aws

SUMMIT
ONLINE

OPE07

Operations for serverless

Chandra S Allaka

Senior Consultant
Amazon Web Services

aws SUMMIT ONLINE

Agenda

Why is operations for serverless different?

Key challenges and solutions

- Dependency management
- Issue identification and resolution
- Change and release management

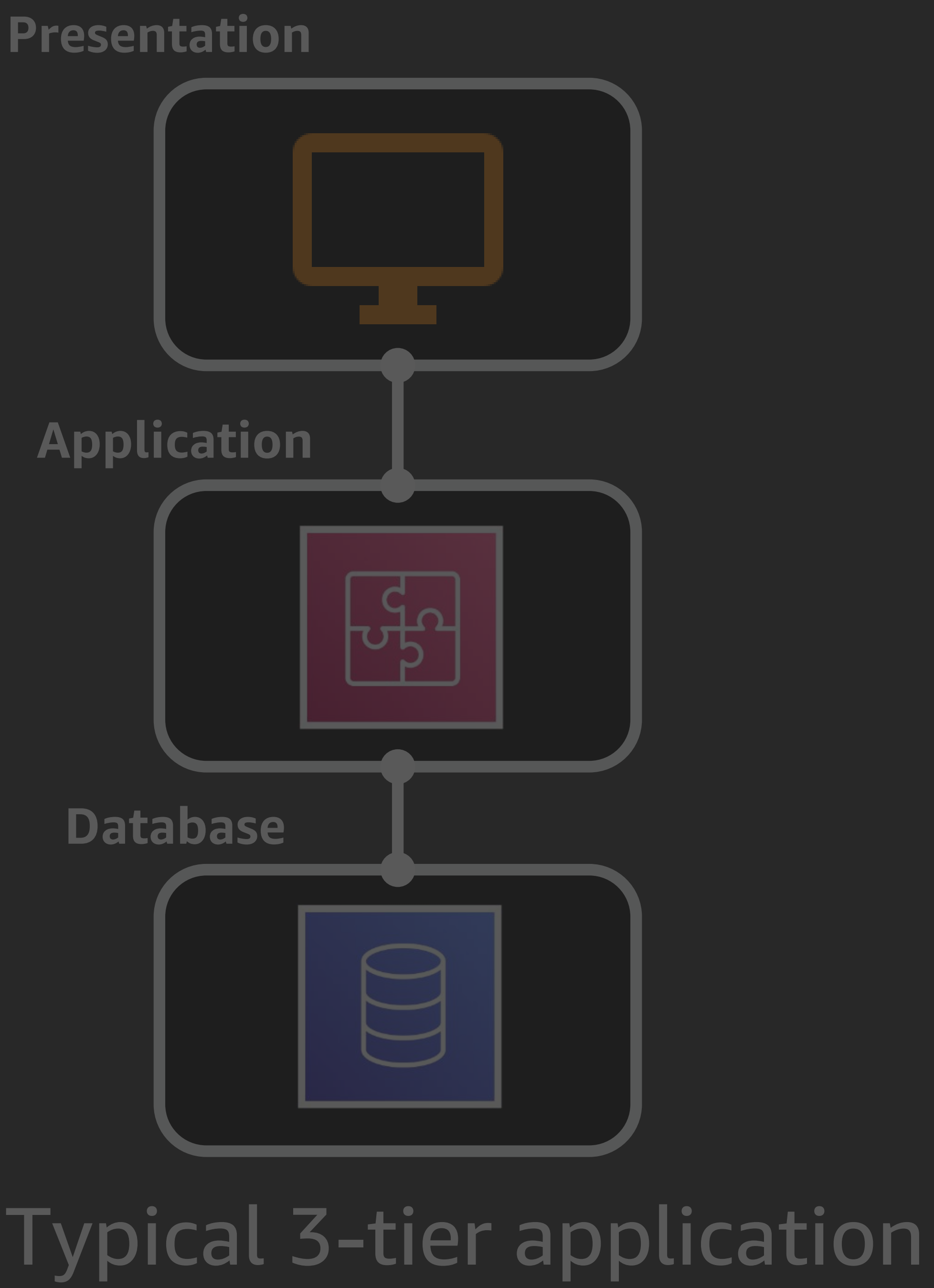
Serverless is the new normal

Increase business agility

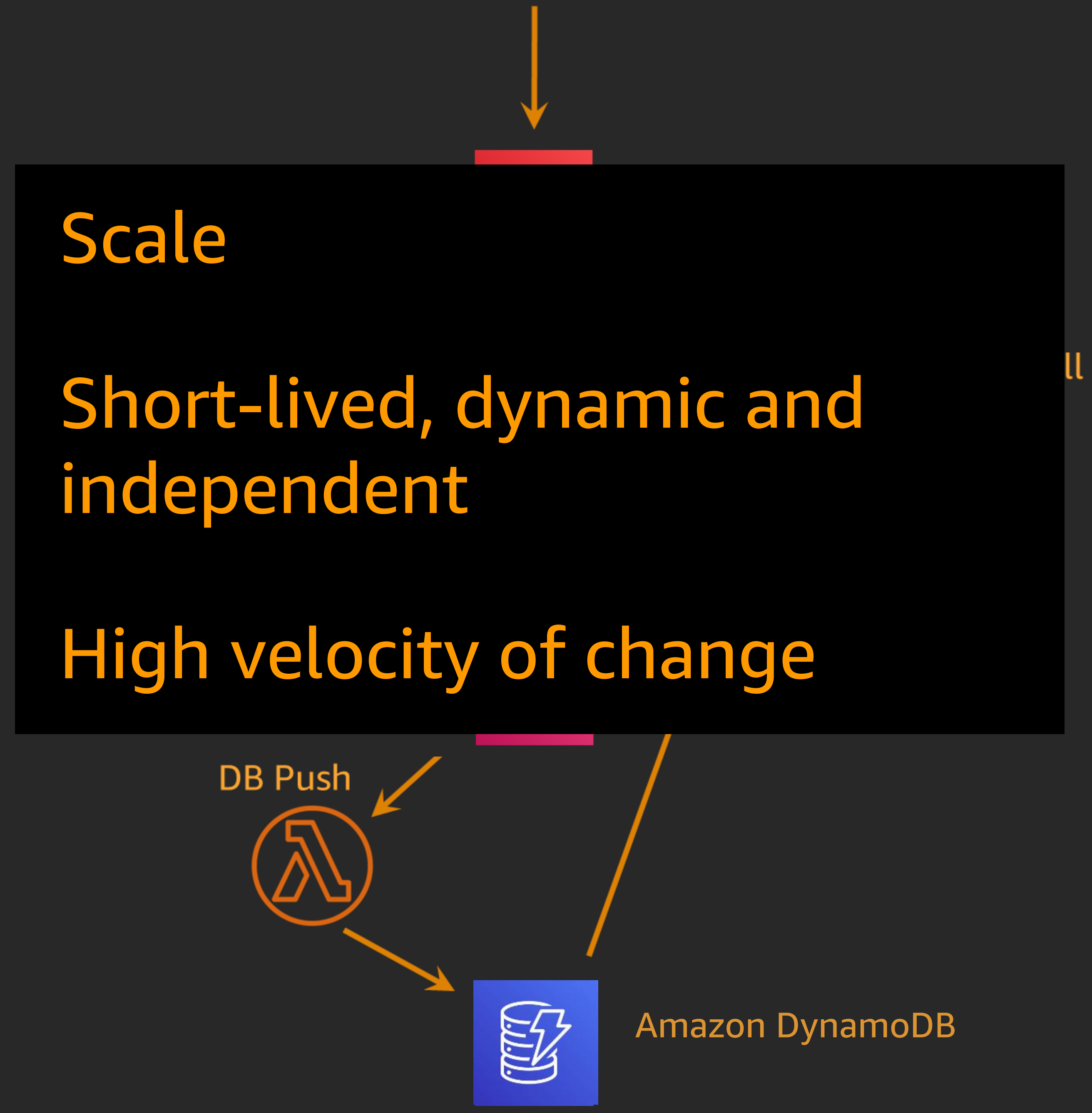
Reduce undifferentiated heavy lifting

Optimise costs by paying only for what you use

Why is operations for serverless different?



Typical 3-tier application



Scale

Short-lived, dynamic and independent

High velocity of change

Key operational challenges

OC 1

Dependency management

OC 2

Issue identification and resolution

OC 3

Change and release management

OC 1 - Dependency management

Key operational challenges

OC 1

Dependency management

OC 2

Issue identification and resolution

OC 3

Change and release management

Dependency management – why to manage dependencies?

Failure impact analysis

Faster issue resolution

Security impact

Change risk management

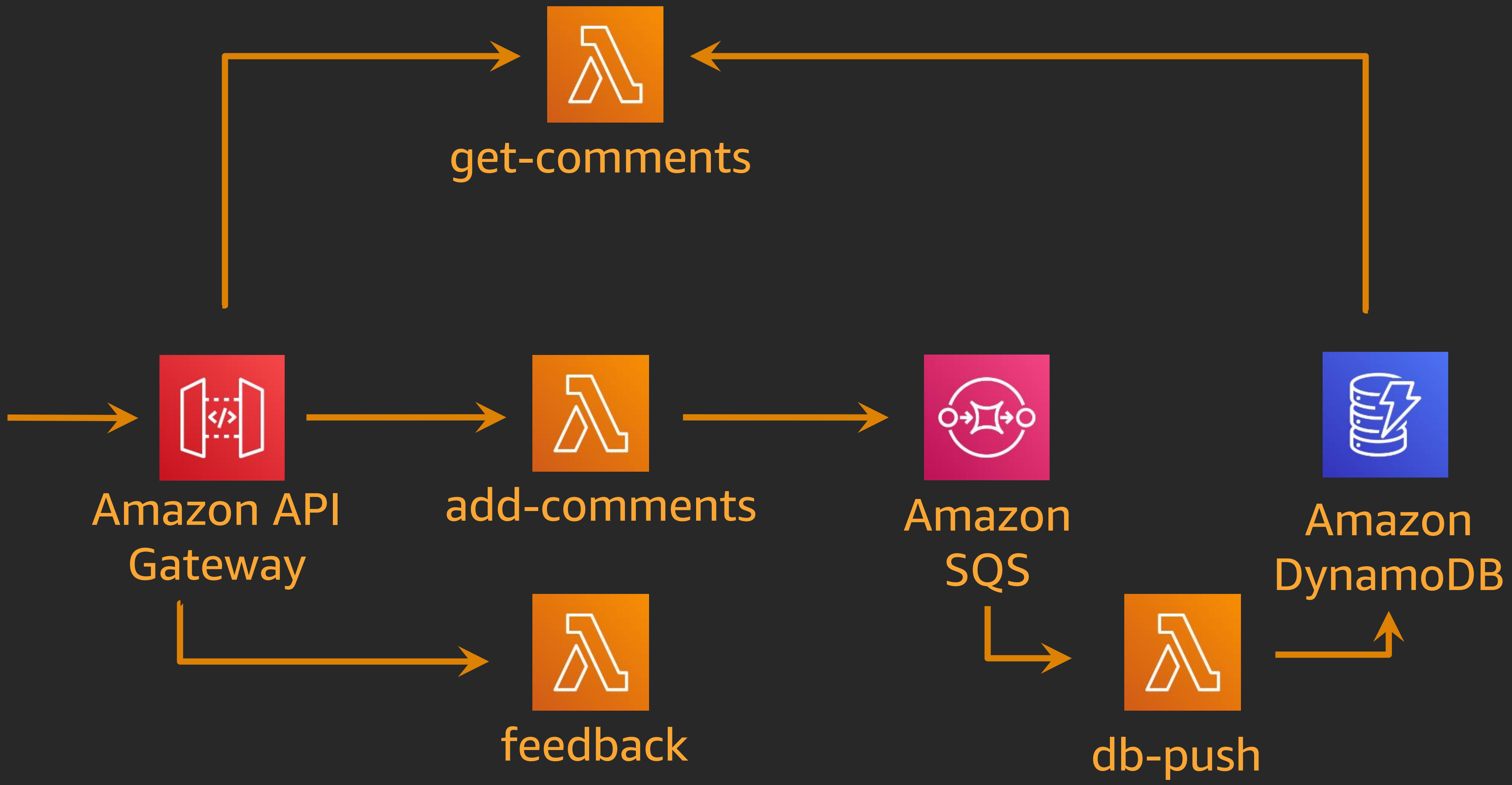
Let’s look at this sample application

A simple online feedback application

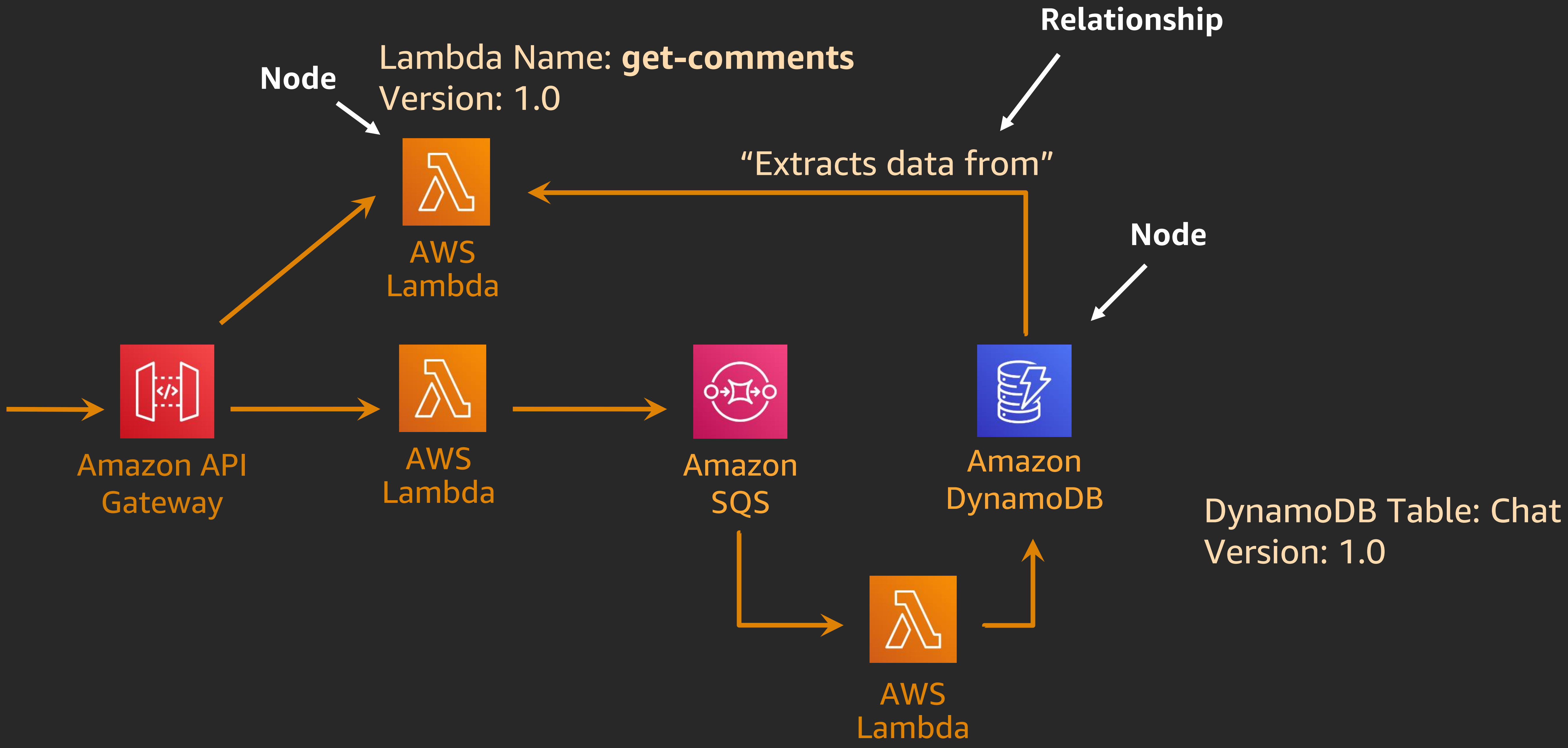
Demo

aws SUMMIT ONLINE

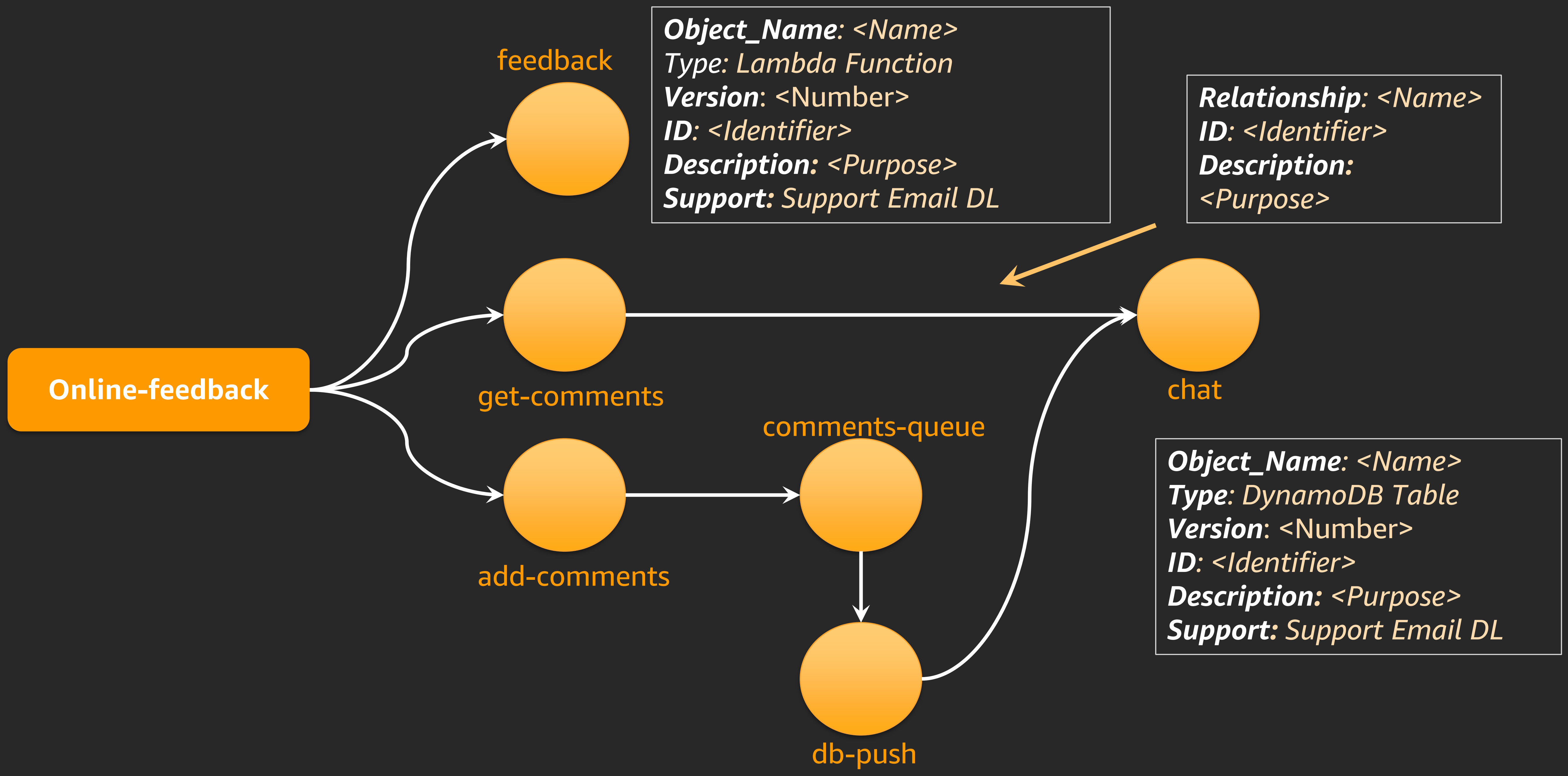
Application architecture



Identifying dependencies



Service/application map illustration

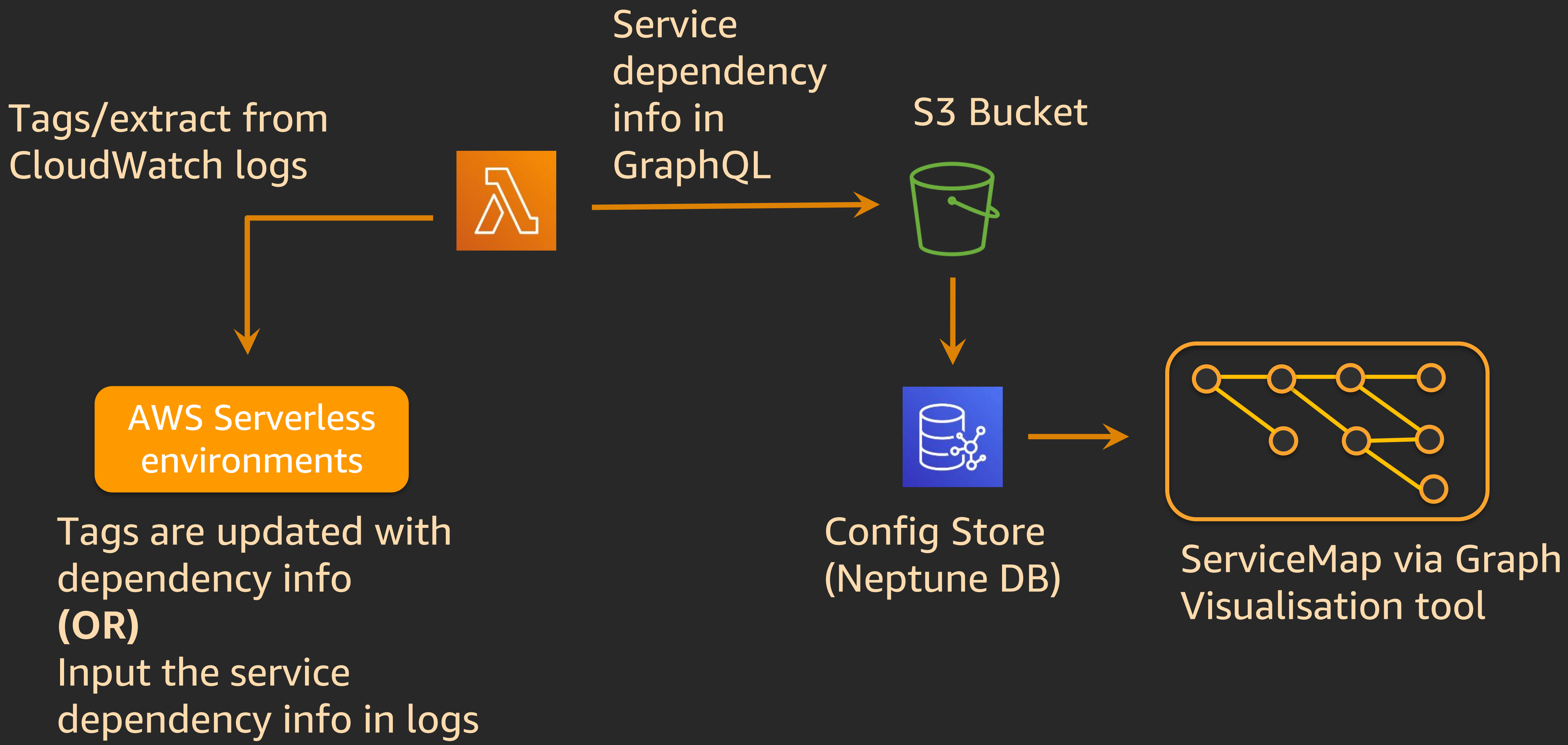


Object_Name: *<Name>*
Type: *Lambda Function*
Version: <Number>
ID: *<Identifier>*
Description: *<Purpose>*
Support: *Support Email DL*

Relationship: *<Name>*
ID: *<Identifier>*
Description:
<Purpose>

Object_Name: *<Name>*
Type: *DynamoDB Table*
Version: <Number>
ID: *<Identifier>*
Description: *<Purpose>*
Support: *Support Email DL*

Dependency management – solution overview


Tags/extract from CloudWatch logs
Service dependency info in GraphQL
S3 Bucket
AWS Serverless environments
Tags are updated with dependency info
(OR)
Input the service dependency info in logs
Config Store (Neptune DB)
ServiceMap via Graph Visualisation tool

Building dependency matrix

Sample tagging mechanism to identify dependencies

```
Upstream:   Fn:<function name-version, function name-version>;

Downstream: Fn:<function name-version>;SQS:<SQS Queue name>
```

Identifying the dependencies from the graph db

```
gremlin> g.V().has('name', 'add-
comments').out('depends').valueMap()


==>{name=[comments-queue]}
```

Dependency management – key take away

Dependency management is key to issue resolution and change control

Build mechanisms to identify function dependencies

OC 2 – Issue identification and resolution

Key operational challenges

OC 1

Dependency and change management

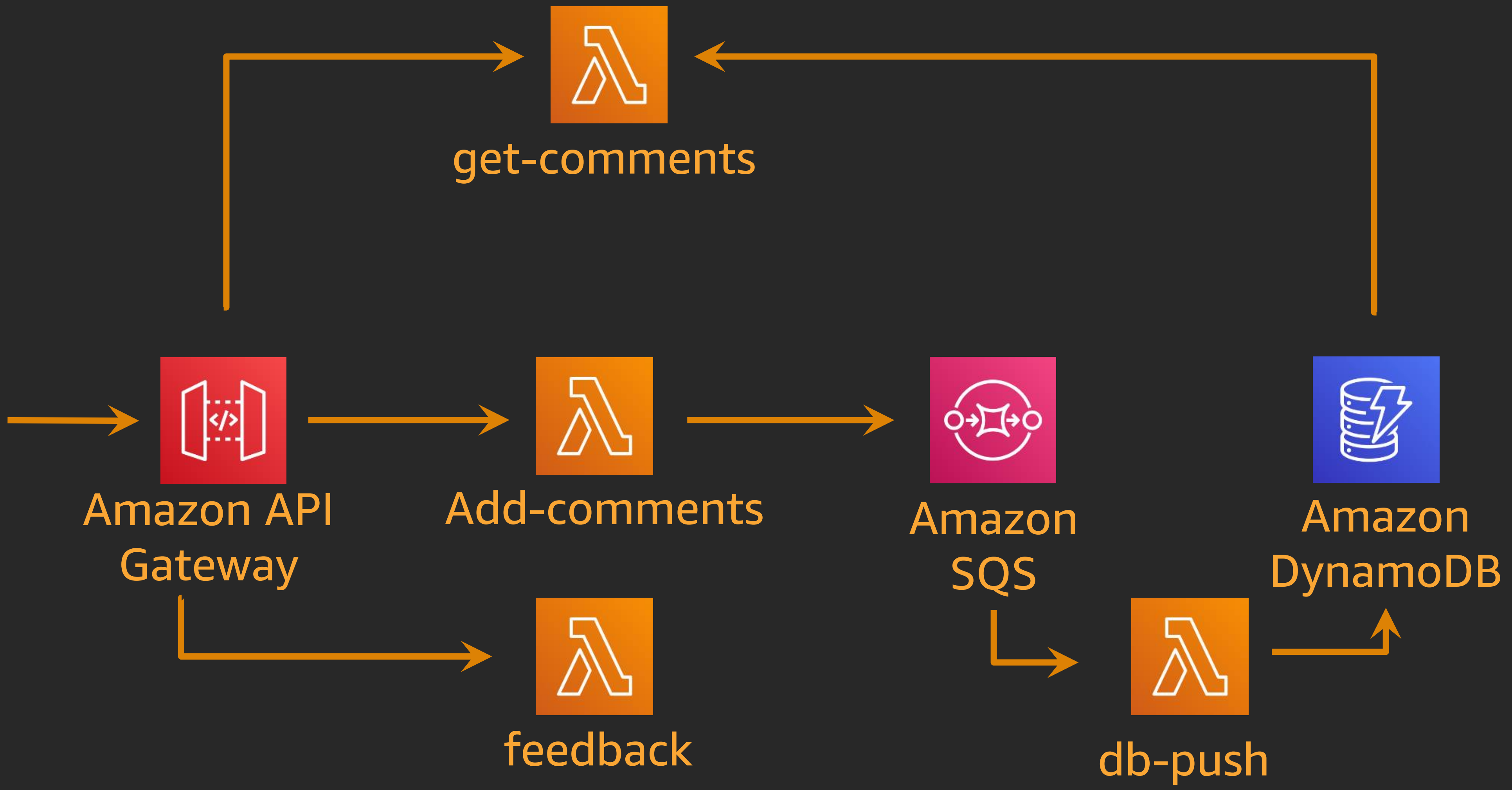
OC 2

Issue identification and resolution

OC 3

Change and release management

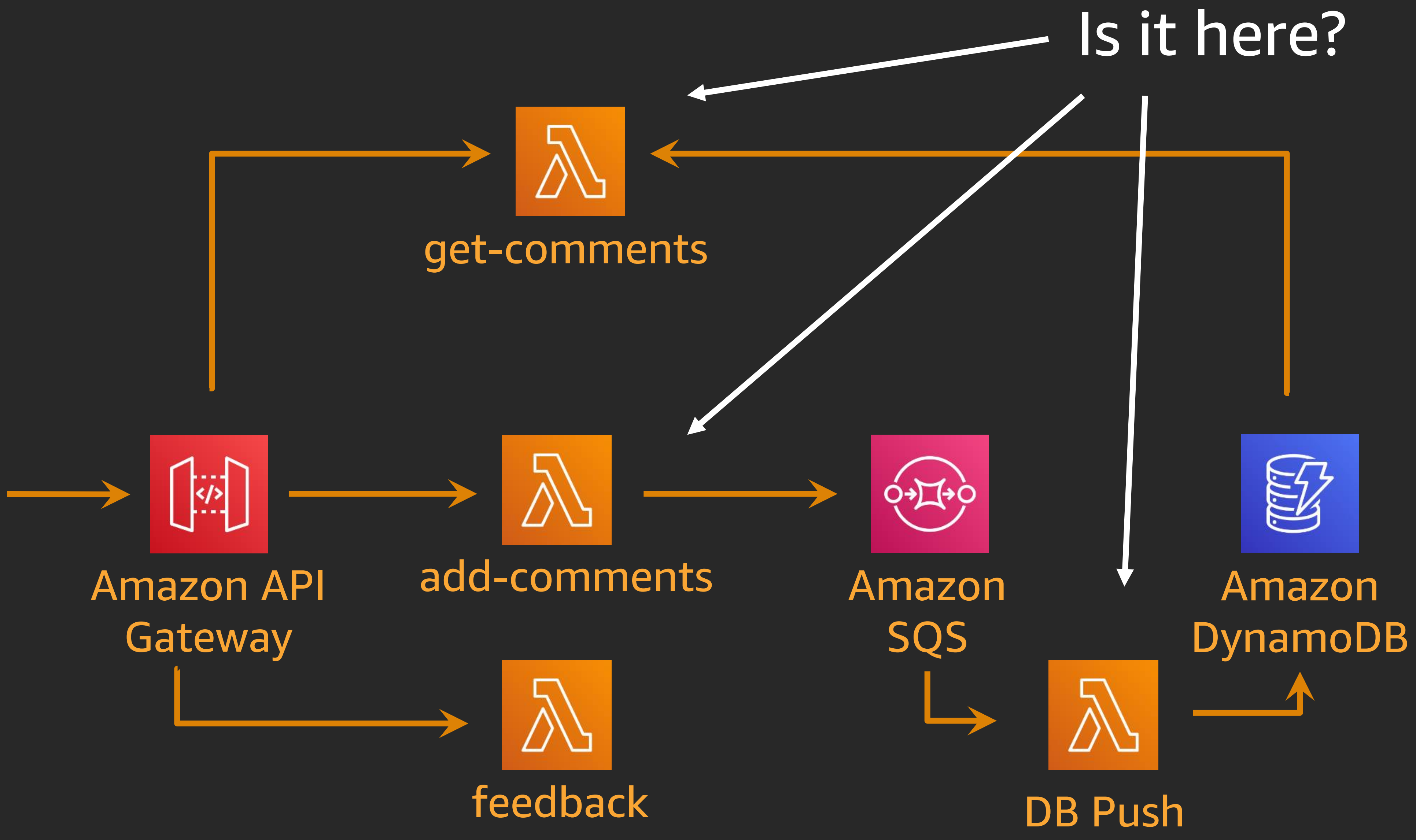
Application architecture of the demo app



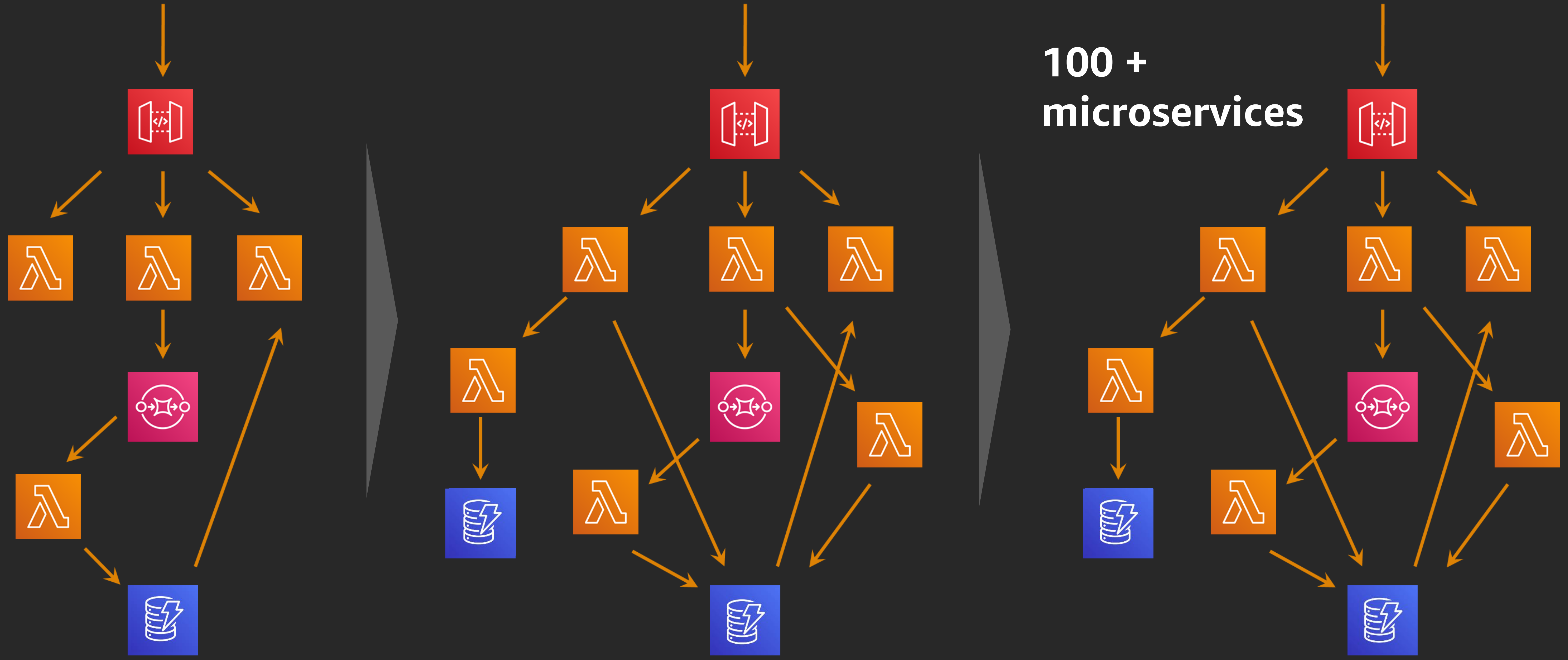
But applications break invariably

How do you identify where the application breaks?

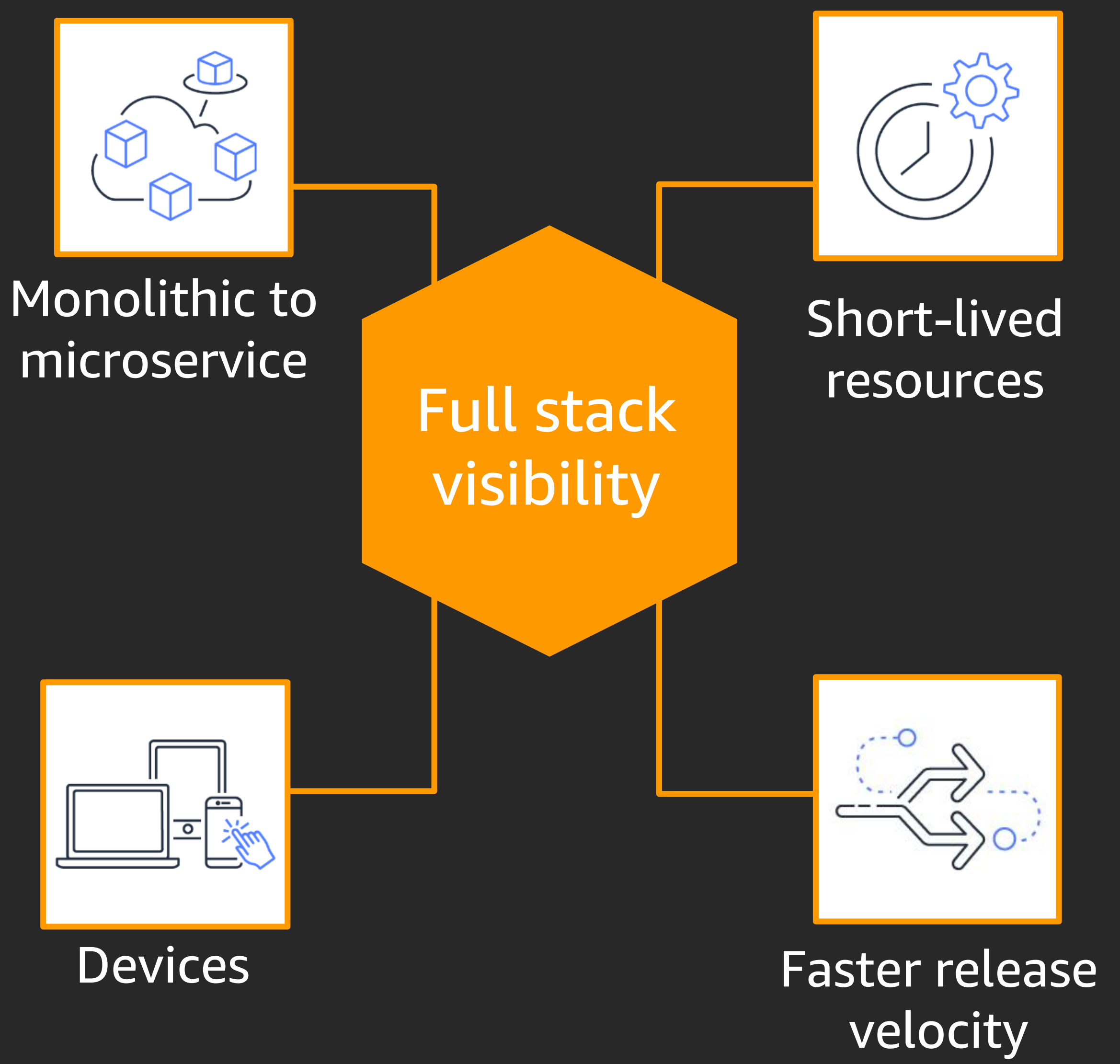
Application architecture of the demo app



And the challenges grow with scaling…



Traditional monitoring must evolve to manage these challenges



Introducing observability

“The system attribute that provides the measure of how well internal states of a system can be inferred from knowledge of its external outputs”

Wikipedia

Introducing observability

In-built

"The system attribute that provides the measure of how well internal states of a system can be inferred from knowledge of its external outputs"

Wikipedia

Visibility

Visibility requires metrics, logs and traces

CloudWatch Metrics

CloudWatch Logs

AWS X-Ray Traces

“The system attribute that provides the measure of how well internal states of a system can be inferred from knowledge of its external outputs”

Metrics

Pre-built metrics

Custom metrics and Log Filters

Logs

Structured logging

Correlation across the landscape

Log insights

Structured logging

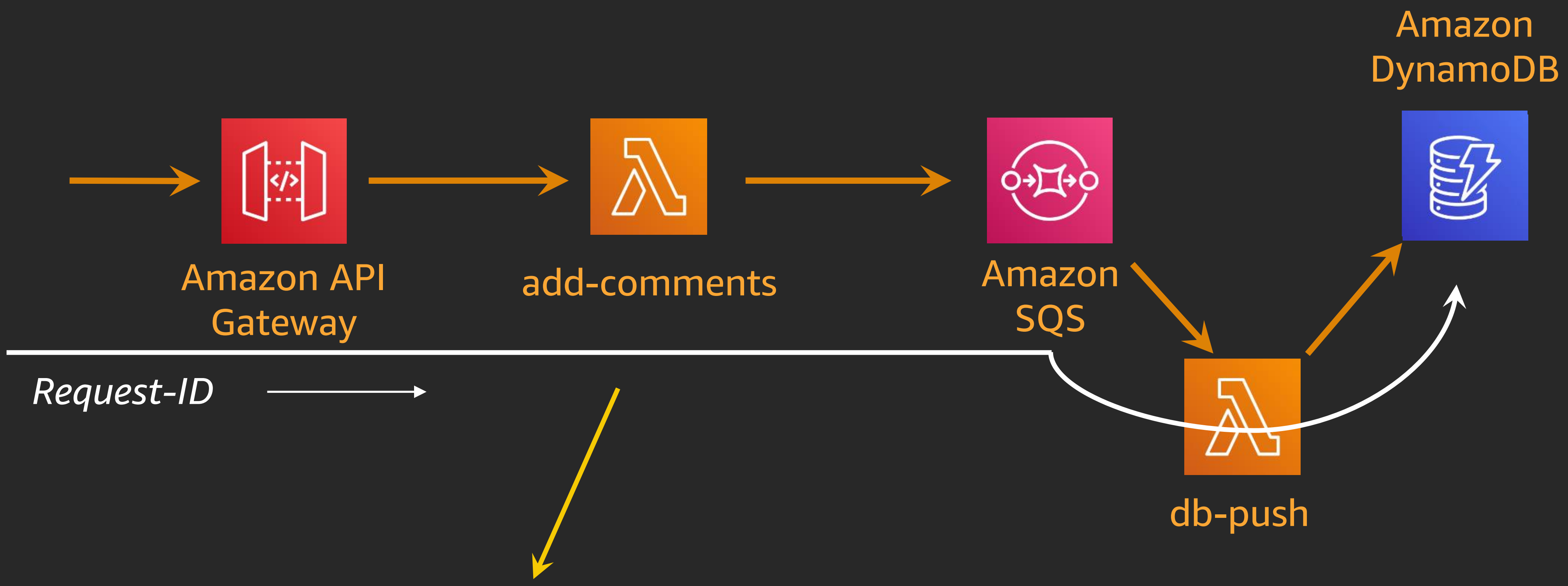
Standardise logging across the functions through a custom logger

Easy to query across the log files

Sample Structured Log

```
{
    "timestamp": "2019-11-26 18:17:33,774",
    "level": "INFO",
    "service": "booking",
    "lambda_function_arn": "arn:aws:lambda:xxx:acct:function:test",
    "correlation_id": "1234-xyzd-abcd",
    "lambda_request_id": "52fdfc07-2182-154f-163f5f0f9a621d72",
    “key_activity”: “Update DB”
    "message": {
        "operation": "update_item",
        "details:": {  ….    },
            "ResponseMetadata": {
                "RequestId": "GNVV4KQNSO5AEMVJF66Q9ASUAAJG",
                "HTTPStatusCode": 200,
                "HTTPHeaders": { …. },
            }
        }
```

Log correlation



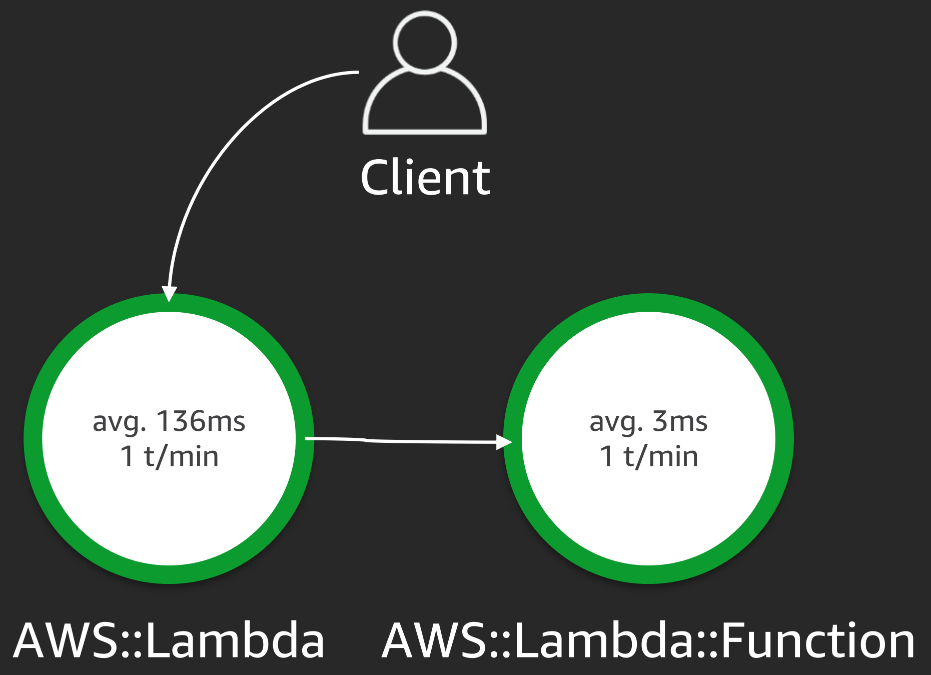
```
def index(event, context):
  logger.info("API Gateway Request ID : " +
  event['requestContext']['requestId'])
```

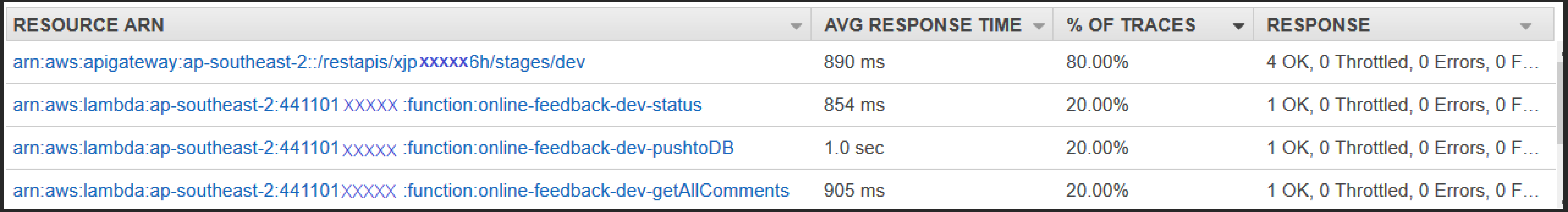
Tracing – AWS – X-Ray

Review request behavior

Discover application issues

Find bottlenecks to improve application performance





RESOURCE ARN	AVG RESPONSE TIME	% OF TRACES	RESPONSE
arn:aws:apigateway:ap-southeast-2::/restapis/xjpxxxxx6h/stages/dev	890 ms	80.00%	4 OK, 0 Throttled, 0 Errors, 0 F...
arn:aws:lambda:ap-southeast-2:441101 XXXXX :function:online-feedback-dev-status	854 ms	20.00%	1 OK, 0 Throttled, 0 Errors, 0 F...
arn:aws:lambda:ap-southeast-2:441101 XXXXX :function:online-feedback-dev-pushtoDB	1.0 sec	20.00%	1 OK, 0 Throttled, 0 Errors, 0 F...
arn:aws:lambda:ap-southeast-2:441101XXXXX :function:online-feedback-dev-getAllComments	905 ms	20.00%	1 OK, 0 Throttled, 0 Errors, 0 F...

Issue identification and resolution – key take away

Custom monitoring via CloudWatch Metric filters

Structured Logging

Log Correlation

Instrument for tracing

OC 3 – Change and release management

Key operational challenges

OC 1

Dependency and change management

OC 2

Issue identification and resolution

OC 3

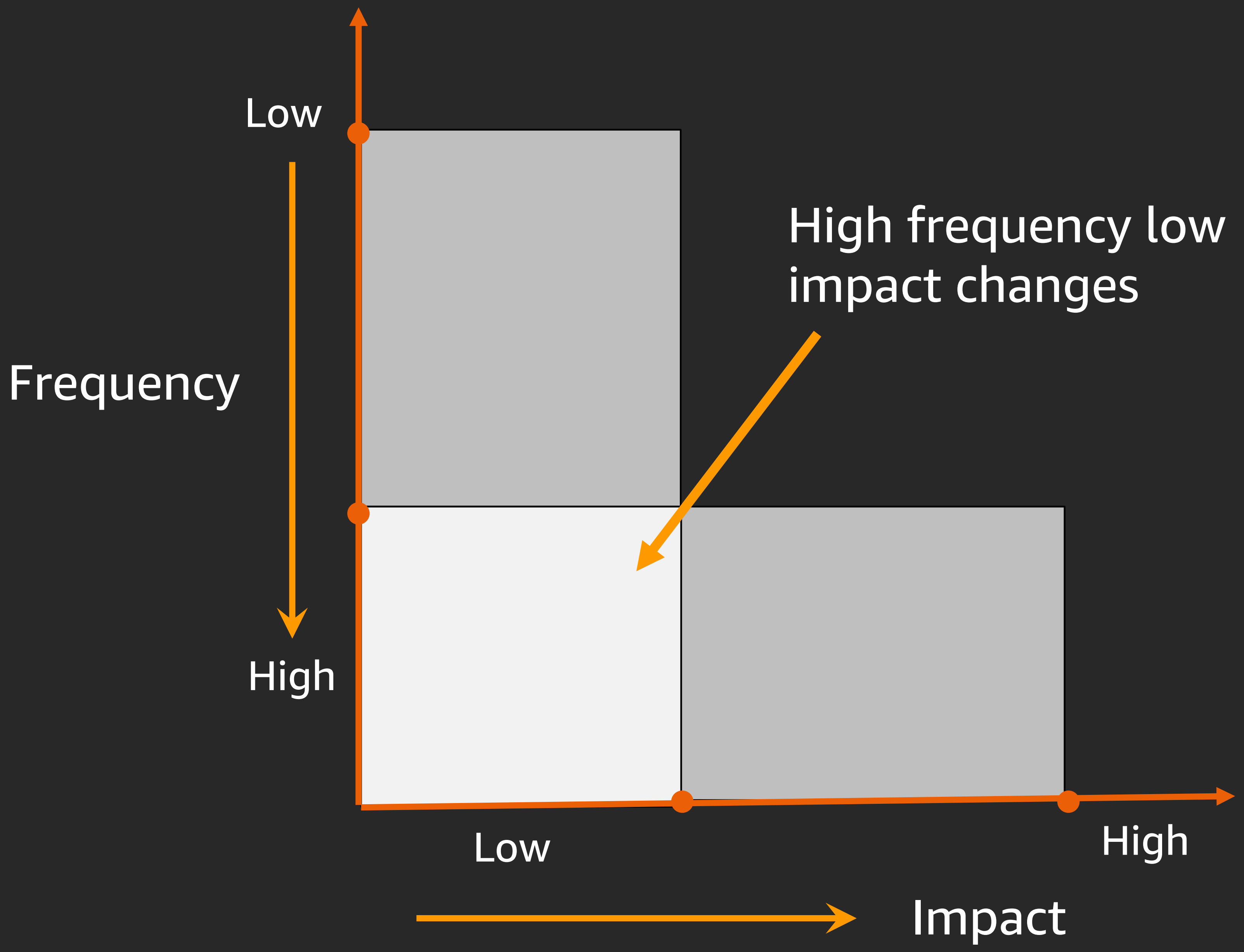
Change and release management

Change and release management

Traditional change management processes and mechanisms need to evolve to manage rapid changes in a serverless environment.

- High frequency of changes
- Multiple moving parts
- Lot more dependencies

Change classification and process transformation



Change and release process transformations

Adopt change 'pull' mechanisms

- Lambda versioning
- Support for n-3 versions

Release process transformation

- Blue-green & Canary deployments
- AWS API Gateway Canary release deployment

More small and frequent changes

Change and release management – key take away

Classify the changes

Small and frequent changes

Optimise existing processes – Reduce risks through versioning, canary deployment features

Summary and call to action

Realise that operations for serverless is different

Design and **build** with operations in mind

Thank you!

Chandra S Allaka

callaka@amazon.com