AWS re:Invent

DECEMBER 2 – 6, 2024 | LAS VEGAS, NV

DEV339

Supercharge Lambda functions with Powertools for AWS Lambda

Raphael Manke

(he/him)
Senior IT Consultant, Cloud
codecentric AG

Amazon CloudWatch alarm

aws APP 7:40 PM

Today

CloudWatch Alarm | Failed-storing-notification-id-crmAlarm | eu-central-1 | Account: 296062566320

Threshold Crossed: 1 datapoint [1.0 (17/10/24 17:35:00)] was greater than the threshold (0.0).

List dashboards | Query logs | ⋮

Namespace
customerNotifications

Metric
failed-storing-notification-id-crm

Timestamp
Thu, 17 Oct 2024 17:40:55 UTC

Alarm Description
This alarm monitors the 'failed-storing-notification-id-crm' metric emitted by the 'notification-service'.
The alarm is triggered when the metric value is greater than 0, indicating a failure in customer notifications.

To query the logs for the data points that triggered the alarm, use the following CloudWatch Logs Insights query:

```
  fields @timestamp, @message,
x_ray_trace_id
  | filter @message like /failed-
storing-notification-id-crm/
  | sort @timestamp desc
```

See less

See more

Query the logs

Logs Insights Info

Start tailing | 5m | 30m | 1h | 3h | 12h | Custom | Compare (Off) | UTC timezone

Select log groups, and then run a query or choose a sample query.

Select up to 50 log groups.

Browse log groups

/aws/lambda/ReinventPowertoolsStack-queueHandlerFunction7800B2-mT09jABbKG9A

Clear all

```
fields @timestamp, @message, xray_trace_id
  | filter @message like /failed-storing-notification-id-crm/
  | sort @timestamp desc
```

Query generator

Run query | Cancel | Save | History

Discovered fields

Queries

Query the logs

Logs (13)

Export results ▼ | Add to dashboard | ⚙

Showing 13 of 13 records matched ⓘ

276 records (136.1 kB) scanned in 1.0s @ 285 records/s (140.9 kB/s)

Show histogram

#	@timestamp	@message	xray_trace_id
▶ 1	2024-10-17T17:45:17.7…	{"_aws":{"Timestamp":1729187117798,"CloudWatchMetrics":[{"Namespace"…	1-67114d2d-27d9ec0c7434c1bb6c56ae57
▶ 2	2024-10-17T17:44:47.7…	{"_aws":{"Timestamp":1729187087760,"CloudWatchMetrics":[{"Namespace"…	1-67114d0f-92f47107a5b092255b3744d4
▶ 3	2024-10-17T17:44:17.8…	{"_aws":{"Timestamp":1729187057827,"CloudWatchMetrics":[{"Namespace"…	1-67114cf1-8386eb3fccc2af911199a7fe
▶ 4	2024-10-17T17:43:47.7…	{"_aws":{"Timestamp":1729187027742,"CloudWatchMetrics":[{"Namespace"…	1-67114cd3-0f039bf7600344ddc402dae7

Query the logs

Logs (13)

Export results ▼ | Add to dashboard

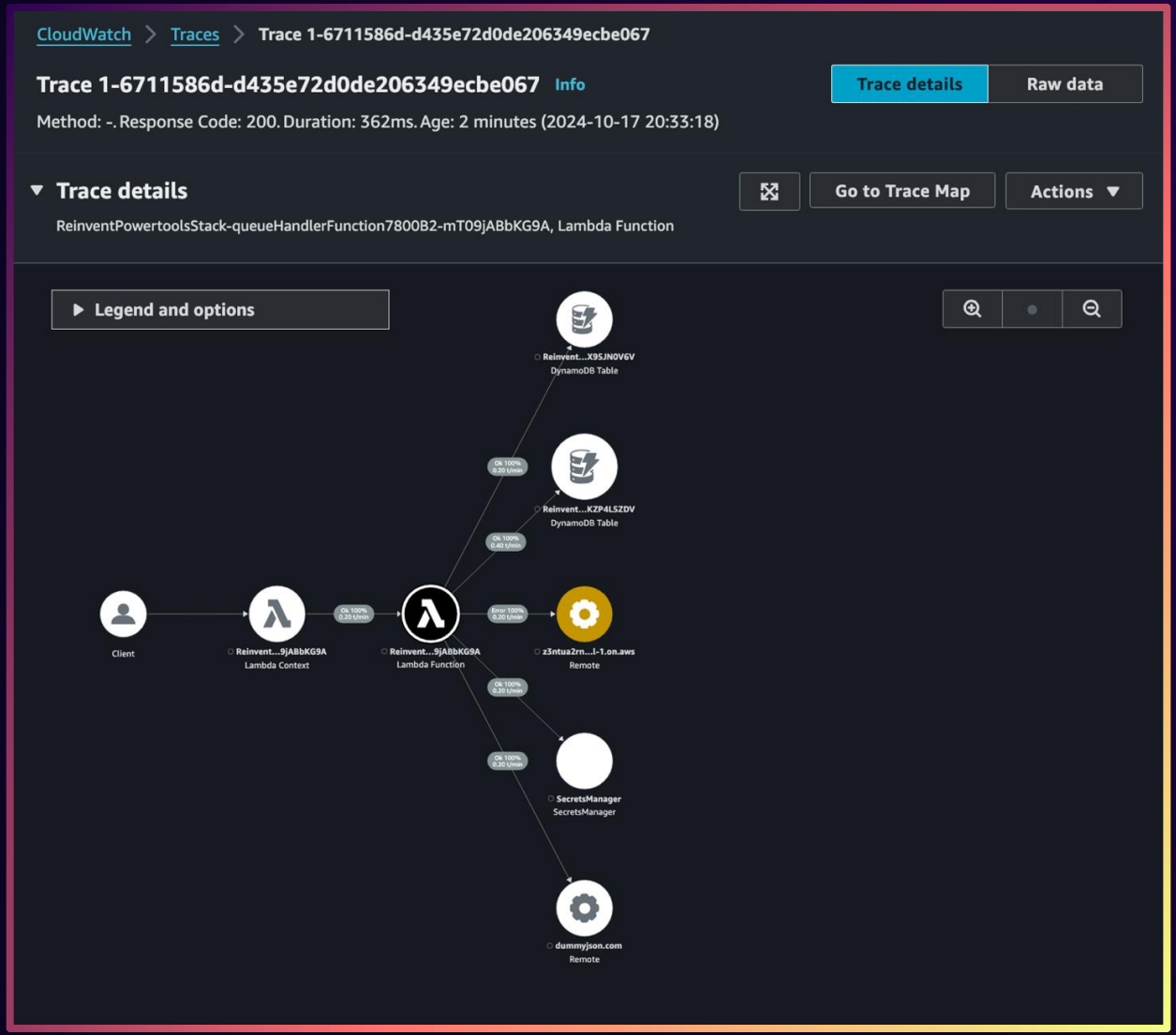
Showing 13 of 13 records matched

Show histogram

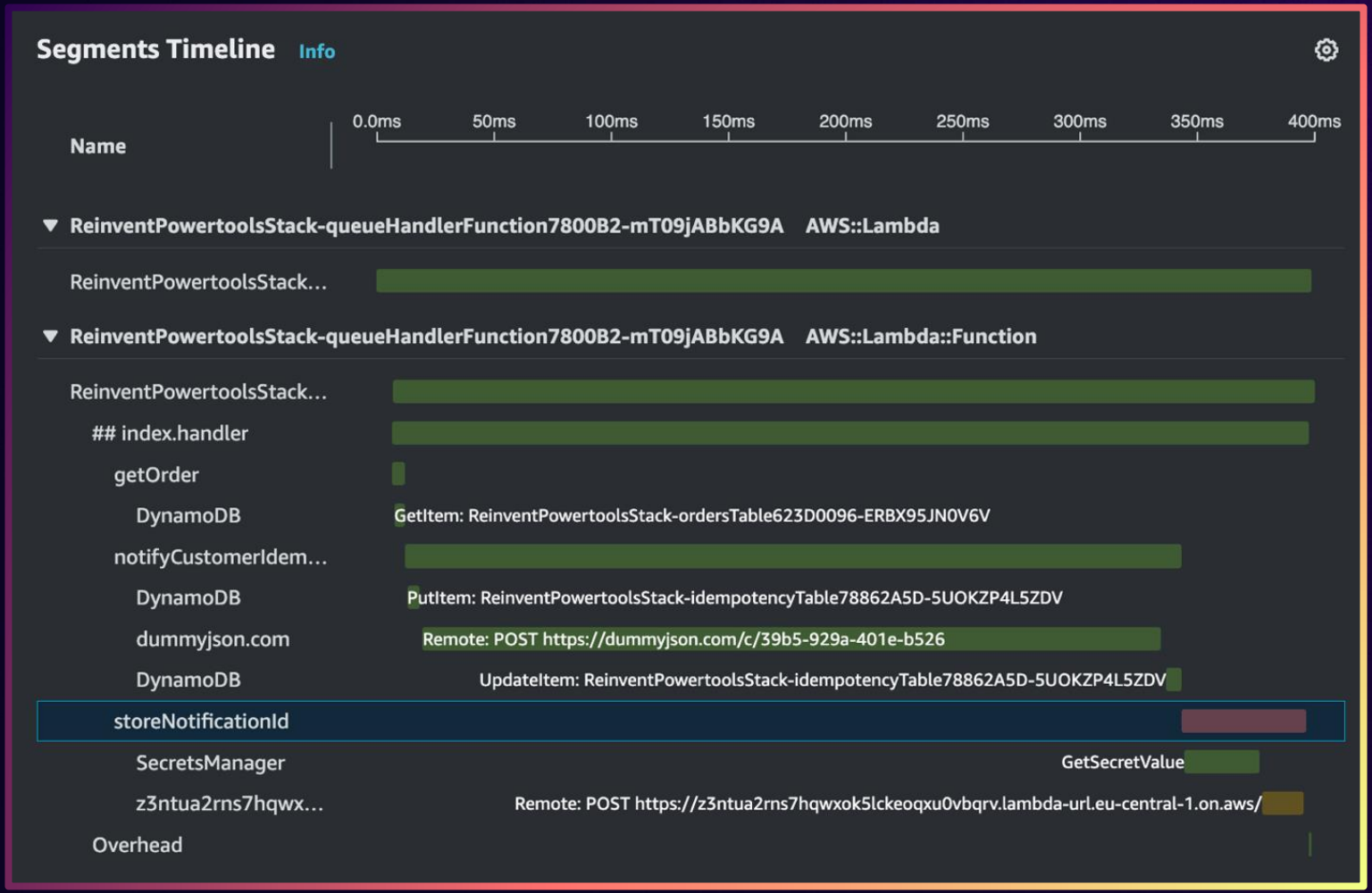
276 records (136.1 kB) scanned in 1.0s @ 285 records/s (140.9 kB/s)

#	@timestamp	@message	xray_trace_id
▶ 1	2024-10-17T17:45:17.7…	{"_aws":{"Timestamp":1729187117798,"CloudWatchMetrics":[{"Namespace…	1-67114d2d-27d9ec0c7434c1bb6c56ae57
▶ 2	2024-10-17T17:44:47.7…	{"_aws":{"Timestamp":1729187087760,"CloudWatchMetrics":[{"Namespace…	1-67114d0f-92f47107a5b092255b3744d4
▶ 3	2024-10-17T17:44:17.8…	{"_aws":{"Timestamp":1729187057827,"CloudWatchMetrics":[{"Namespace…	1-67114cf1-8386eb3fccc2af911199a7fe
▶ 4	2024-10-17T17:43:47.7…	{"_aws":{"Timestamp":1729187027742,"CloudWatchMetrics":[{"Namespace…	1-67114cd3-0f039bf7600344ddc402dae7

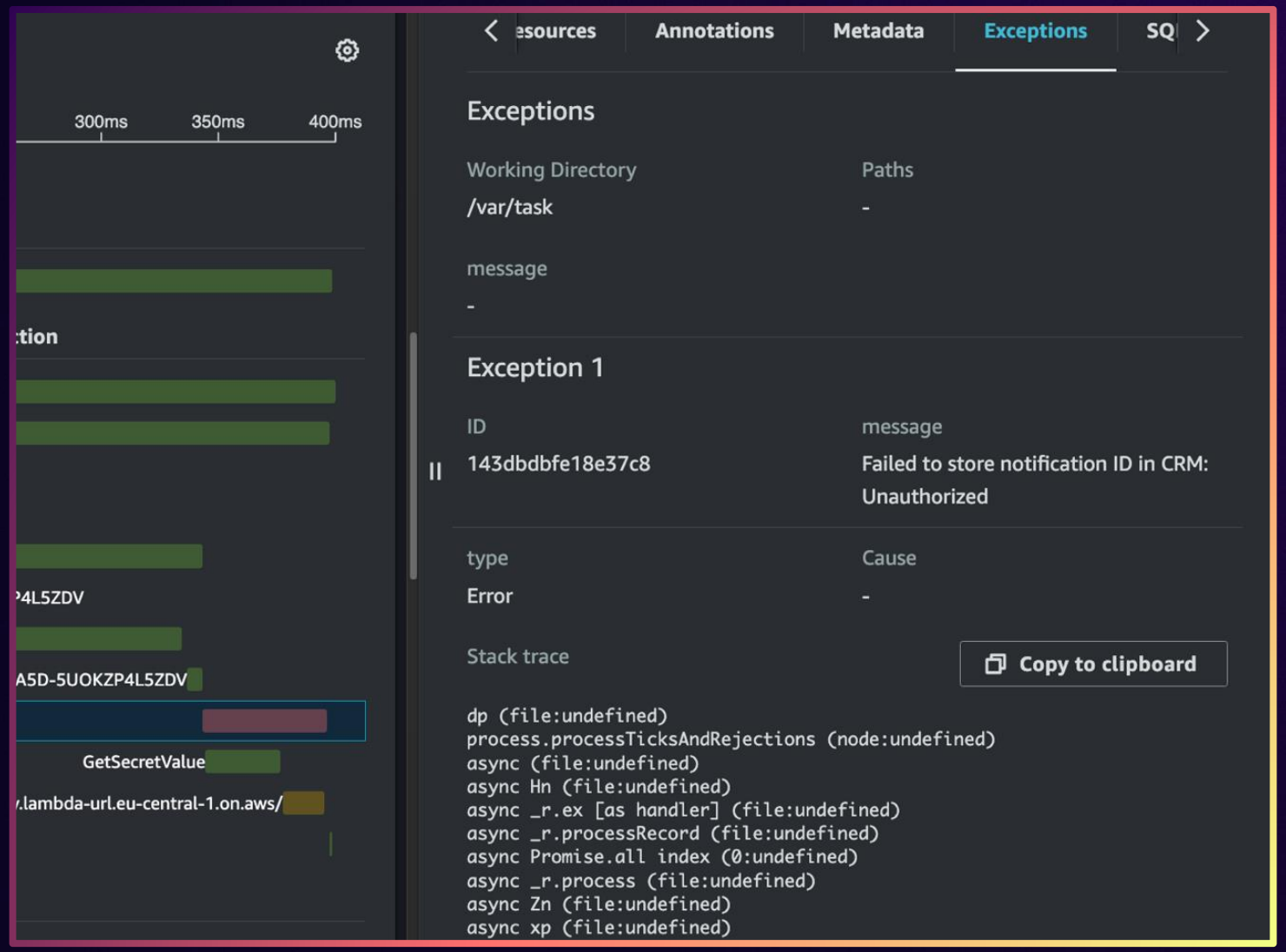
View traces



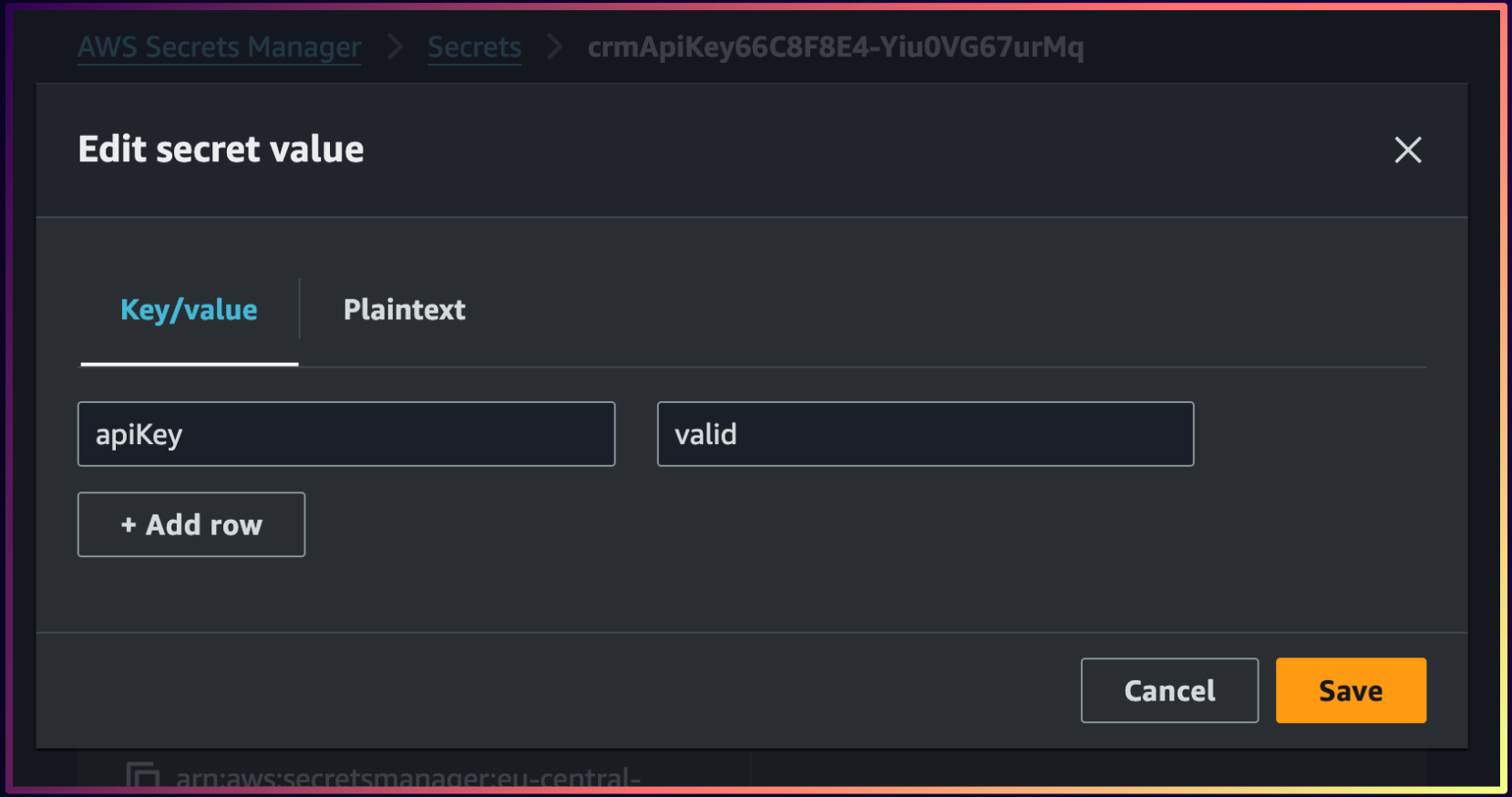
View traces



View traces



Fix the problem



Validate recovery

```
1  fields @timestamp, @message , xray_trace_id, orderId, message
2  | filter orderId = "order-id-1"
```

Query generator

Run query | Cancel | Save | History

Validate recovery

Logs (334) | Patterns (5) | Visualization

Logs (334)

Showing 334 of 334 records matched

774 records (439.0 kB) scanned in 0.8s @ 933 records/s (529.5 kB/s)

	xray_trace_id	orderId	message
bda:eu-central-1:2960…	1-67115d1e-16bbb1fa5286f0f34f90e6	order-id-1	Notification ID stored
bda:eu-central-1:2960…	1-67115d1e-16bbb1fa5286f0f34f90e6	order-id-1	Order found
bda:eu-central-1:2960…	1-67115d00-8d7c85f948122ab35c256c	order-id-1	Failed to process order
bda:eu-central-1:2960…	1-67115d00-8d7c85f948122ab35c256c	order-id-1	Failed to store notification ID
bda:eu-central-1:2960…	1-67115d00-8d7c85f948122ab35c256c		
bda:eu-central-1:2960…	1-67115ce2-58ec0b6efb29df0f41acb6…	order-id-1	Failed to store notification ID
bda:eu-central-1:2960	1-67115ce2-58ec0b6efb29df0f41acb6	order-id-1	Failed to process order

Validate recovery

Segments Timeline Info

Name	Segment status	Response code	Duration	Hosted in
▼ ReinventPowertoolsStack-queueHandlerFunction7800B2-mT09jABbKG9A			AWS::Lambda	
ReinventPowertoolsStack...	OK	200	83ms	
▼ ReinventPowertoolsStack-queueHandlerFunction7800B2-mT09jABbKG9A			AWS::Lambda::Function	
ReinventPowertoolsStack...	OK	-	77ms	
## index.handler	OK	-	74ms	
getOrder	OK	-	5ms	
DynamoDB	OK	200	4ms	GetItem: ReinventPowertoolsStack-ordersTable623D0096-ERBX95JN0V6V
notifyCustomerIdem...	OK	-	8ms	
storeNotificationId	OK	-	61ms	
SecretsManager	OK	200	30ms	GetSecretValue
z3ntua2rns7hqwx...	OK	200	28ms	Remote: POST https://z3ntua2rns7hqwxok5lckeoqxu0vbqrv.lambda-...
▼ DynamoDB			AWS::DynamoDB::Table	
DynamoDB	OK	200	4ms	GetItem: ReinventPowertoolsStack-ordersTable623D0096-ERBX95JN0V6V
DynamoDB	Error (4xx)	400	7ms	PutItem: ReinventPowertoolsStack-idempotencyTable78862A5D-5UOKZP4L5ZDV

0.0ms 10ms 20ms 30ms 40ms 50ms 60ms 70ms 80ms 90ms

65ms

That’s me

Raphael Manke

Senior IT Consultant, Cloud
manke@codecentric.de
www.codecentric.de

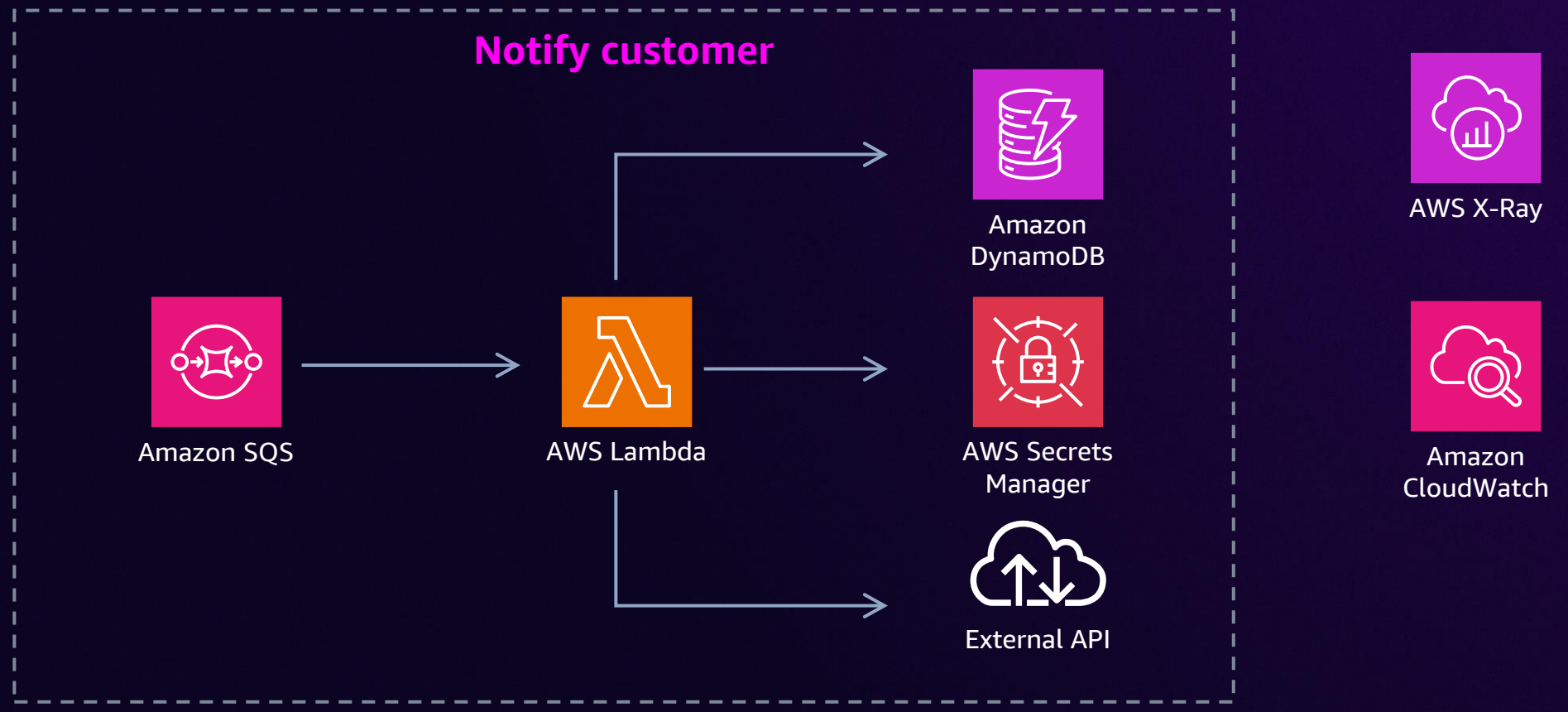
LinkedIn / X / GitHub
@RaphaelManke

Powertools for AWS Lambda

Unleash the power of
<serverless>
with Powertools for AWS

https://powertools.aws.dev

Architecture



Powertools for AWS Lambda – Metrics

Amazon CloudWatch

Instantiate metrics class

```
1  export const metrics = new Metrics({
2      namespace: 'customerNotifications',
3      serviceName: SERVICE_NAME,
4  });
```

Add metric data point

```
1  metrics.addMetric("failedNotification", MetricUnit.Count, 1);
```

Add a metadata attribute

```
1  metrics.addMetadata("xray_trace_id", tracer.getRootXrayTraceId()!);
```

Powertools for AWS Lambda – Metrics

Amazon CloudWatch

Example: Emitting metrics

```
22          }
23
24          logger.info("Notification ID stored", { notificationId });
25      } catch (error) {
26          logger.error("Failed to store notification ID", { error });
27          metrics.addMetric("failedNotification", MetricUnit.Count, 1);
28          throw error;
29      }
30  };
```

Powertools for AWS Lambda – Metrics

Amazon CloudWatch

Outputs metrics in embedded metrics format (EMF)

```
1  {
2      "_aws": {
3          "Timestamp": 1729191138141,
4          "CloudWatchMetrics": [
5              {
6                  "Namespace": "customerNotifications",
7                  "Dimensions": [
8                      [
9                          "service"
10                     ]
11                 ],
12                 "Metrics": [
13                     {
14                         "Name": "failed-storing-notification-id-crm",
15                         "Unit": "Count"
16                     }
17                 ]
18             }
19         ]
20     },
21     "service": "notification-service",
22     "failed-storing-notification-id-crm": 1,
23     "xray_trace_id": "1-67115ce2-58ec0b6efb29df0f41acb697"
24 }
```

Powertools for AWS Lambda – Tracer

AWS X-Ray

Instantiate tracer class and capture HTTPS calls

```
1  export const tracer = new Tracer({
2      serviceName: SERVICE_NAME,
3  });
```

Generate segments for AWS SDK calls

```
1  const dynamodbClient = tracer.captureAWSv3Client(new DynamoDB({}));
```

Add exceptions to trace segments

```
1  tracer.addErrorAsMetadata(err as Error);
```

Add annotations to traces

```
1  tracer.putAnnotation("error", true);
```

Add metadata to segment

```
1  tracer.putMetadata("getCommand", getCommand);
```

Powertools for AWS Lambda – Tracer

AWS X-Ray

Helper function for generating custom segments

```
1  export async function withSegment<T>(segmentName: string, fn: () => Promise<T>): Promise<T> {
2      const parentSubsegment = tracer.getSegment(); // This is the subsegment currently active
3      let subsegment: any | undefined;
4      if (parentSubsegment) {
5          // Create subsegment for the function & set it as active
6          subsegment = parentSubsegment.addNewSubsegment(segmentName);
7          tracer.setSegment(subsegment);
8      }
9      let res: T;
10     try {
11
12         res = await fn();
13         return res
14     } catch (err) {
15         // Add the error as metadata
16         tracer.addErrorAsMetadata(err as Error);
17         throw err;
18     } finally {
19         if (parentSubsegment && subsegment) {
20             // Close subsegment (the AWS Lambda one is closed automatically)
21             subsegment.close();
22             // Set the facade segment as active again
23             tracer.setSegment(parentSubsegment);
24         }
25     }
26 }
```

Powertools for AWS Lambda – Tracer

AWS X-Ray

```
14        const { notificationId } = await withSegment("notifyCustomerIdempotent", async () => {
15            return await notifyCustomerIdempotent({
16                email,
17                message,
18                id,
19            });
20        });
```

Trace view of an instrumented Lambda function

Segments Timeline Info

Name	Segment status	Duration	0.0ms 20ms 40ms 60ms 80ms
ReinventPowertoolsStack...	OK	79ms	
## index.handler	OK	76ms	
getOrder	OK	6ms	
DynamoDB	OK	5ms	GetItem: ReinventPowertoolsStack-ordersTable623D0096-E...
notifyCustomerIdem...	OK	8ms	
storeNotificationId	OK	61ms	
z3ntua2rns7hqwx...	OK	60ms	Remote: POST https://z3ntua2rns7hqwxok5lcke...
Overhead	OK	1ms	

Powertools for AWS Lambda – Tracer

AWS X-Ray

Trace view with a captured exception

Segment details: storeNotificationId

Overview | Resources | Annotations | Metadata | Exceptions | SQL

Exceptions

Working Directory	Paths	message
/var/task	-	-

Exception 1

ID	message	type	Cause
0a068165f121f604	Failed to store notification ID in CRM: Unauthorized	Error	-

Stack trace

Copy to clipboard

```
Hh (file:undefined)
process.processTicksAndRejections (node:undefined)
async (file:undefined)
async Wn (file:undefined)
async zn.QT [as handler] (file:undefined)
async zn.processRecord (file:undefined)
async Promise.all index (0:undefined)
async zn.process (file:undefined)
async Yn (file:undefined)
async Tp (file:undefined)
```

Powertools for AWS Lambda – Parameters

AWS Systems Manager

AWS Secrets Manager

AWS AppConfig

Amazon DynamoDB

Instantiate parameter provider class

```
const secretsClient = tracer.captureAWSv3Client(new SecretsManagerClient({}))
export const secretsProvider = new SecretsProvider(
    { awsSdkV3Client: secretsClient }
);
```

Read and cache parameters

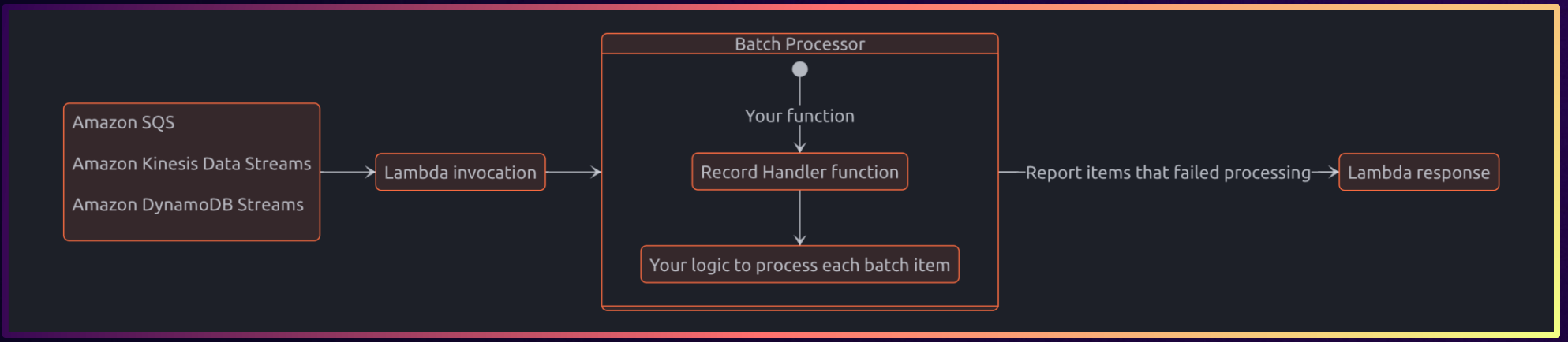
```
interface ApiKeySecret {
    apiKey: string;
}

const crmApiKey = await secretsProvider.get<ApiKeySecret>(process.env.CRM_API_KEY_SECRET_NAME!, {
    transform: "json",
    maxAge: 30,
});
```

Powertools for AWS Lambda – Batch processor

Amazon SQS

Amazon DynamoDB

Amazon Kinesis



Handle a single message in the batch

Orchestrate the processing

```
const recordHandler = async (event: SQSRecord): Promise<any> => {
    // process the record
};

const _handler: SQSHandler = async (event, context) => {
    return processPartialResponse(event, recordHandler, new BatchProcessor(EventType.SQS), {
        context,
        throwOnFullBatchFailure: false,
    })
};
```

Powertools for AWS Lambda – Idempotency

Amazon DynamoDB

Instantiate persistence layer to track progress

Register Lambda context to keep track of timeouts

Wrap function with idempotency helper

```
1  export const persistenceStore = new DynamoDBPersistenceLayer({
2      tableName: process.env.IDEMPOTENCY_TABLE_NAME!,
3      awsSdkV3Client: dynamodb
4
5  });
6  export const config = new IdempotencyConfig({});
7
8
9  const _handler: SQSHandler = async (event, context) => {
10     config.registerLambdaContext(context);
11
12     // ... some handler
13 };
```

```
1  const notifyCustomerIdempotent = makeIdempotent(notifyCustomer, {
2      persistenceStore,
3      config,
4  });
```

Powertools for AWS Lambda – Logger

Amazon CloudWatch

Instantiate logger class

```
export const logger = new Logger({
    serviceName: SERVICE_NAME,
});
```

Add persistent log attributes

```
logger.appendKeys({
    orderId: order.id,
});
```

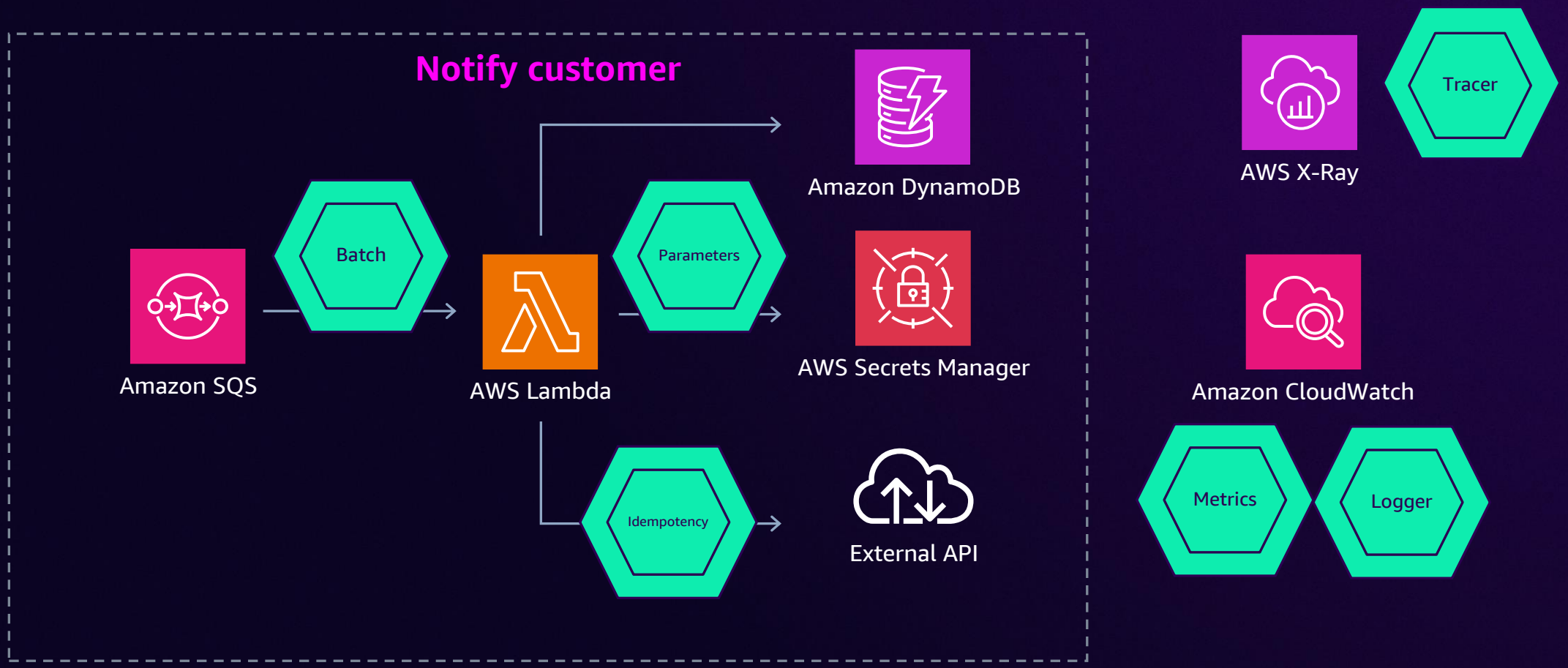
Drop in replacement additional log attributes

```
logger.info("Notification ID stored", { notificationId });

logger.error("Failed to process order", { error });
```

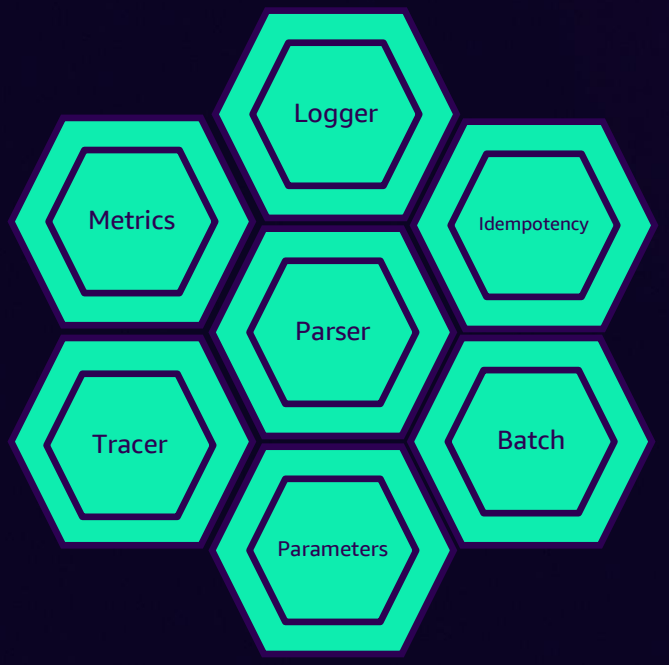
Inject standard attributes

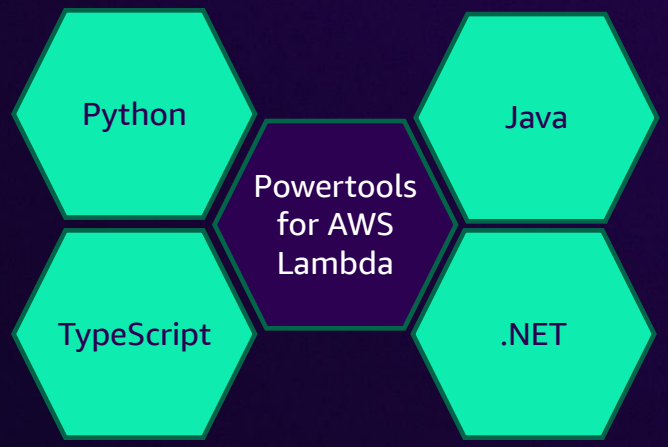
```
export const handler = middy(_handler)
    .use(injectLambdaContext(logger, {
        resetKeys: true,
    }))
```

Supercharged Lambda function with Powertools for AWS Lambda

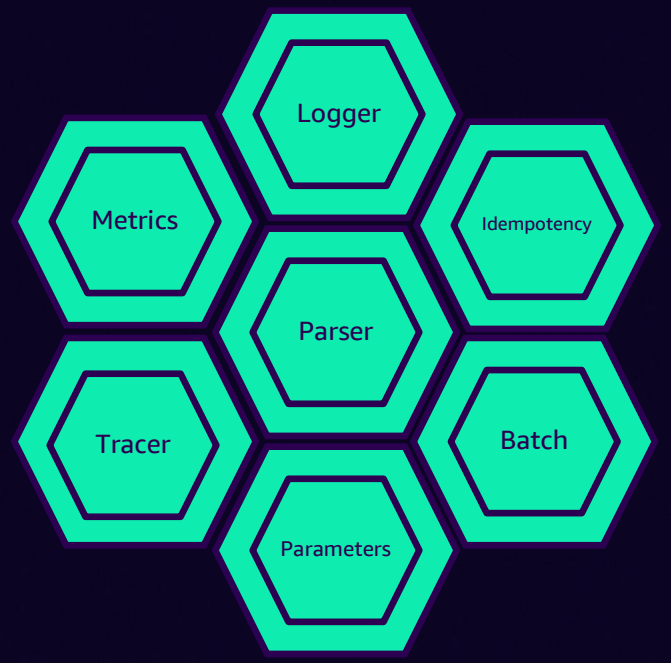


Powertools for AWS Lambda





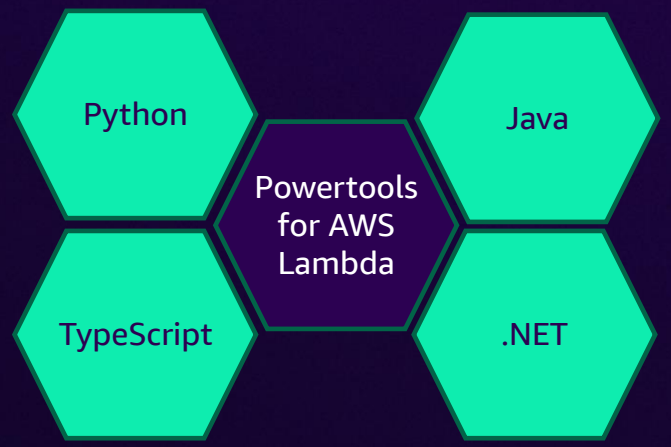
Powertools for AWS Lambda



More Powertools sessions

Code	Type	Time
SVS306	Workshop	Tuesday @ Mandalay Bay 11:30 AM – 1:30 PM
OPN402	Breakout	Tuesday @ Mandalay Bay 5:30 AM – 6:30 PM
OPN301-R1	Workshop	Wednesday @ MGM Grand 8:30 AM – 10:30 AM
SVS311	Code talk	Wednesday @ Wynn 1:00 PM – 2:00 PM

https://powertools.aws.dev



Thank you!

Please complete the session survey in the mobile app

Raphael Manke

Senior IT Consultant, Cloud
codecentric AG

LinkedIn / X / GitHub
@RaphaelManke