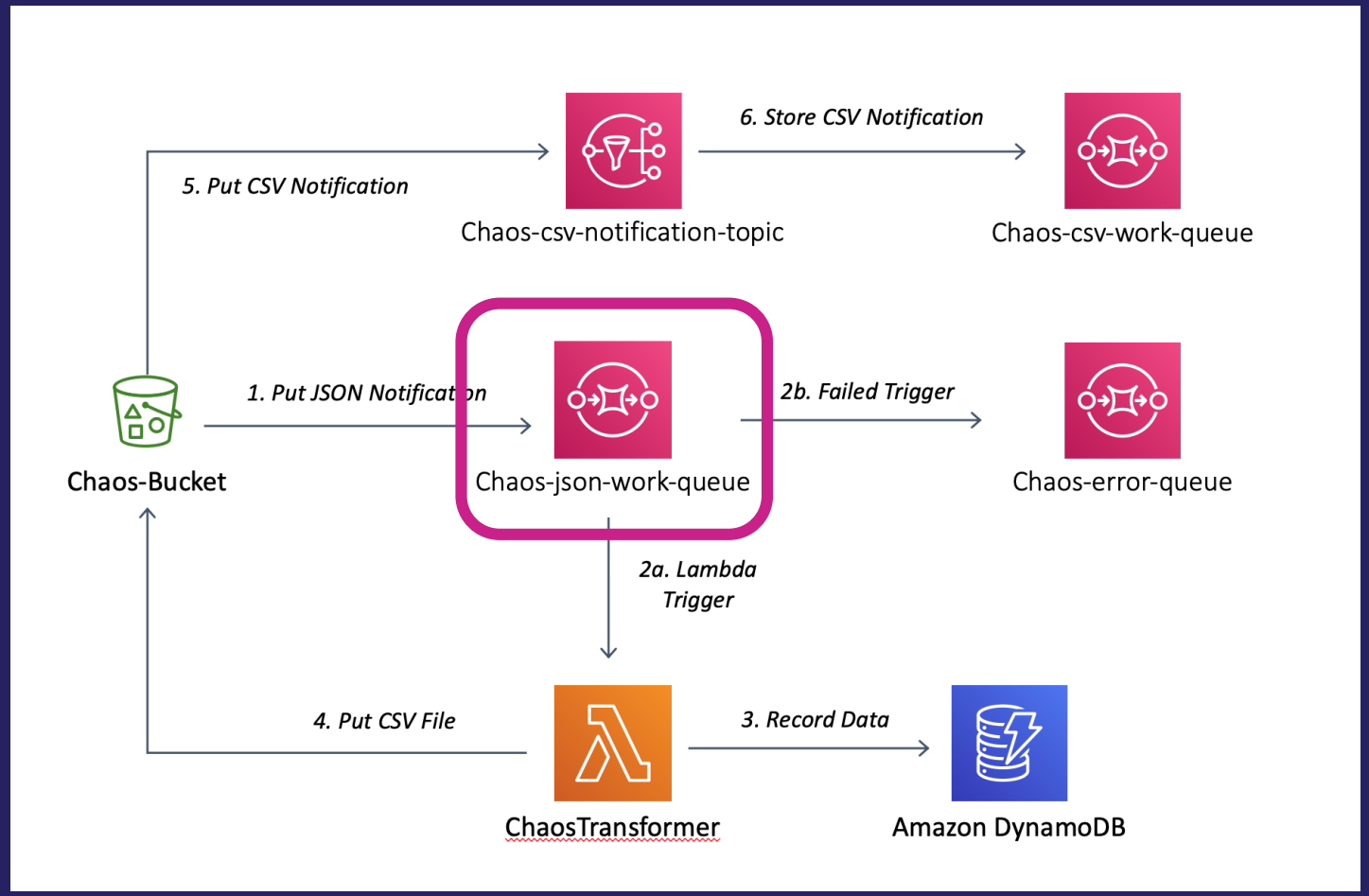
Serverless Data Transformation

- Convert JSON to CSV
- JSON Objects written to Chaos-Bucket



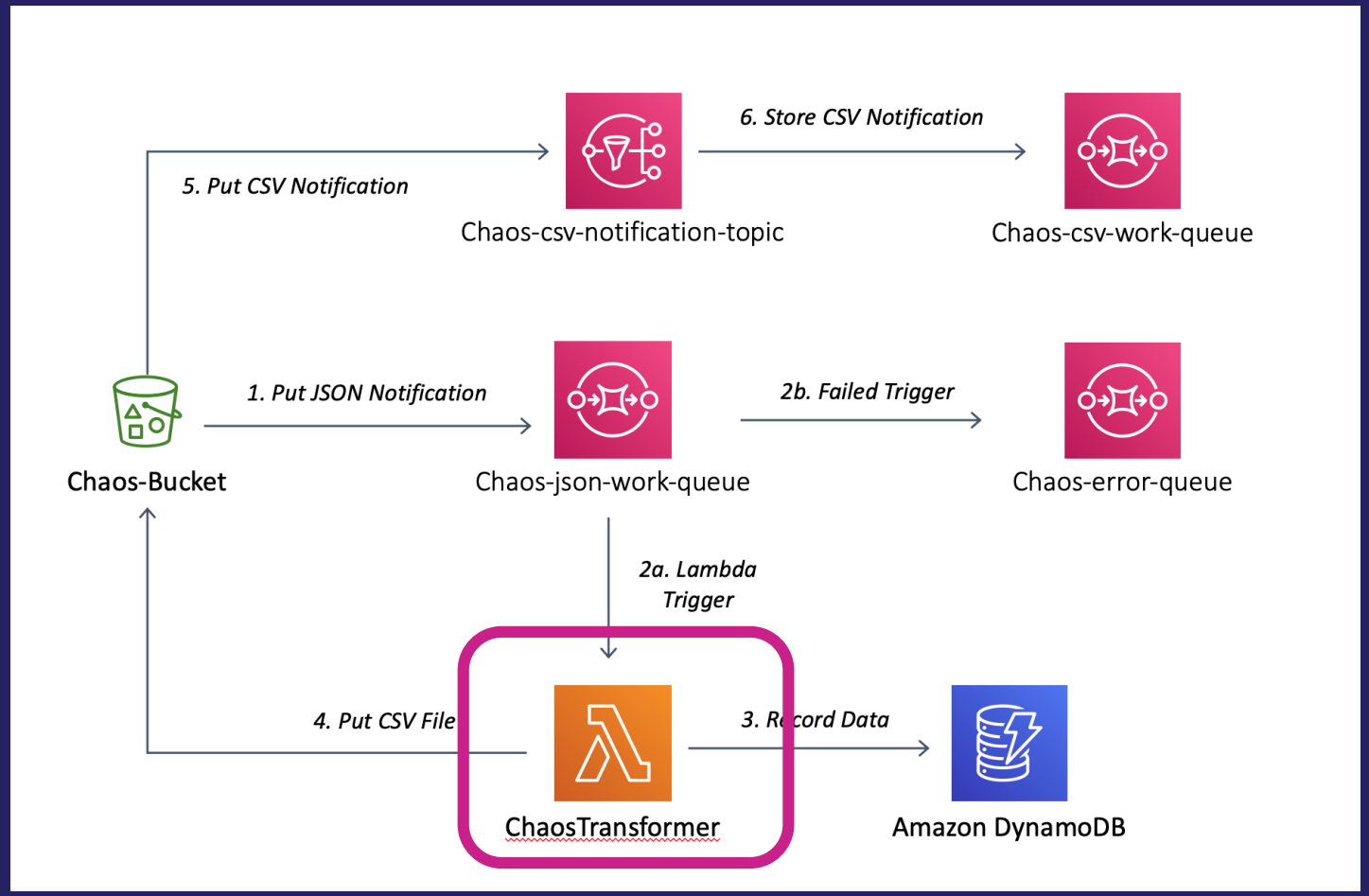
Serverless Data Transformation

- Convert JSON to CSV
- JSON Objects written to Chaos-Bucket
- Notifications sent via SQS



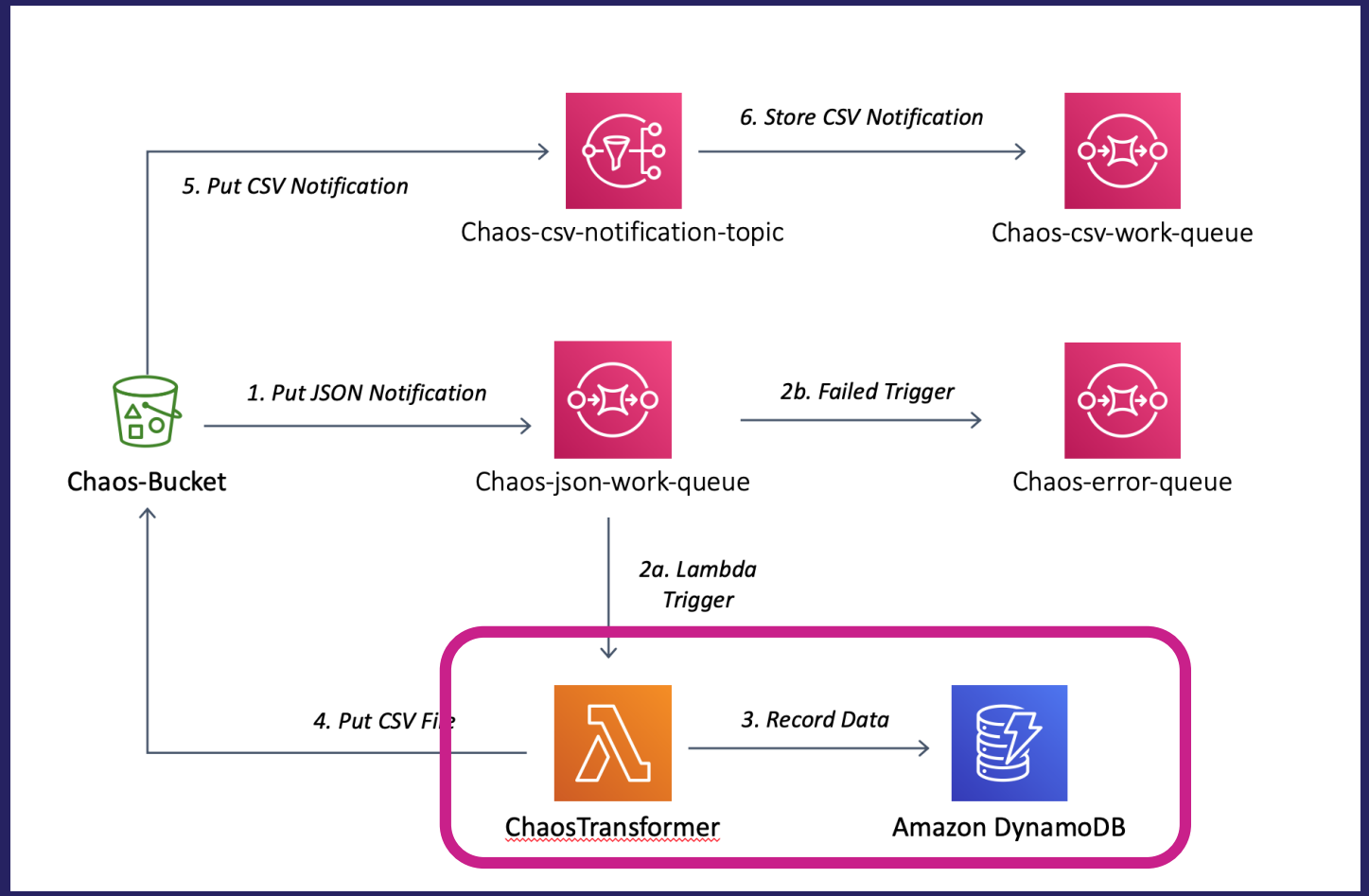
Serverless Data Transformation

- Convert JSON to CSV
- JSON Objects written to Chaos-Bucket
- Notifications sent via SQS
- Converted CSV Objects written to Chaos-Bucket



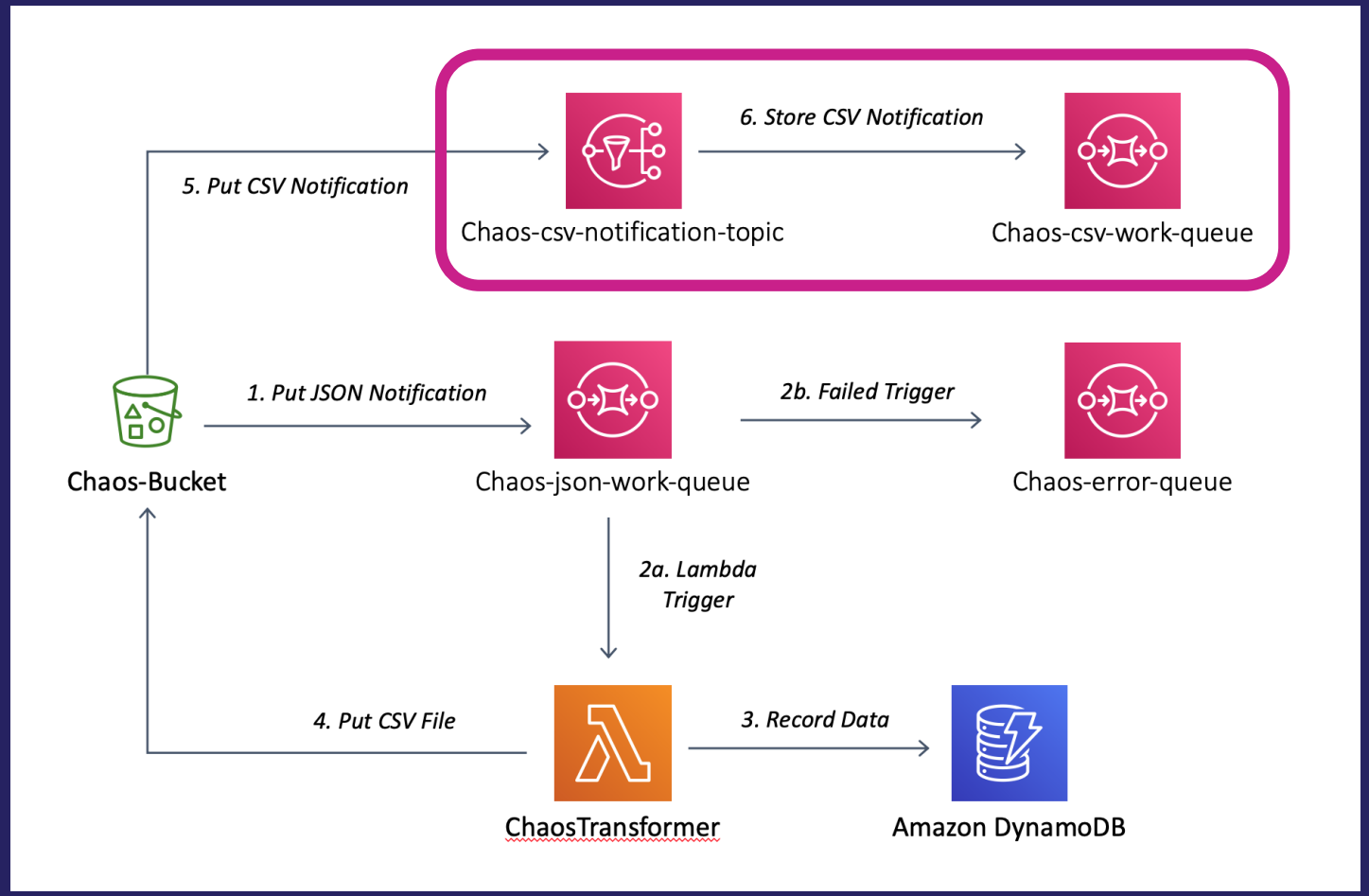
Serverless Data Transformation

- Convert JSON to CSV
- JSON Objects written to Chaos-Bucket
- Notifications sent via SQS
- Converted CSV Objects written to Chaos-Bucket
- Transform recorded to DynamoDB



Serverless Data Transformation

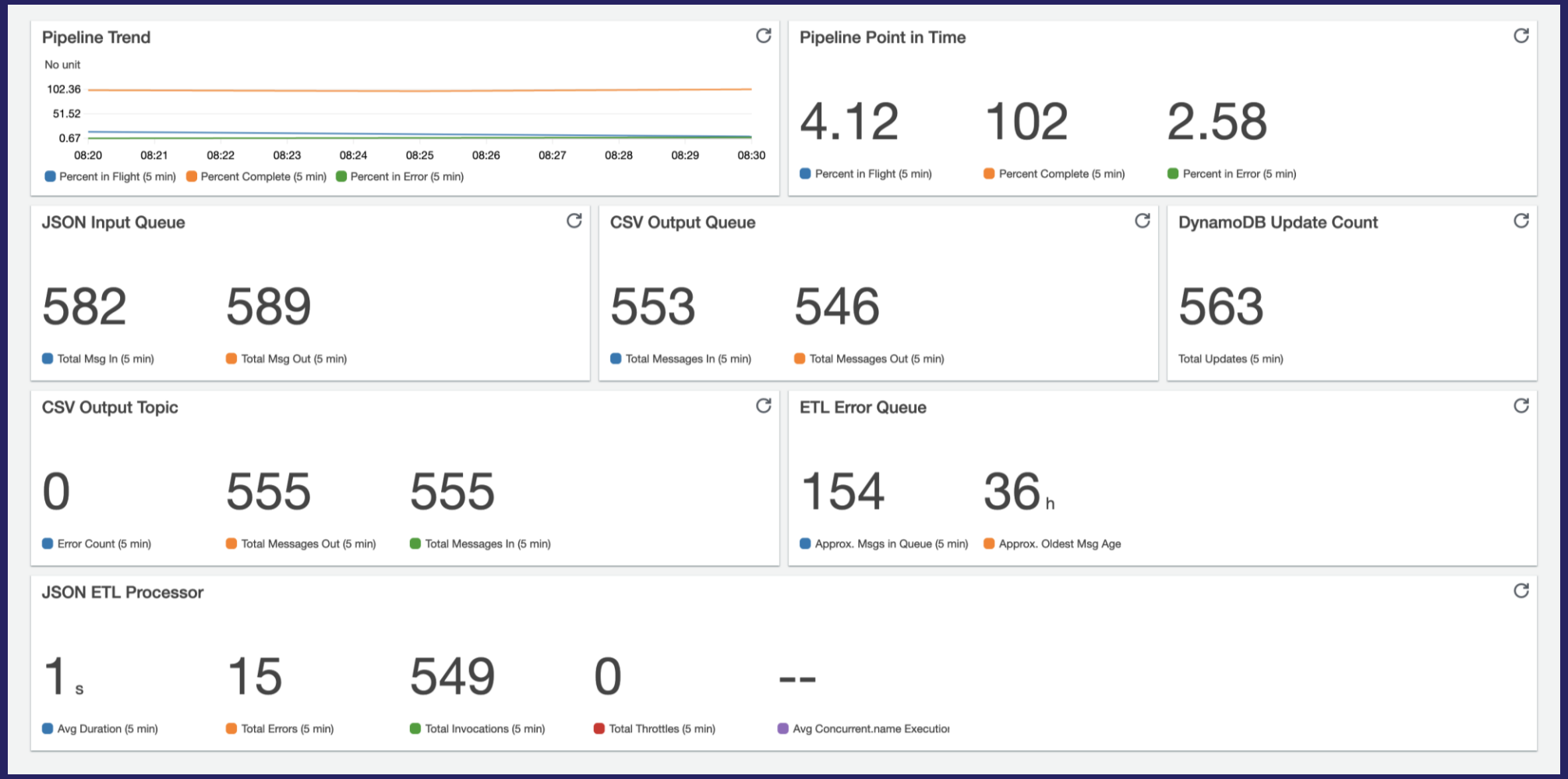
- Convert JSON to CSV
- JSON Objects written to Chaos-Bucket
- Notifications sent via SQS
- Converted CSV Objects written to Chaos-Bucket
- Transform recorded to DynamoDB
- Notifications sent via SNS



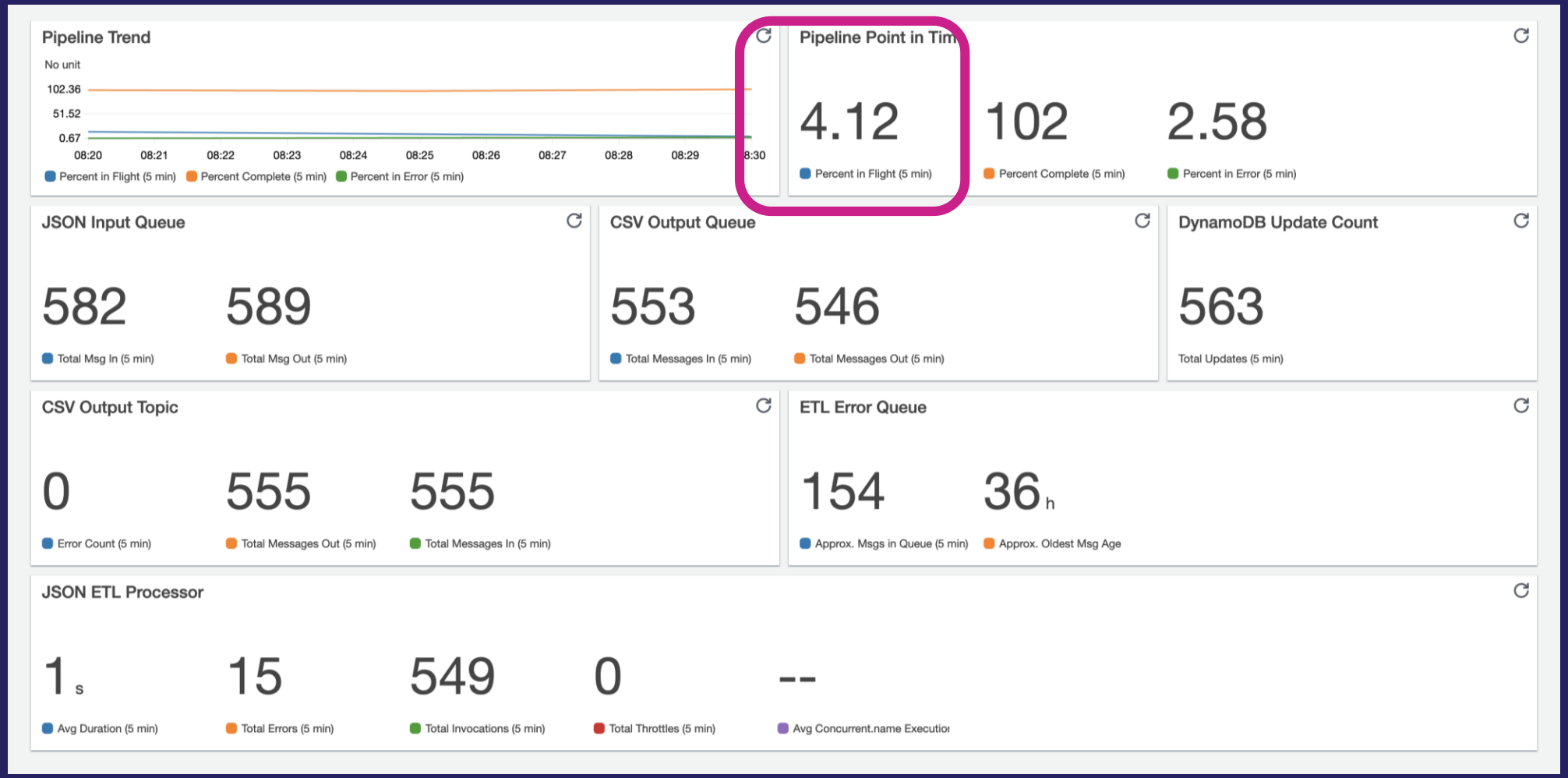
Service Health Measurements

Service Level Indicators		Service Level Objectives
% of messages currently being processed	→	No more than 80% of messages should be in flight
% of messages that have been processed	→	Between 90 and 100% of messages have been processed
% of messages that could not be processed	→	No more than 5% of messages failed processing

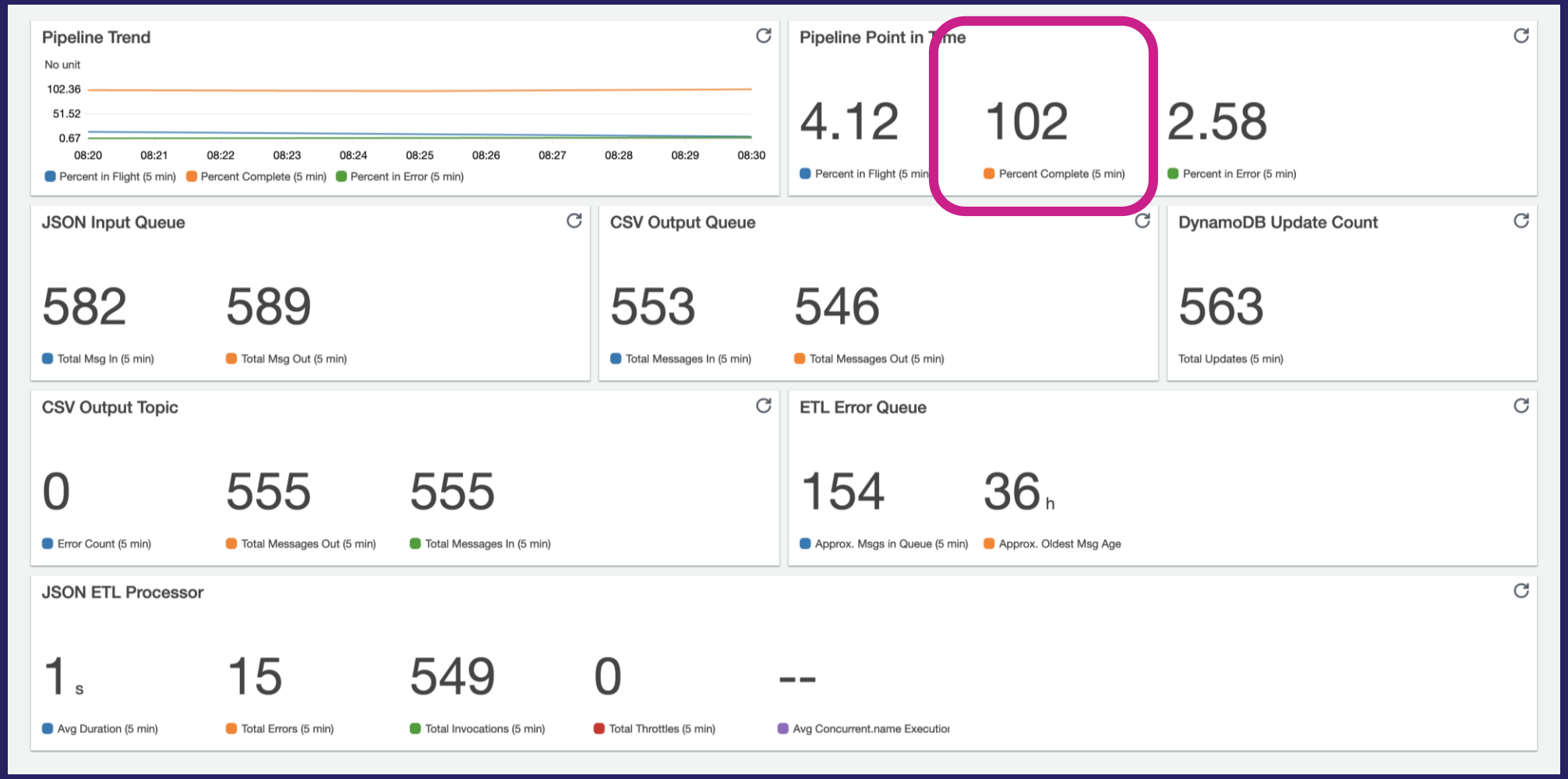
Monitoring



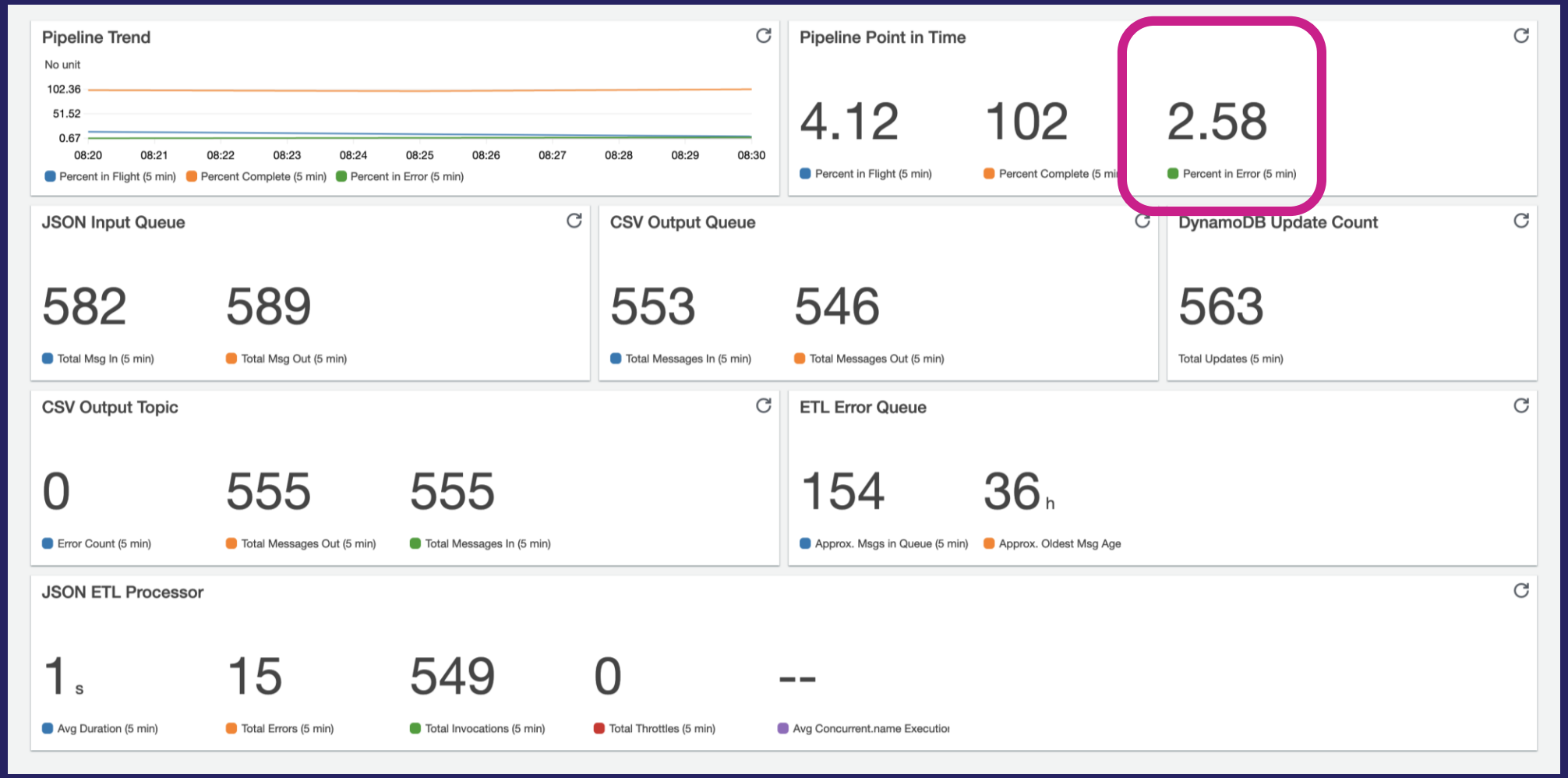
Monitoring



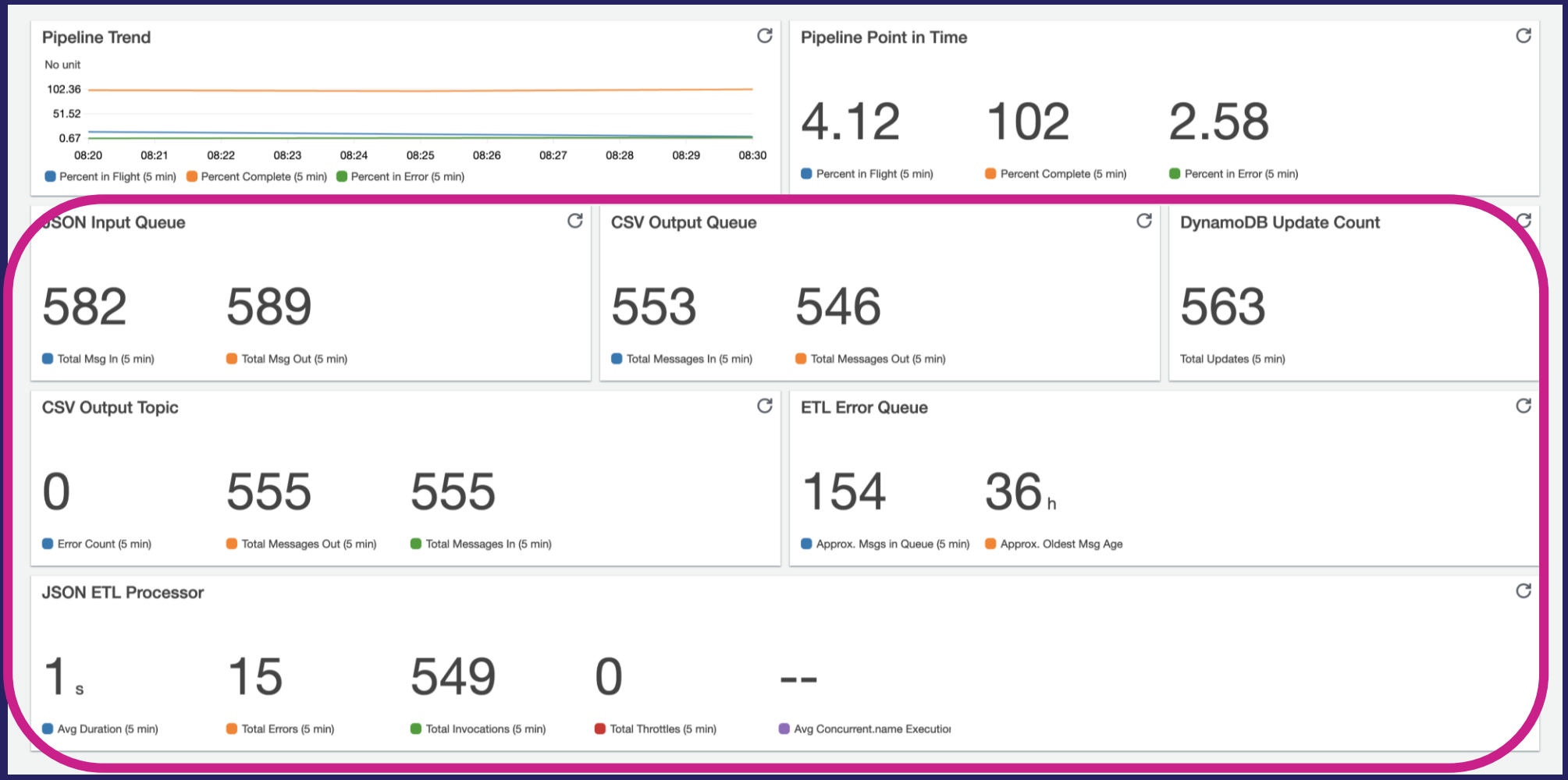
Monitoring



Monitoring



Monitoring



Experiment 1 – Configuration Manipulation

Serverless Manipulation Experiment

Hypothesis	**When SQS invocation of Lambda is disrupted the SLO for messages in flight will not be exceeded.**
Fault Simulated	Disruption of Lambda control plane
Service Level Indicators	• % messages in flight, SLO (< 80%)
Method	1. Place system under steady load of 120 documents per minute 2. Set Reserved Concurrency for data processing Lambda function to 0 3. Observe system for 5 minutes 4. Reset Reserved Concurrency limit 5. Wait for steady state to return

DEMONSTRATION

Experiment 2 – Source Code Manipulation

Source Code Manipulation Experiment

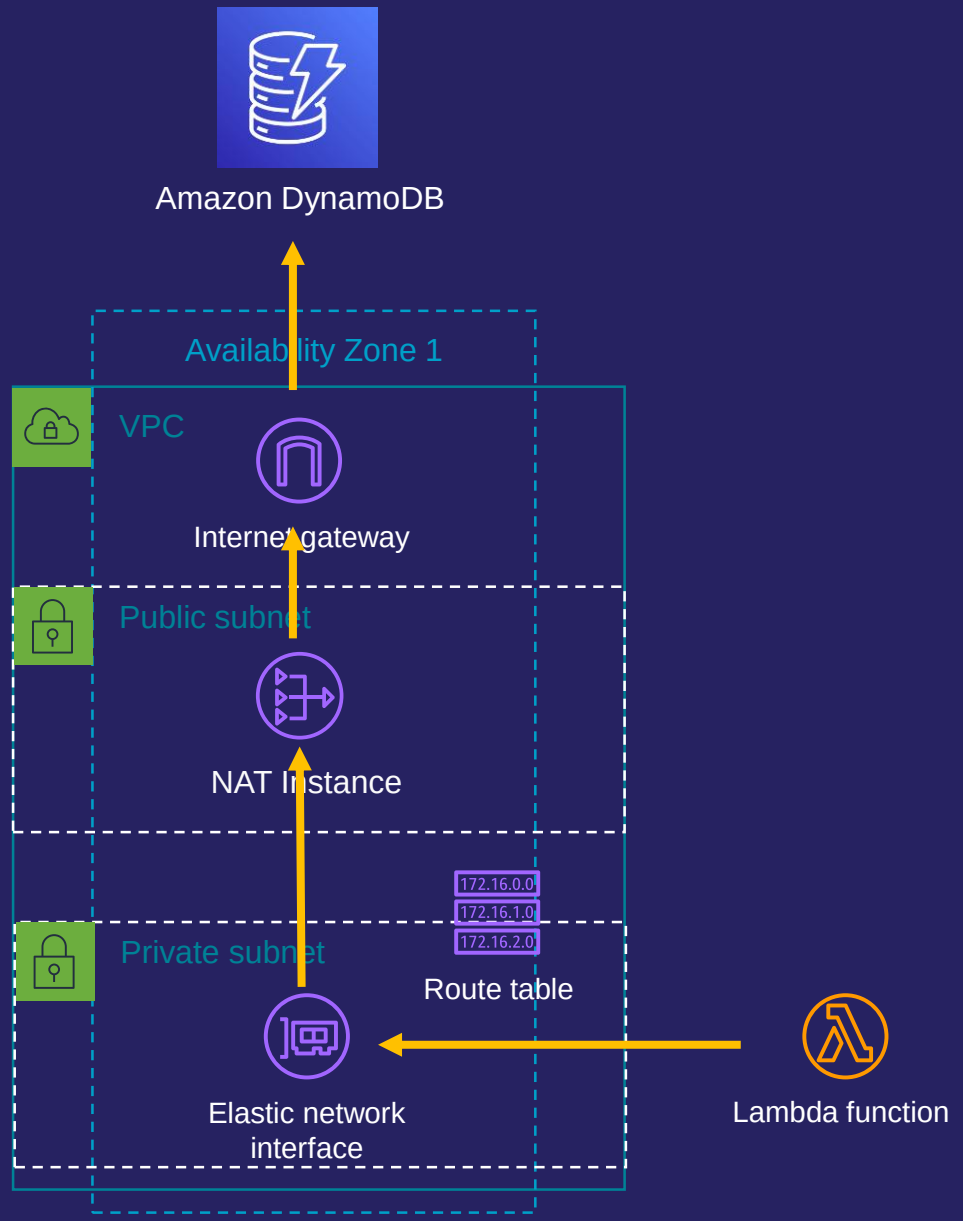
Hypothesis	**Denying 50% of the requests to DynamoDB will not cause the system to drop below 90% successful processing rate.**
Fault Simulated	• Network disruption between Lambda and DynamodDB
Service Level Indicators	• % messages complete, SLO (90% < 100%)
Method	1. Have failure-lambda library embedded in Lambda function 2. Apply a steady state load of 120 files per minute 3. Set Parameter Store parameter to configure denying DynamoDB access from 50% of Lambda invocations 4. Observe system for 5 minutes 5. Reset Parameter Store parameter 6. Wait for steady state to return

DEMONSTRATION

Experiment 3 – Network Manipulation

Network Architecture

- Bind Serverless services to VPC
- Use routing tables to direct traffic to a NAT instance
- NAT instance using iptables, tc, and HTTP proxy
- Iptables directs HTTP traffic to the HTTP proxy
- Iptables enables masquerading
- tc (Traffic Control) throttles / delays / loses TCP packets



Network Manipulation Experiment

Hypothesis	**The service will keep % complete above 90% when up to 70% of TCP packets to DynamoDB are lost from Availability Zone A.**
Fault Simulated	• Network disruption between Lambda and DynamoDB
Service Level Indicators	• % messages complete(90% < 100%)
Method	1. Bind the Lambda function to Availability Zone A 2. Deploy a NAT instance with a transparent proxy to filter traffic for DynamoDB 3. Use a route table to direct requests for DynamoDB through the NAT instance 4. Place the system under steady load of 120 files per minute 5. Use Traffic Control (tc) to cause 70% packet loss for traffic to DynamoDB 6. Observe system for 5 minutes 7. Clear packet loss for traffic to DynamoDB 8. Wait for steady state to be achieved

DEMONSTRATION

Recap

Common Faults

Fault Injection Techniques

- Source Code Manipulation
- Environment Manipulation
- Network Manipulation
- Configuration Manipulation

Demonstration

Available Tools

- NAT instance / network proxy
  - Traffic Control
  - HTTP Proxy
- Failure Lambda library
- AWS Lambda Chaos Injection Library
- Service configuration

Further Reading

Paper: Building Mission Critical Financial Services Applications on AWS

Blog post: Failure Modes and Continuous Resilience
medium.com/@adrianco

Blog post: Towards Continuous Resilience
medium.com/@adhorn

Workshop: Resilience Engineering Workshop
resilience.workshop.aws

Workshop: Well Architected Labs – Reliability
wellarchitectedlabs.com/reliability

Thank you!

Jason Barto

Principal Solutions Architect, AWS
@jasbarto