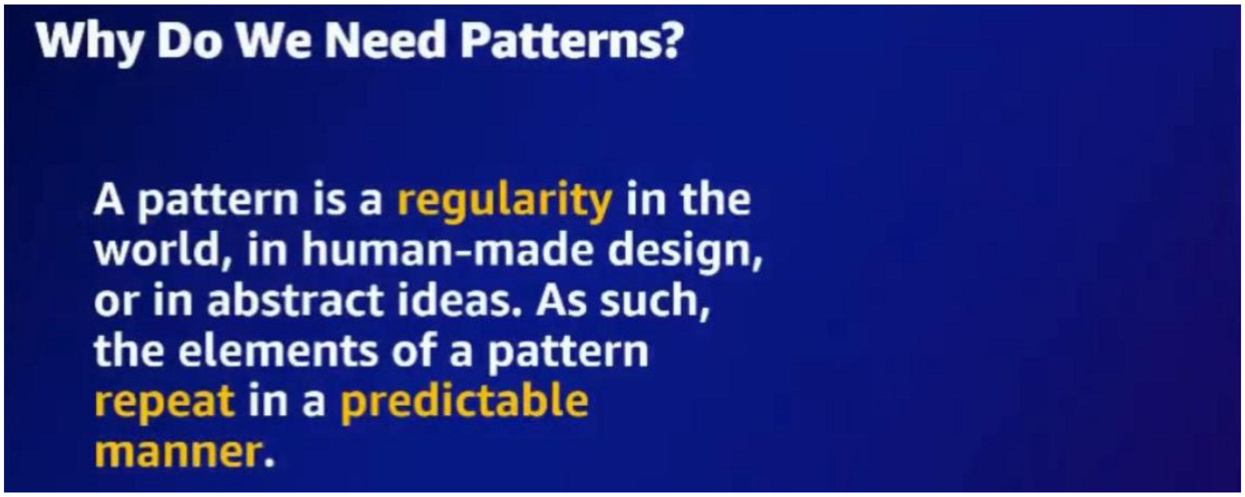
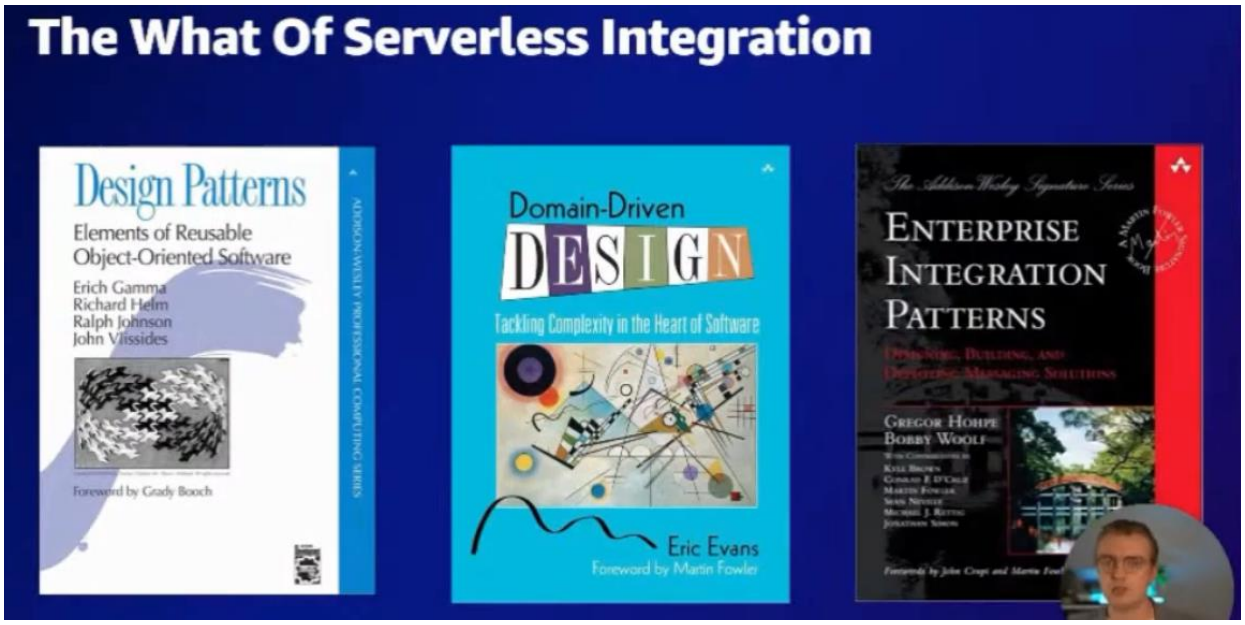


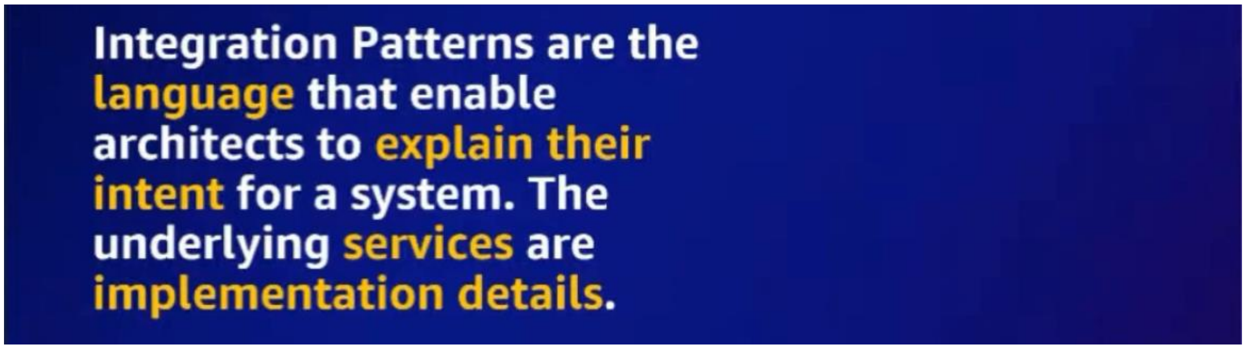
## Why Do We Need Patterns?

A pattern is a **regularity** in the world, in human-made design, or in abstract ideas. As such, the elements of a pattern **repeat** in a **predictable manner**.



## The What Of Serverless Integration

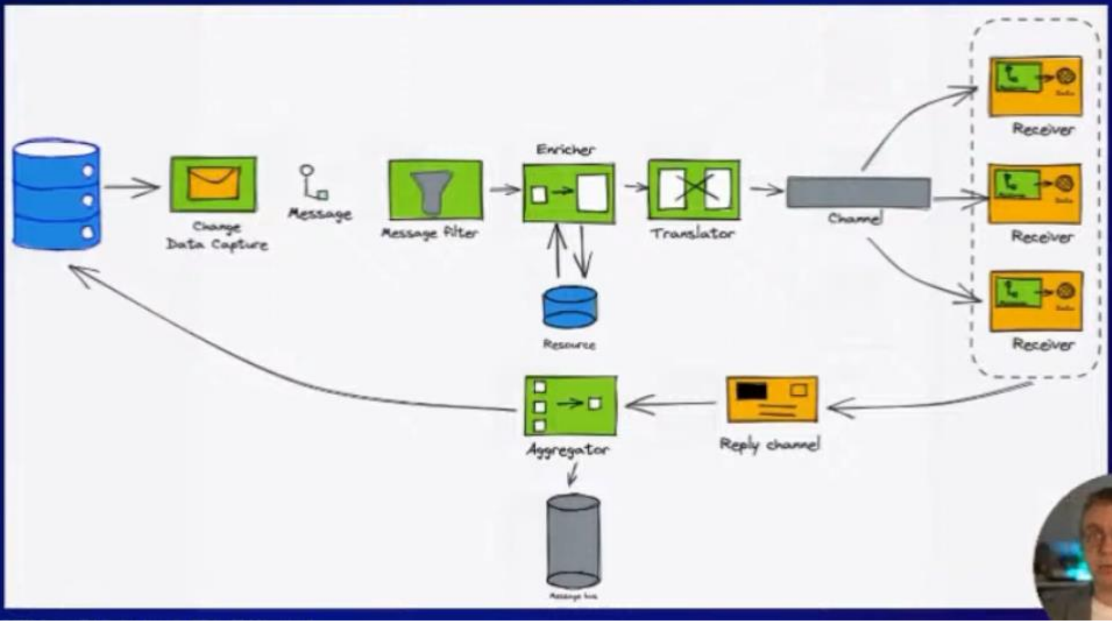
Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
Addison-Wesley Professional Computing Series
Domain-Driven DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler
The Addison-Wesley Signature Series
Enterprise Integration Patterns
Designing, Building, and Deploying Messaging Solutions
Gregor Hohpe
Bobby Woolf



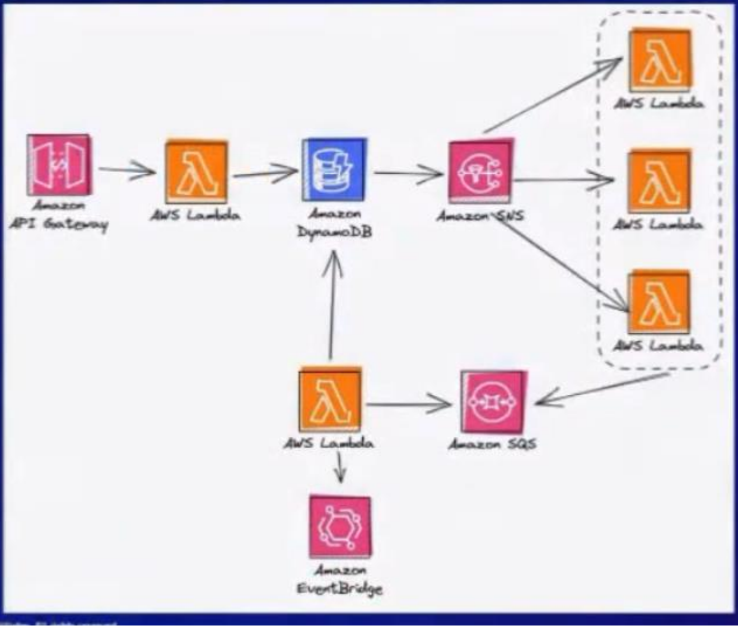
Integration Patterns are the **language** that enable architects to **explain their intent** for a system. The underlying **services** are **implementation details**.

## Patterns in Architecture

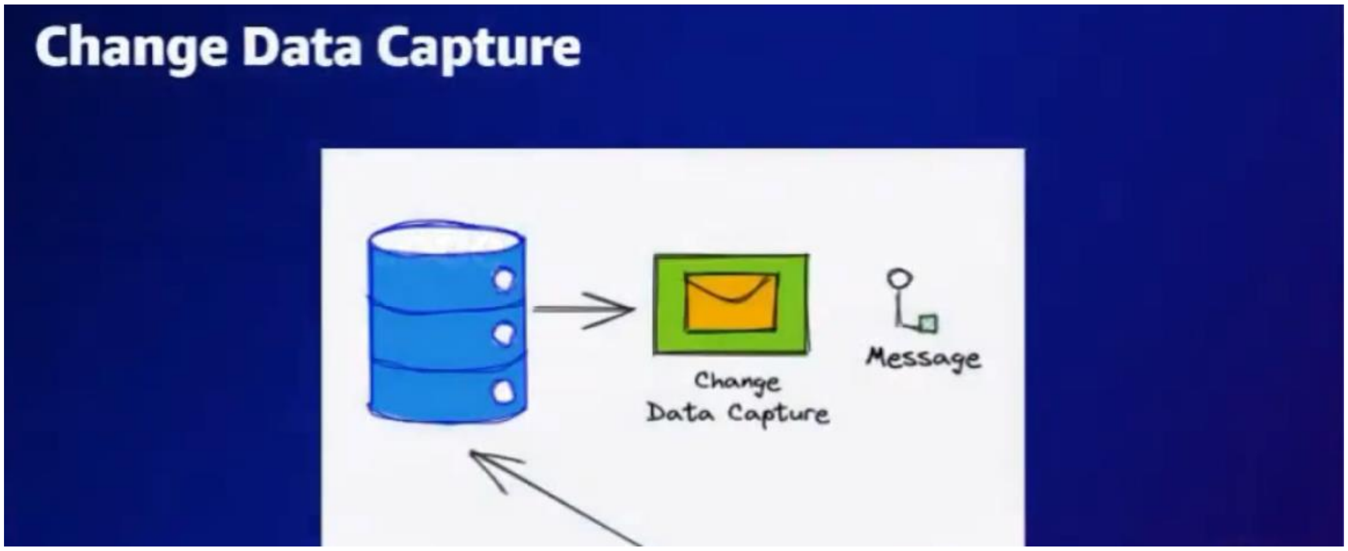
## The How Of Serverless Integration



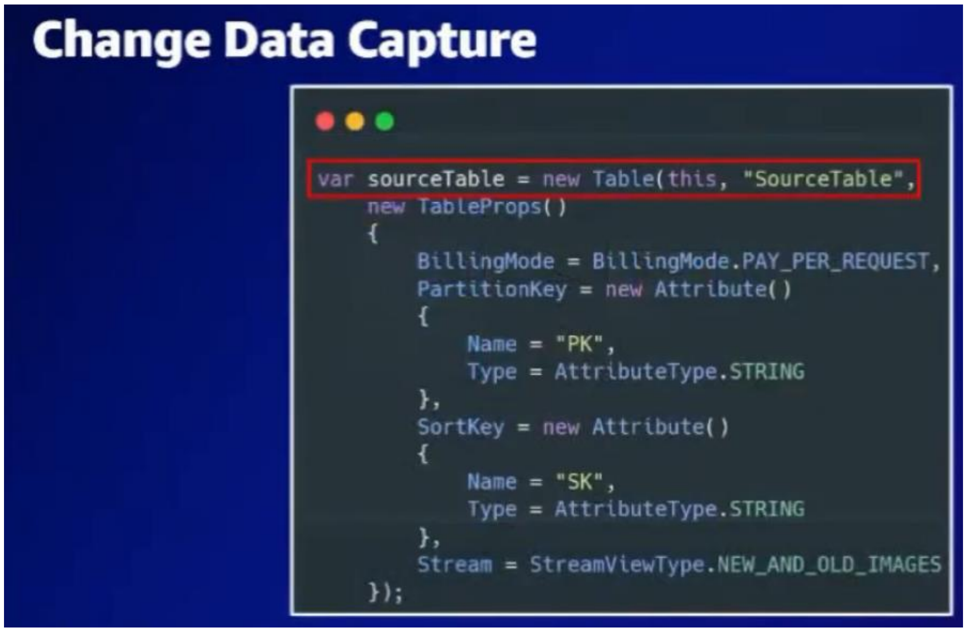
## The How Of Serverless Integration



## Patterns in Code

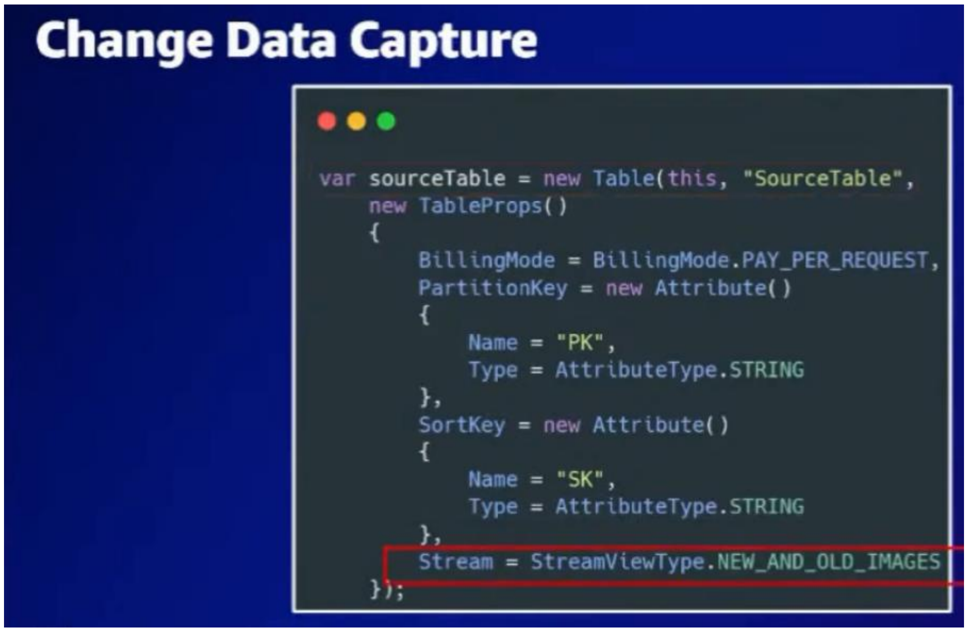


## Change Data Capture



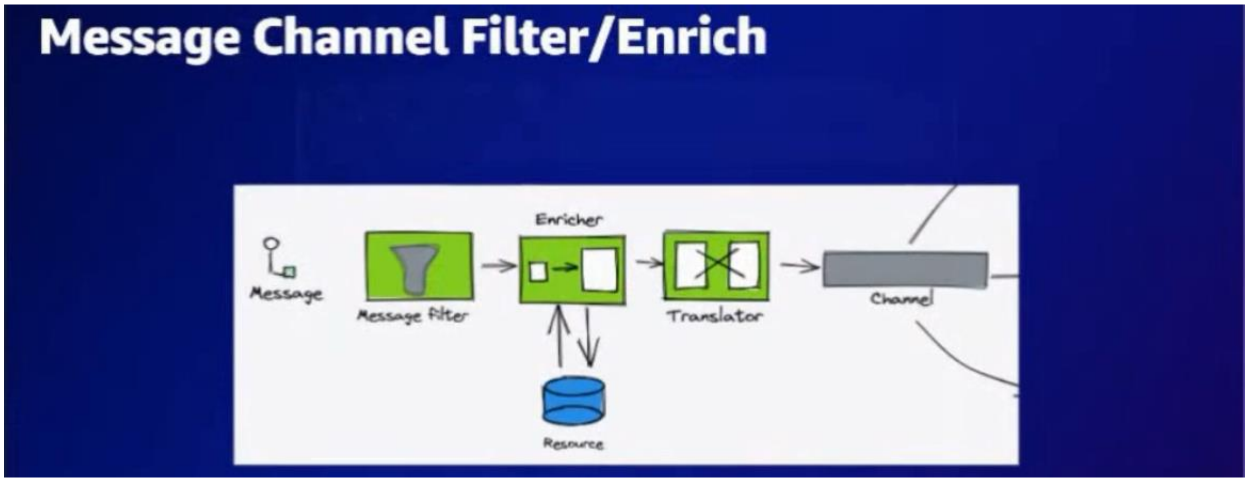
## Change Data Capture

```
var sourceTable = new Table(this, "SourceTable",
    new TableProps()
    {
        BillingMode = BillingMode.PAY_PER_REQUEST,
        PartitionKey = new Attribute()
        {
            Name = "PK",
            Type = AttributeType.STRING
        },
        SortKey = new Attribute()
        {
            Name = "SK",
            Type = AttributeType.STRING
        },
        Stream = StreamViewType.NEW_AND_OLD_IMAGES
    });
```

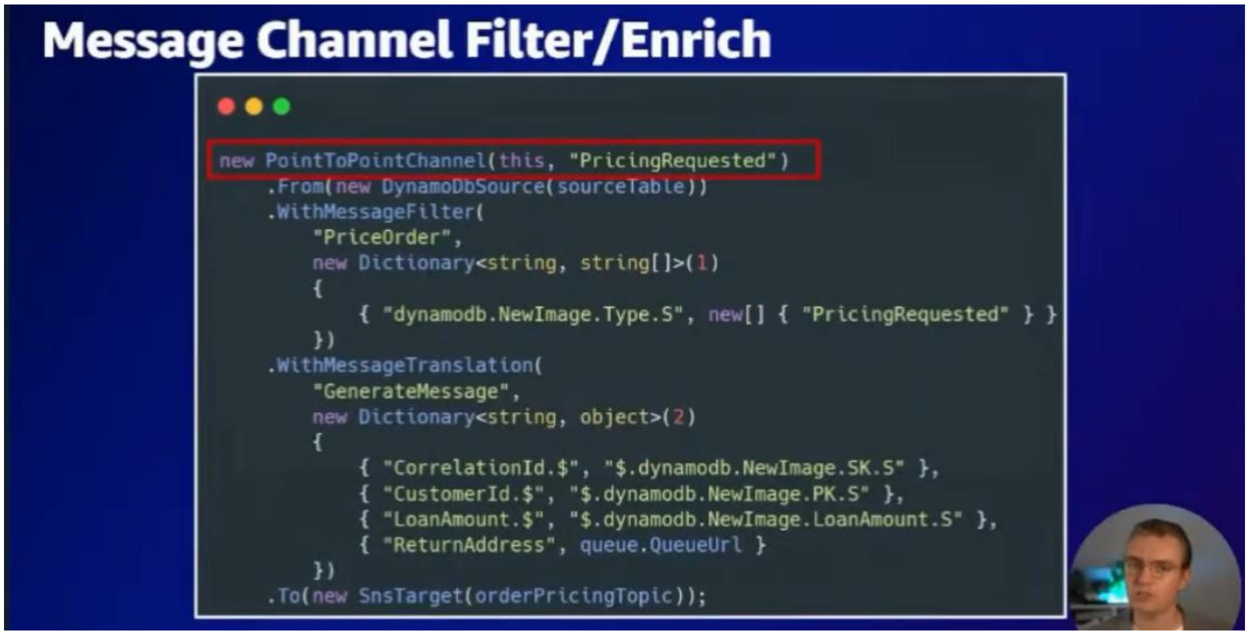


## Change Data Capture

```
var sourceTable = new Table(this, "SourceTable",
    new TableProps()
    {
        BillingMode = BillingMode.PAY_PER_REQUEST,
        PartitionKey = new Attribute()
        {
            Name = "PK",
            Type = AttributeType.STRING
        },
        SortKey = new Attribute()
        {
            Name = "SK",
            Type = AttributeType.STRING
        },
        Stream = StreamViewType.NEW_AND_OLD_IMAGES
    });
```

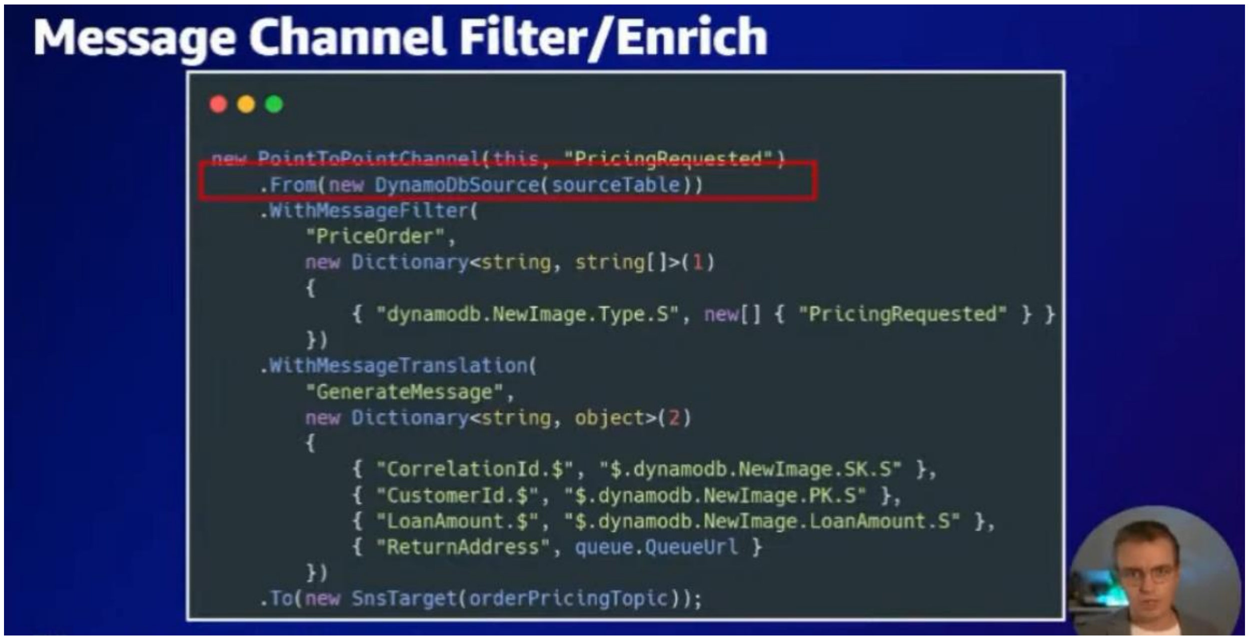


## Message Channel Filter/Enrich



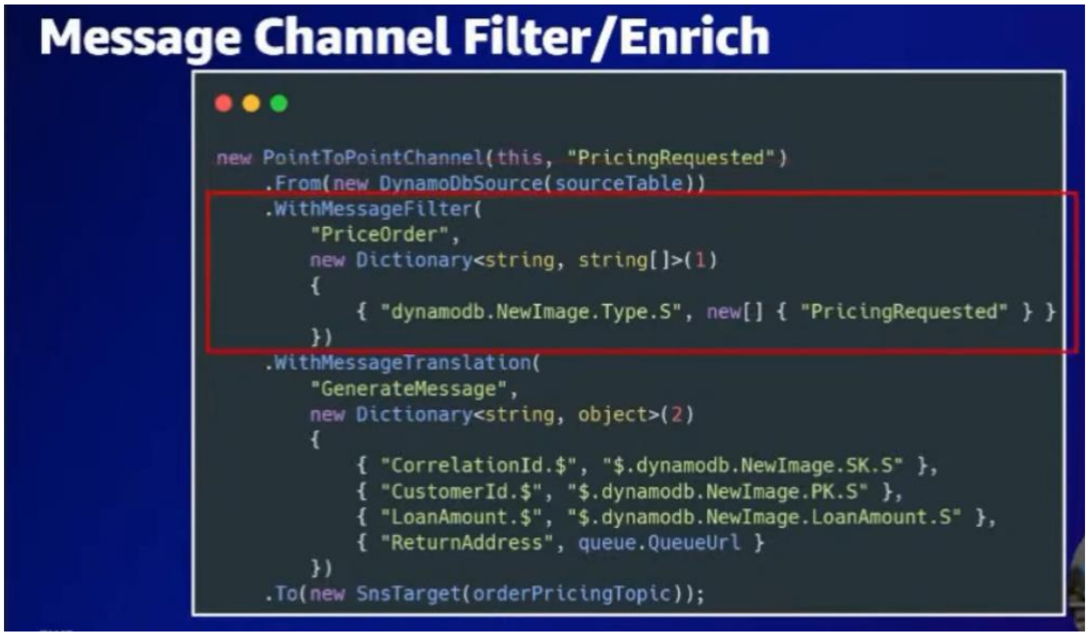
## Message Channel Filter/Enrich

```
new PointToPointChannel(this, "PricingRequested")
    .From(new DynamoDbSource(sourceTable))
    .WithMessageFilter(
        "PriceOrder",
        new Dictionary<string, string[]>(1)
        {
            { "dynamodb.NewImage.Type.S", new[] { "PricingRequested" } }
        })
    .WithMessageTranslation(
        "GenerateMessage",
        new Dictionary<string, object>(2)
        {
            { "CorrelationId.$", "$.dynamodb.NewImage.SK.S" },
            { "CustomerId.$", "$.dynamodb.NewImage.PK.S" },
            { "LoanAmount.$", "$.dynamodb.NewImage.LoanAmount.S" },
            { "ReturnAddress", queue.QueueUrl }
        })
    .To(new SnsTarget(orderPricingTopic));
```

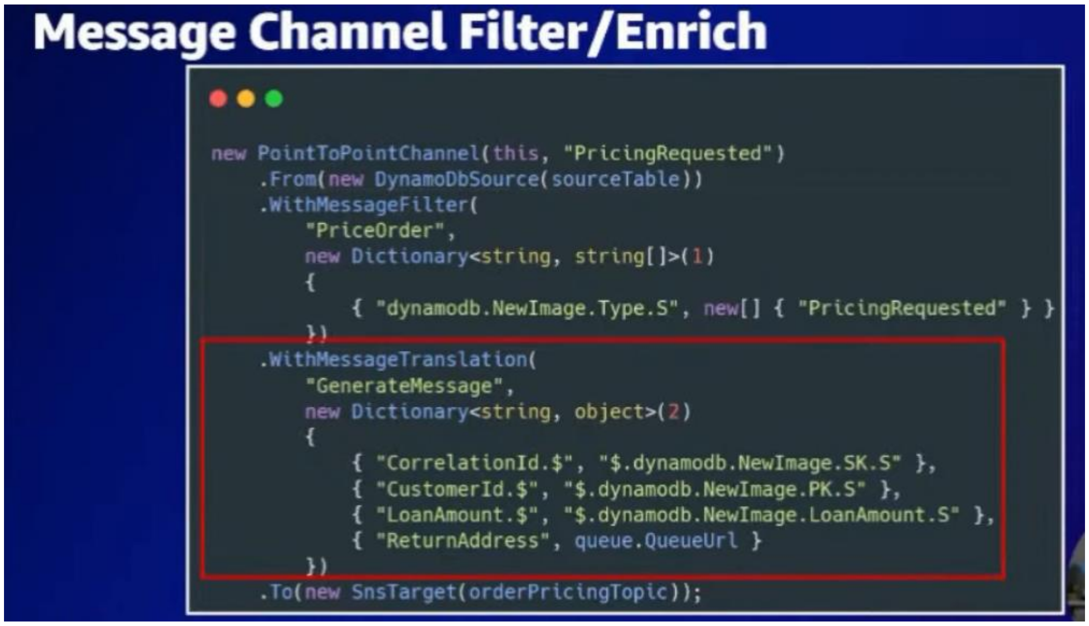


## Message Channel Filter/Enrich

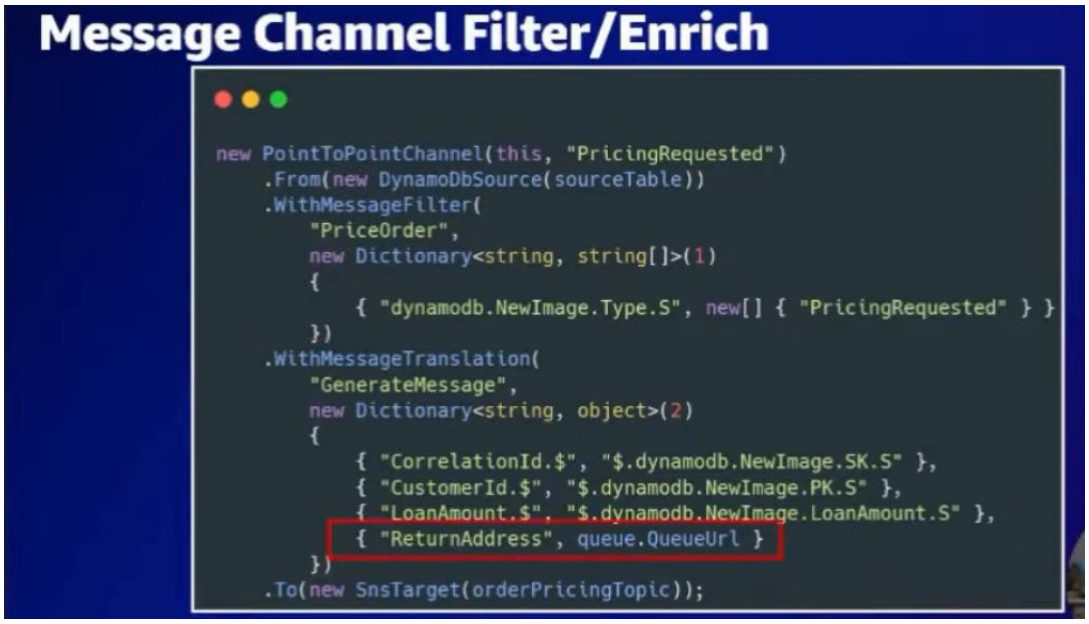
```
new PointToPointChannel(this, "PricingRequested")
    .From(new DynamoDbSource(sourceTable))
    .WithMessageFilter(
        "PriceOrder",
        new Dictionary<string, string[]>(1)
        {
            { "dynamodb.NewImage.Type.S", new[] { "PricingRequested" } }
        })
    .WithMessageTranslation(
        "GenerateMessage",
        new Dictionary<string, object>(2)
        {
            { "CorrelationId.$", "$.dynamodb.NewImage.SK.S" },
            { "CustomerId.$", "$.dynamodb.NewImage.PK.S" },
            { "LoanAmount.$", "$.dynamodb.NewImage.LoanAmount.S" },
            { "ReturnAddress", queue.QueueUrl }
        })
    .To(new SnsTarget(orderPricingTopic));
```


## Message Channel Filter/Enrich

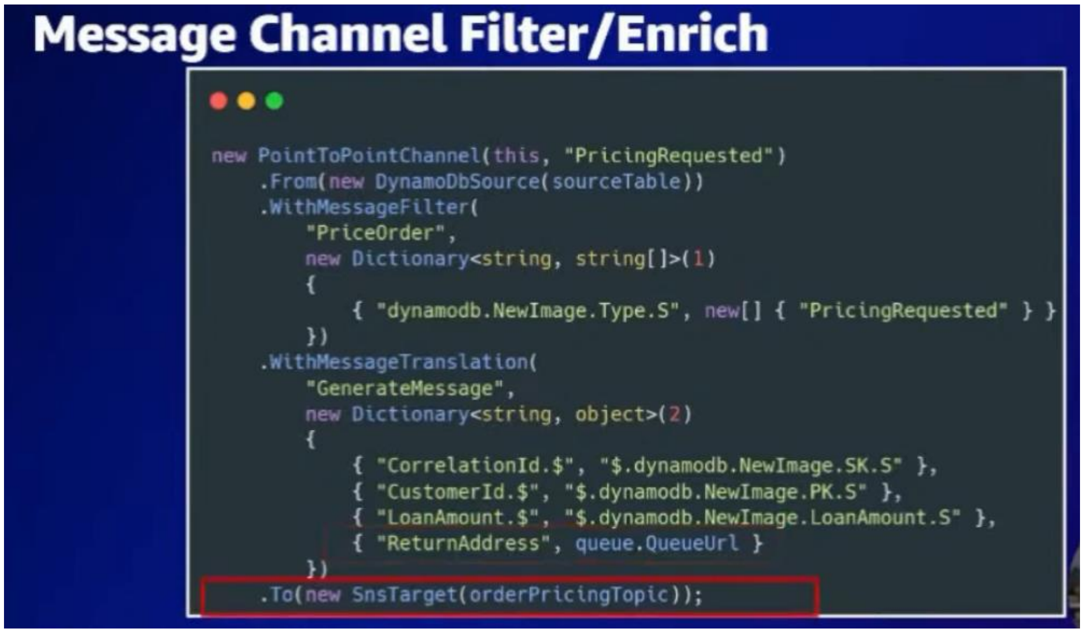
```
new PointToPointChannel(this, "PricingRequested")
    .From(new DynamoDbSource(sourceTable))
    .WithMessageFilter(
        "PriceOrder",
        new Dictionary<string, string[]>(1)
        {
            { "dynamodb.NewImage.Type.S", new[] { "PricingRequested" } }
        })
    .WithMessageTranslation(
        "GenerateMessage",
        new Dictionary<string, object>(2)
        {
            { "CorrelationId.$", "$.dynamodb.NewImage.SK.S" },
            { "CustomerId.$", "$.dynamodb.NewImage.PK.S" },
            { "LoanAmount.$", "$.dynamodb.NewImage.LoanAmount.S" },
            { "ReturnAddress", queue.QueueUrl }
        })
    .To(new SnsTarget(orderPricingTopic));
```


## Message Channel Filter/Enrich

```
new PointToPointChannel(this, "PricingRequested")
    .From(new DynamoDbSource(sourceTable))
    .WithMessageFilter(
        "PriceOrder",
        new Dictionary<string, string[]>(1)
        {
            { "dynamodb.NewImage.Type.S", new[] { "PricingRequested" } }
        })
    .WithMessageTranslation(
        "GenerateMessage",
        new Dictionary<string, object>(2)
        {
            { "CorrelationId.$", "$.dynamodb.NewImage.SK.S" },
            { "CustomerId.$", "$.dynamodb.NewImage.PK.S" },
            { "LoanAmount.$", "$.dynamodb.NewImage.LoanAmount.S" },
            { "ReturnAddress", queue.QueueUrl }
        })
    .To(new SnsTarget(orderPricingTopic));
```

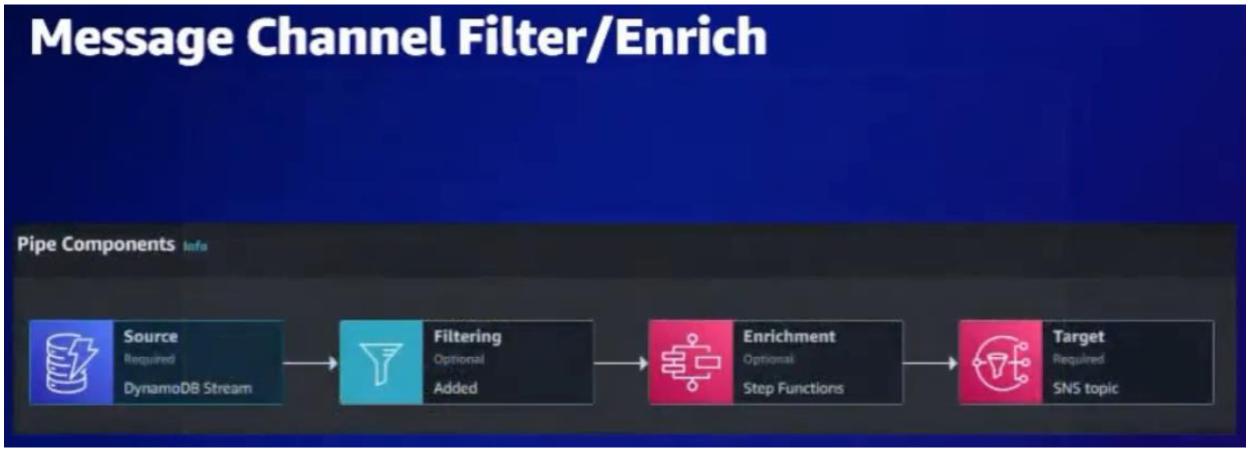

## Message Channel Filter/Enrich

```
new PointToPointChannel(this, "PricingRequested")
    .From(new DynamoDbSource(sourceTable))
    .WithMessageFilter(
        "PriceOrder",
        new Dictionary<string, string[]>(1)
        {
            { "dynamodb.NewImage.Type.S", new[] { "PricingRequested" } }
        })
    .WithMessageTranslation(
        "GenerateMessage",
        new Dictionary<string, object>(2)
        {
            { "CorrelationId.$", "$.dynamodb.NewImage.SK.S" },
            { "CustomerId.$", "$.dynamodb.NewImage.PK.S" },
            { "LoanAmount.$", "$.dynamodb.NewImage.LoanAmount.S" },
            { "ReturnAddress", queue.QueueUrl }
        })
    .To(new SnsTarget(orderPricingTopic));
```

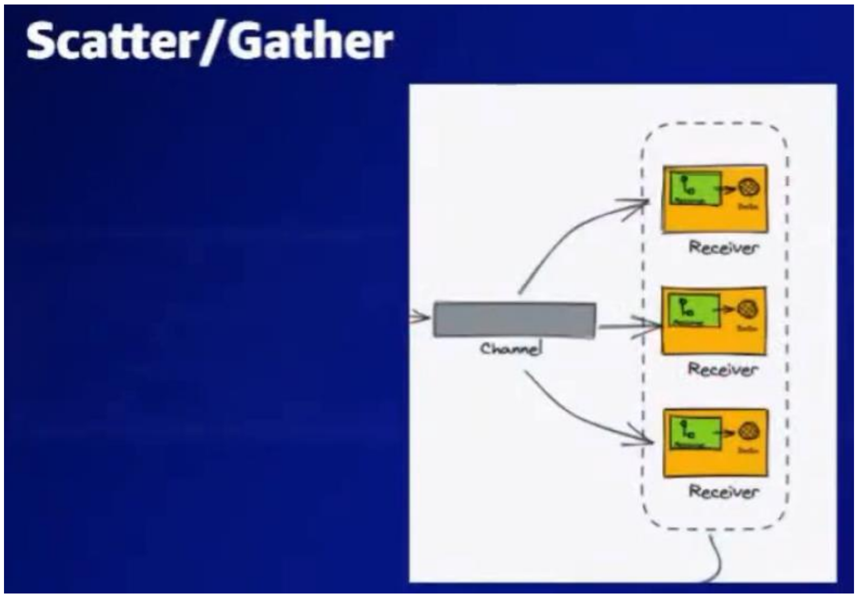


## Message Channel Filter/Enrich

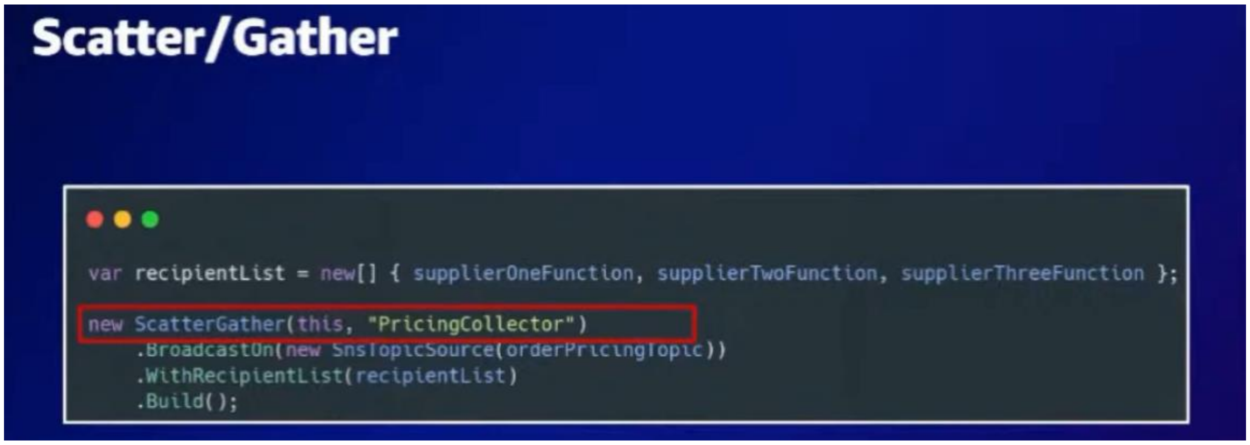
```
new PointToPointChannel(this, "PricingRequested")
    .From(new DynamoDbSource(sourceTable))
    .WithMessageFilter(
        "PriceOrder",
        new Dictionary<string, string[]>(1)
        {
            { "dynamodb.NewImage.Type.S", new[] { "PricingRequested" } }
        })
    .WithMessageTranslation(
        "GenerateMessage",
        new Dictionary<string, object>(2)
        {
            { "CorrelationId.$", "$.dynamodb.NewImage.SK.S" },
            { "CustomerId.$", "$.dynamodb.NewImage.PK.S" },
            { "LoanAmount.$", "$.dynamodb.NewImage.LoanAmount.S" },
            { "ReturnAddress", queue.QueueUrl }
        })
    .To(new SnsTarget(orderPricingTopic));
```



## Message Channel Filter/Enrich

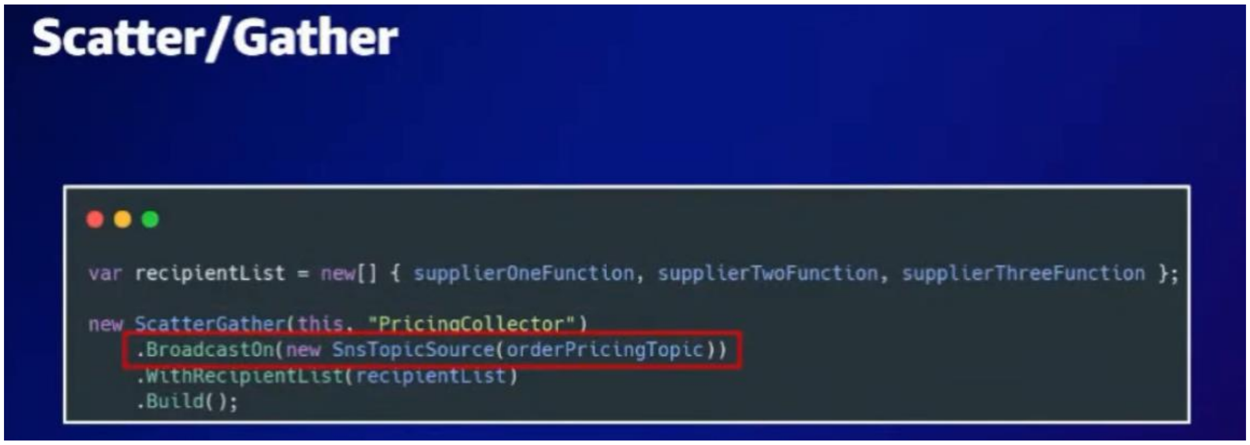


## Scatter/Gather



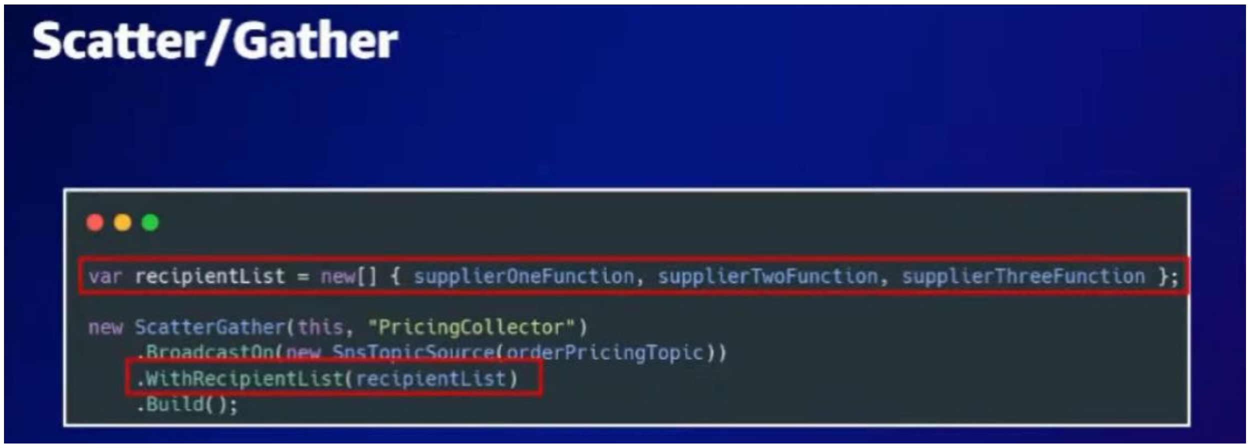
## Scatter/Gather

```
var recipientList = new[] { supplierOneFunction, supplierTwoFunction, supplierThreeFunction };

new ScatterGather(this, "PricingCollector")
    .BroadcastOn(new SnsTopicSource(orderPricingTopic))
    .WithRecipientList(recipientList)
    .Build();
```



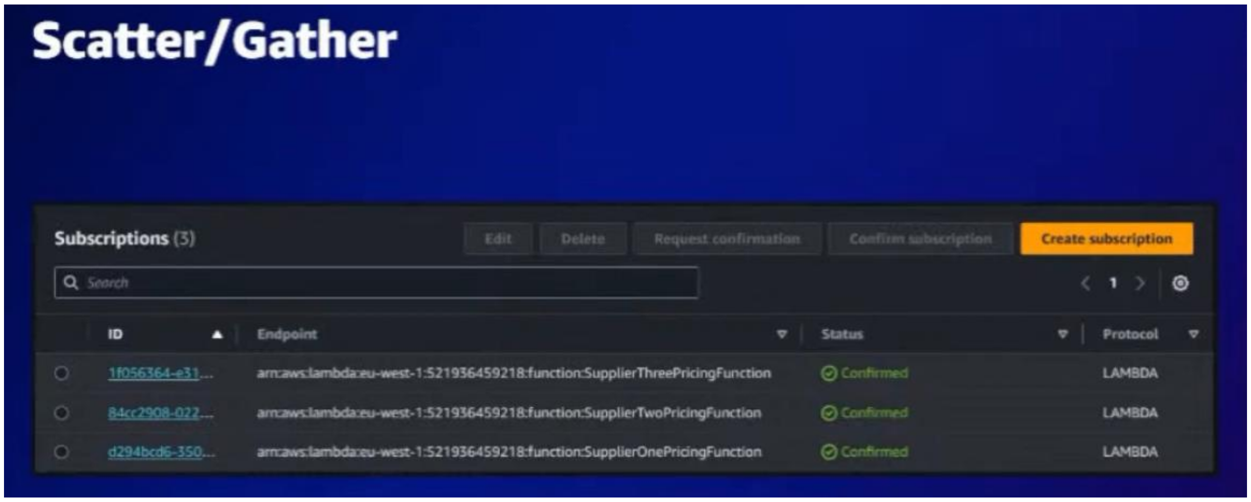
## Scatter/Gather

```
var recipientList = new[] { supplierOneFunction, supplierTwoFunction, supplierThreeFunction };

new ScatterGather(this, "PricingCollector")
    .BroadcastOn(new SnsTopicSource(orderPricingTopic))
    .WithRecipientList(recipientList)
    .Build();
```



## Scatter/Gather

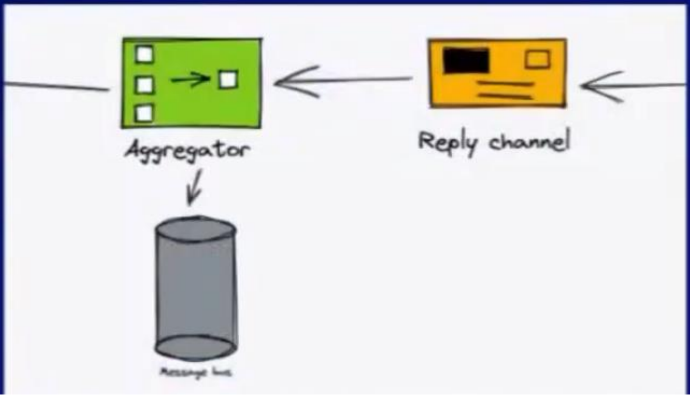
```
var recipientList = new[] { supplierOneFunction, supplierTwoFunction, supplierThreeFunction };

new ScatterGather(this, "PricingCollector")
    .BroadcastOn(new SnsTopicSource(orderPricingTopic))
    .WithRecipientList(recipientList)
    .Build();
```



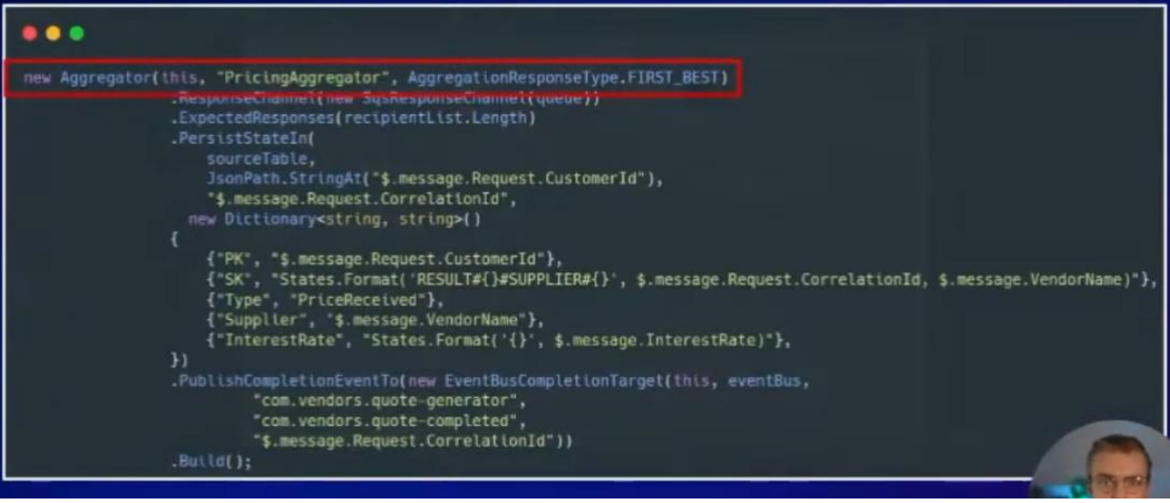
## Scatter/Gather

| ID | Endpoint | Status | Protocol |
|---|---|---|---|
| 1f056364-e31... | arn:aws:lambda:eu-west-1:521936459218:function:SupplierThreePricingFunction | Confirmed | LAMBDA |
| 84cc2908-022... | arn:aws:lambda:eu-west-1:521936459218:function:SupplierTwoPricingFunction | Confirmed | LAMBDA |
| d294bcd6-350... | arn:aws:lambda:eu-west-1:521936459218:function:SupplierOnePricingFunction | Confirmed | LAMBDA |

## Scatter/Gather

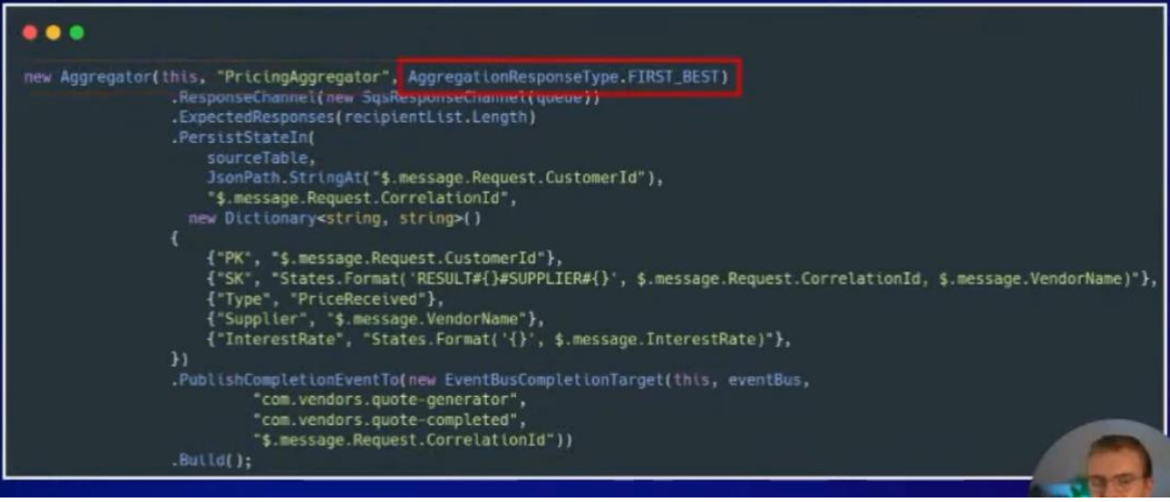


## Scatter/Gather

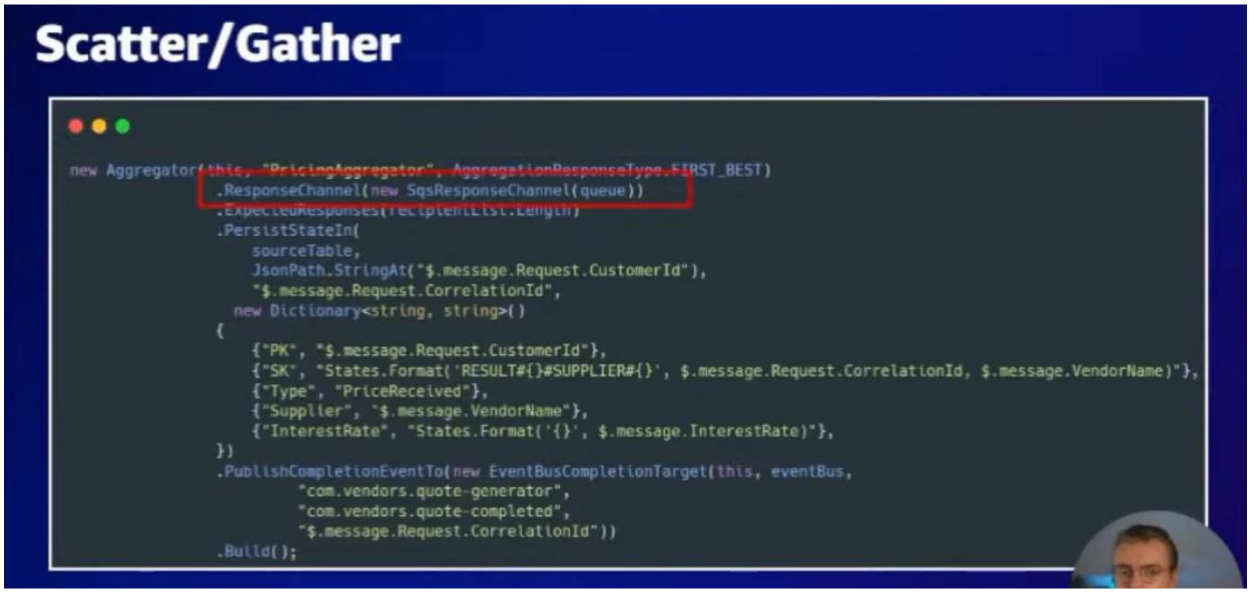


```
new Aggregator(this, "PricingAggregator", AggregationResponseType.FIRST_BEST)
                .ResponseChannel(new SqsResponseChannel(queue))
                .ExpectedResponses(recipientList.Length)
                .PersistStateIn(
                    sourceTable,
                    JsonPath.StringAt("$.message.Request.CustomerId"),
                    "$.message.Request.CorrelationId",
                  new Dictionary<string, string>()
                {
                    {"PK", "$.message.Request.CustomerId"},
                    {"SK", "States.Format('RESULT#{}#SUPPLIER#{}', $.message.Request.CorrelationId, $.message.VendorName)"},
                    {"Type", "PriceReceived"},
                    {"Supplier", "$.message.VendorName"},
                    {"InterestRate", "States.Format('{}', $.message.InterestRate)"},
                })
                .PublishCompletionEventTo(new EventBusCompletionTarget(this, eventBus,
                         "com.vendors.quote-generator",
                         "com.vendors.quote-completed",
                         "$.message.Request.CorrelationId"))
                .Build();
```

## Scatter/Gather

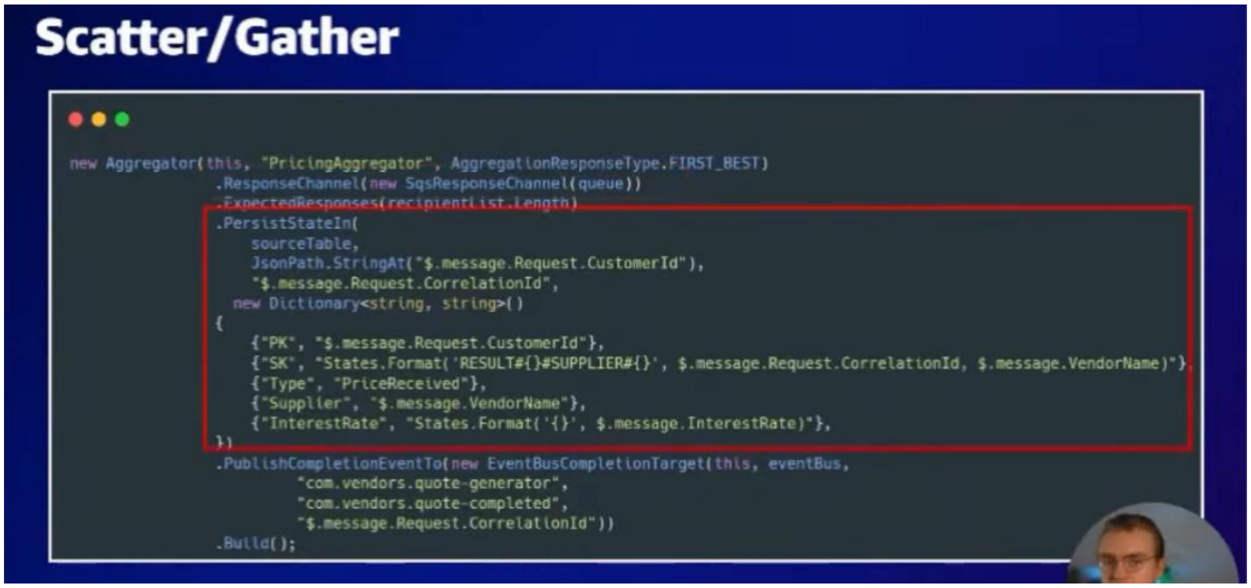


```
new Aggregator(this, "PricingAggregator", AggregationResponseType.FIRST_BEST)
                .ResponseChannel(new SqsResponseChannel(queue))
                .ExpectedResponses(recipientList.Length)
                .PersistStateIn(
                    sourceTable,
                    JsonPath.StringAt("$.message.Request.CustomerId"),
                    "$.message.Request.CorrelationId",
                  new Dictionary<string, string>()
                {
                    {"PK", "$.message.Request.CustomerId"},
                    {"SK", "States.Format('RESULT#{}#SUPPLIER#{}', $.message.Request.CorrelationId, $.message.VendorName)"},
                    {"Type", "PriceReceived"},
                    {"Supplier", "$.message.VendorName"},
                    {"InterestRate", "States.Format('{}', $.message.InterestRate)"},
                })
                .PublishCompletionEventTo(new EventBusCompletionTarget(this, eventBus,
                         "com.vendors.quote-generator",
                         "com.vendors.quote-completed",
                         "$.message.Request.CorrelationId"))
                .Build();
```



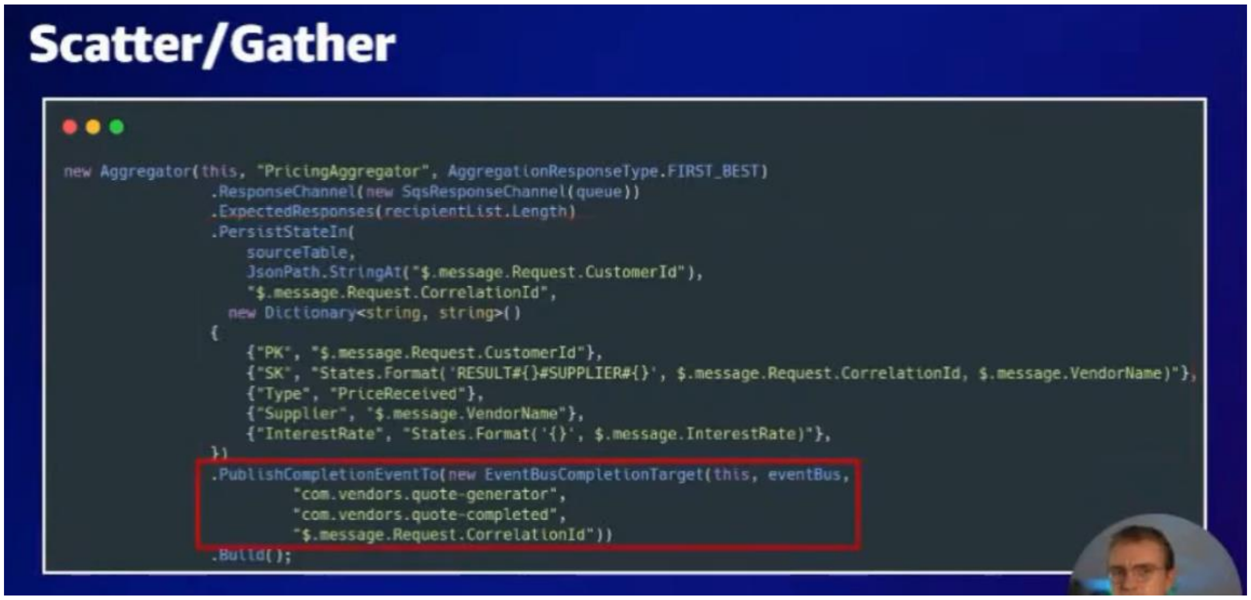
## Scatter/Gather

```
new Aggregator(this, "PricingAggregator", AggregationResponseType.FIRST_BEST)
                .ResponseChannel(new SqsResponseChannel(queue))
                .ExpectedResponses(recipientList.Length)
                .PersistStateIn(
                    sourceTable,
                    JsonPath.StringAt("$.message.Request.CustomerId"),
                    "$.message.Request.CorrelationId",
                  new Dictionary<string, string>()
                {
                    {"PK", "$.message.Request.CustomerId"},
                    {"SK", "States.Format('RESULT#{}#SUPPLIER#{}', $.message.Request.CorrelationId, $.message.VendorName)"},
                    {"Type", "PriceReceived"},
                    {"Supplier", "$.message.VendorName"},
                    {"InterestRate", "States.Format('{}', $.message.InterestRate)"},
                })
                .PublishCompletionEventTo(new EventBusCompletionTarget(this, eventBus,
                         "com.vendors.quote-generator",
                         "com.vendors.quote-completed",
                         "$.message.Request.CorrelationId"))
                .Build();
```



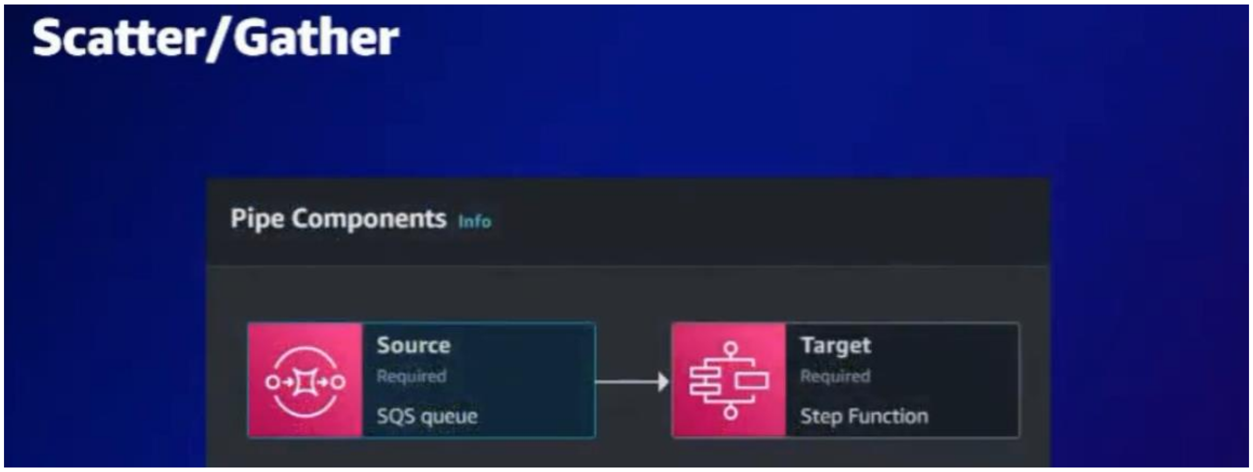
## Scatter/Gather

```
new Aggregator(this, "PricingAggregator", AggregationResponseType.FIRST_BEST)
                .ResponseChannel(new SqsResponseChannel(queue))
                .ExpectedResponses(recipientList.Length)
                .PersistStateIn(
                    sourceTable,
                    JsonPath.StringAt("$.message.Request.CustomerId"),
                    "$.message.Request.CorrelationId",
                  new Dictionary<string, string>()
                {
                    {"PK", "$.message.Request.CustomerId"},
                    {"SK", "States.Format('RESULT#{}#SUPPLIER#{}', $.message.Request.CorrelationId, $.message.VendorName)"},
                    {"Type", "PriceReceived"},
                    {"Supplier", "$.message.VendorName"},
                    {"InterestRate", "States.Format('{}', $.message.InterestRate)"},
                })
                .PublishCompletionEventTo(new EventBusCompletionTarget(this, eventBus,
                         "com.vendors.quote-generator",
                         "com.vendors.quote-completed",
                         "$.message.Request.CorrelationId"))
                .Build();
```



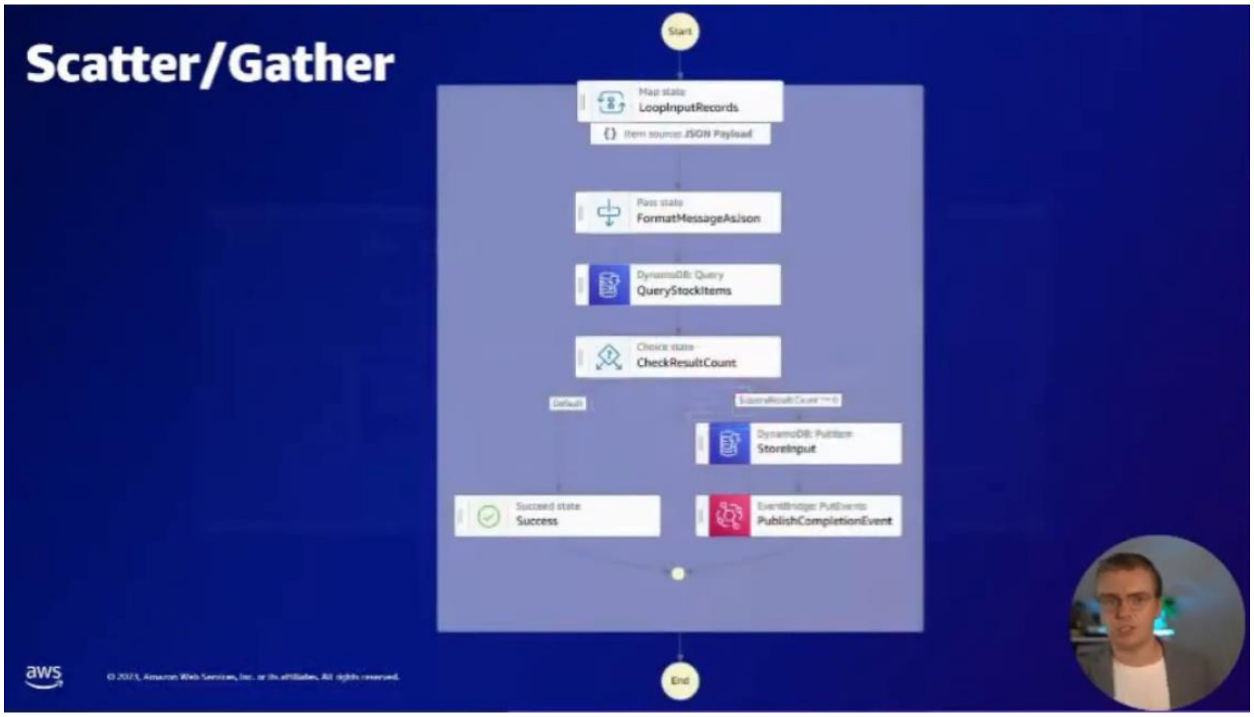
## Scatter/Gather

```
new Aggregator(this, "PricingAggregator", AggregationResponseType.FIRST_BEST)
                .ResponseChannel(new SqsResponseChannel(queue))
                .ExpectedResponses(recipientList.Length)
                .PersistStateIn(
                    sourceTable,
                    JsonPath.StringAt("$.message.Request.CustomerId"),
                    "$.message.Request.CorrelationId",
                  new Dictionary<string, string>()
                {
                    {"PK", "$.message.Request.CustomerId"},
                    {"SK", "States.Format('RESULT#{}#SUPPLIER#{}', $.message.Request.CorrelationId, $.message.VendorName)"},
                    {"Type", "PriceReceived"},
                    {"Supplier", "$.message.VendorName"},
                    {"InterestRate", "States.Format('{}', $.message.InterestRate)"},
                })
                .PublishCompletionEventTo(new EventBusCompletionTarget(this, eventBus,
                         "com.vendors.quote-generator",
                         "com.vendors.quote-completed",
                         "$.message.Request.CorrelationId"))
                .Build();
```

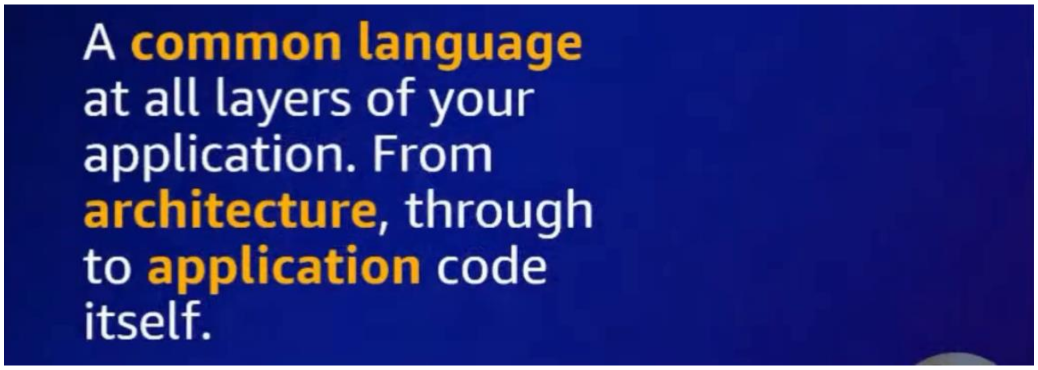


## Scatter/Gather

Pipe Components Info



## Scatter/Gather



A **common language** at all layers of your application. From **architecture**, through to **application** code itself.

**Codified** best practices

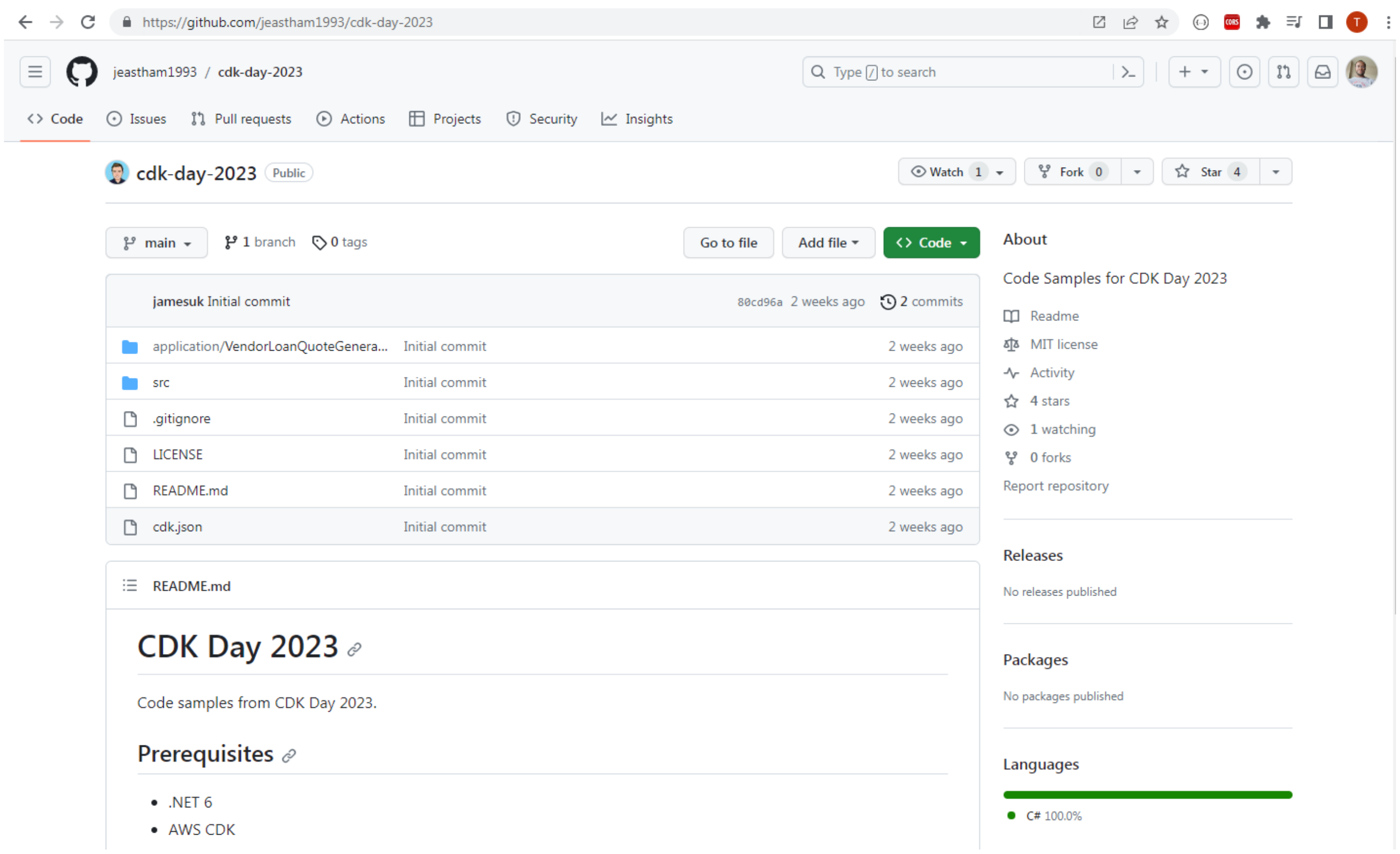
**Simplified** developer experience

Thank You!

/in/James-Eastham

@plantpowerjames

https://github.com/jeastham1993/cdk-day-2023



https://github.com/jeastham1993/cdk-day-2023

jeastham1993 / cdk-day-2023

cdk-day-2023 Public

main · 1 branch · 0 tags

jamesuk Initial commit · 80cd96a 2 weeks ago · 2 commits

| | | |
|---|---|---|
| application/VendorLoanQuoteGenera... | Initial commit | 2 weeks ago |
| src | Initial commit | 2 weeks ago |
| .gitignore | Initial commit | 2 weeks ago |
| LICENSE | Initial commit | 2 weeks ago |
| README.md | Initial commit | 2 weeks ago |
| cdk.json | Initial commit | 2 weeks ago |

README.md

CDK Day 2023

Code samples from CDK Day 2023.

Prerequisites

- .NET 6
- AWS CDK

About

Code Samples for CDK Day 2023

Readme · MIT license · Activity · 4 stars · 1 watching · 0 forks

Releases: No releases published

Packages: No packages published

Languages: C# 100.0%

[https://github.com/jeastham1993/cdk-day-2023](https://github.com/jeastham1993/cdk-day-2023)