
AWS
re:Invent

SVS402

Decoupling serverless workloads with Amazon EventBridge

James Beswick

Senior Developer Advocate

AWS Serverless

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

About me



James Beswick

- Email: jbeswick@amazon.com
- Twitter: @jbesw

Senior Developer Advocate – AWS Serverless

Self-confessed serverless geek

Previously

- Software Developer
- Product Manager

Agenda

The challenges with microservices

Integration patterns with Amazon EventBridge

Extending the AWS Cloud with (software as a service) SaaS partners

Controlling access to resources

Decoupling architecture with events and AWS Serverless Application Model (SAM)



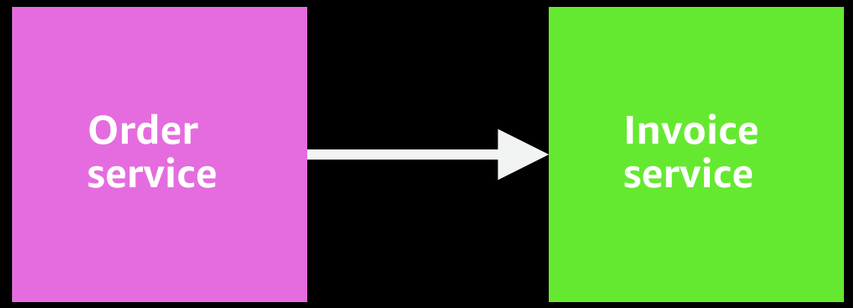
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The challenges with microservices

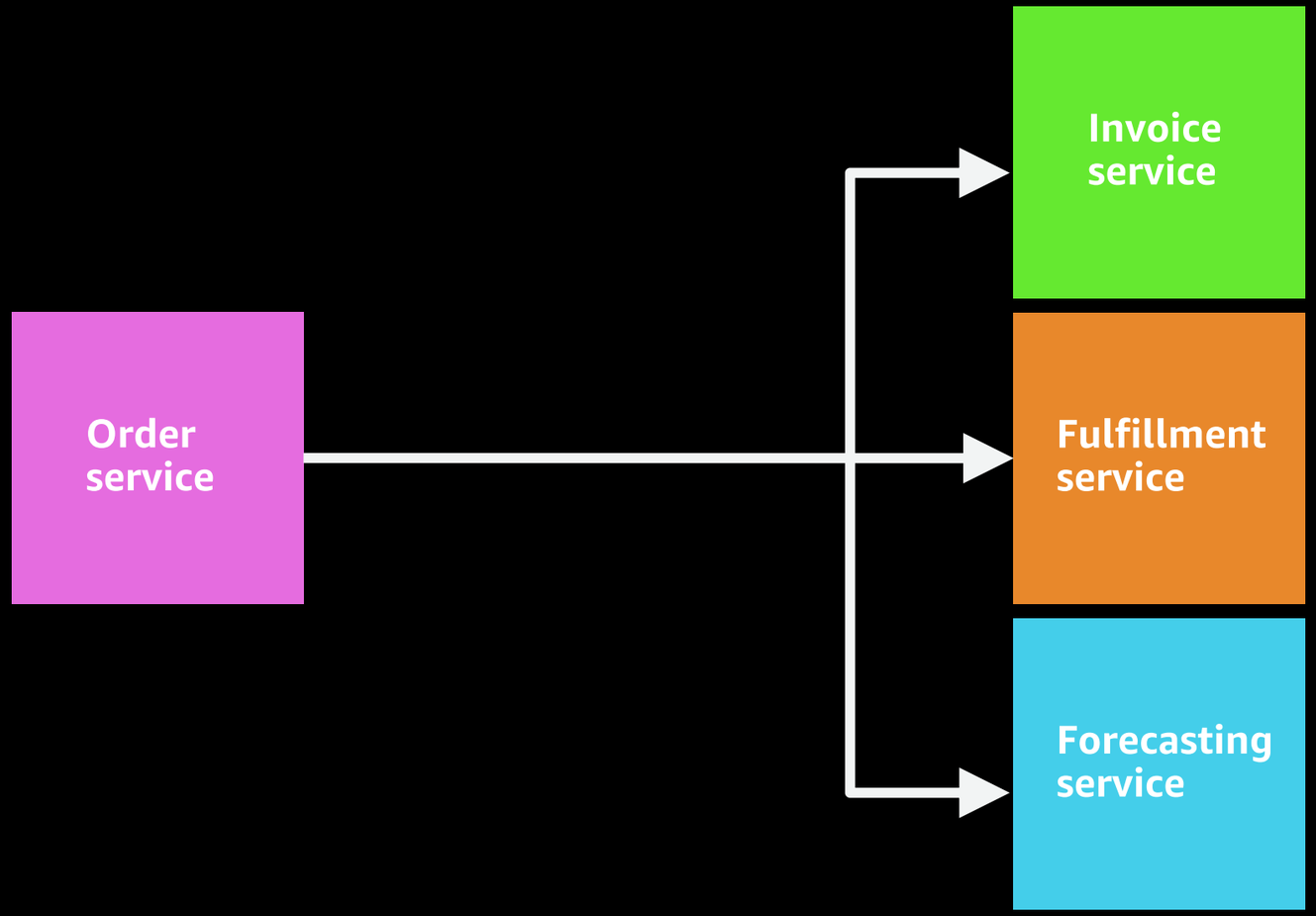


© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

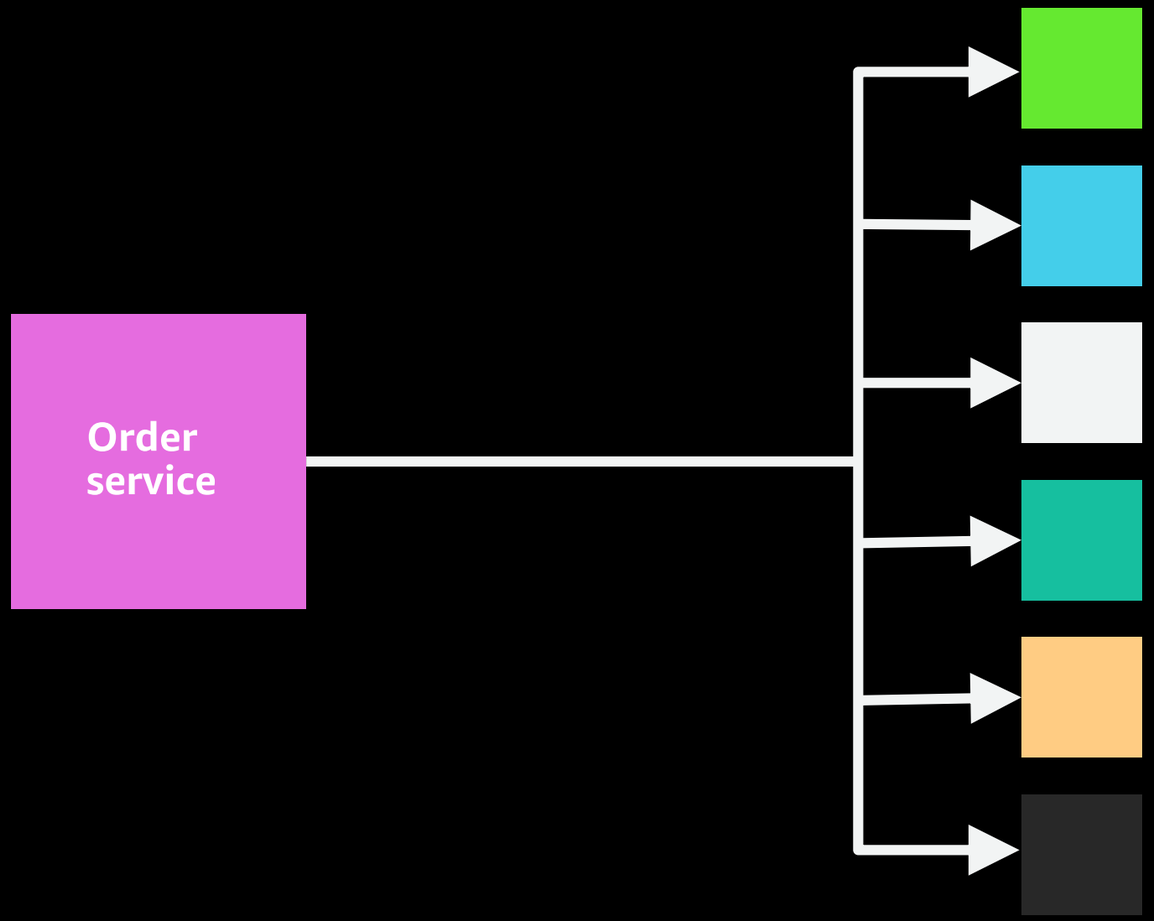
V1: Simple integration



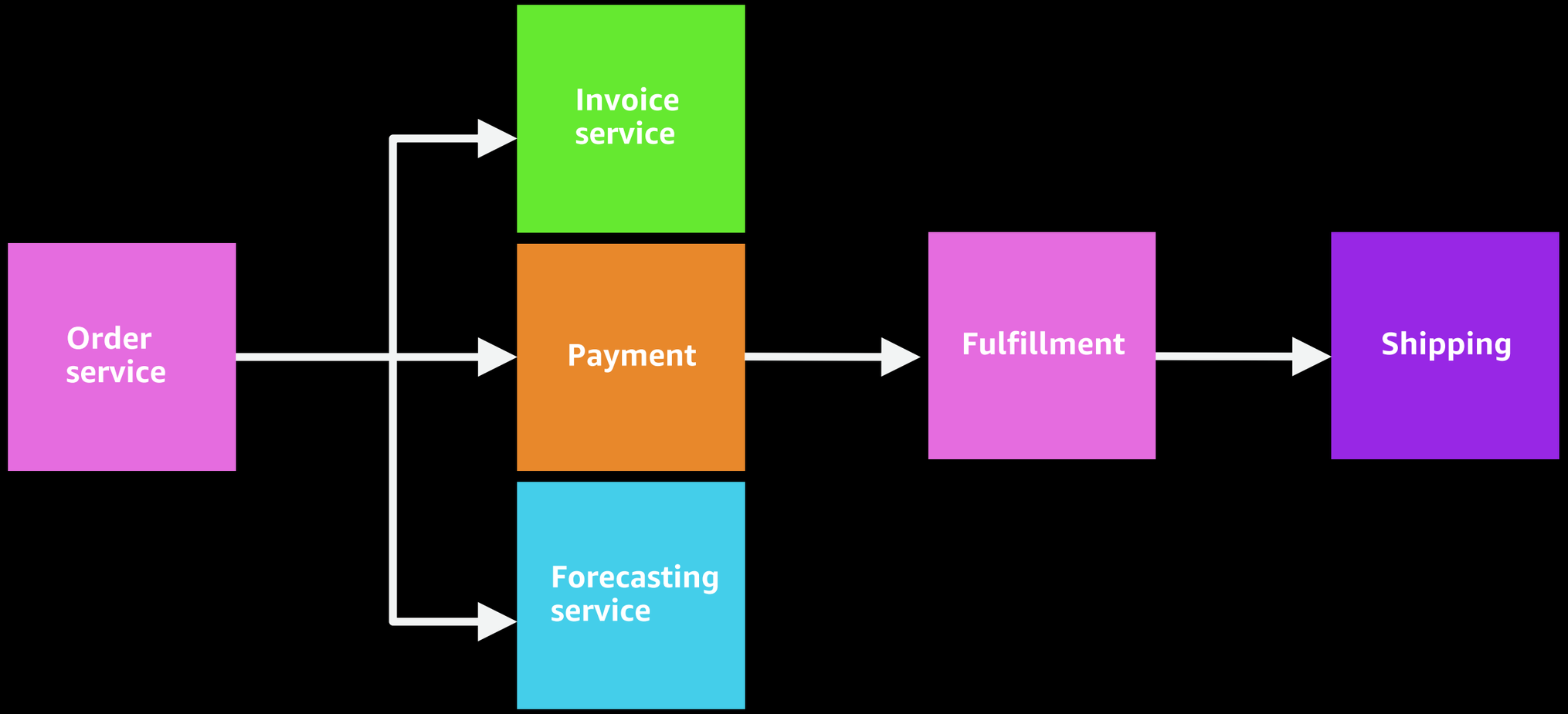
V2: More integrations are added



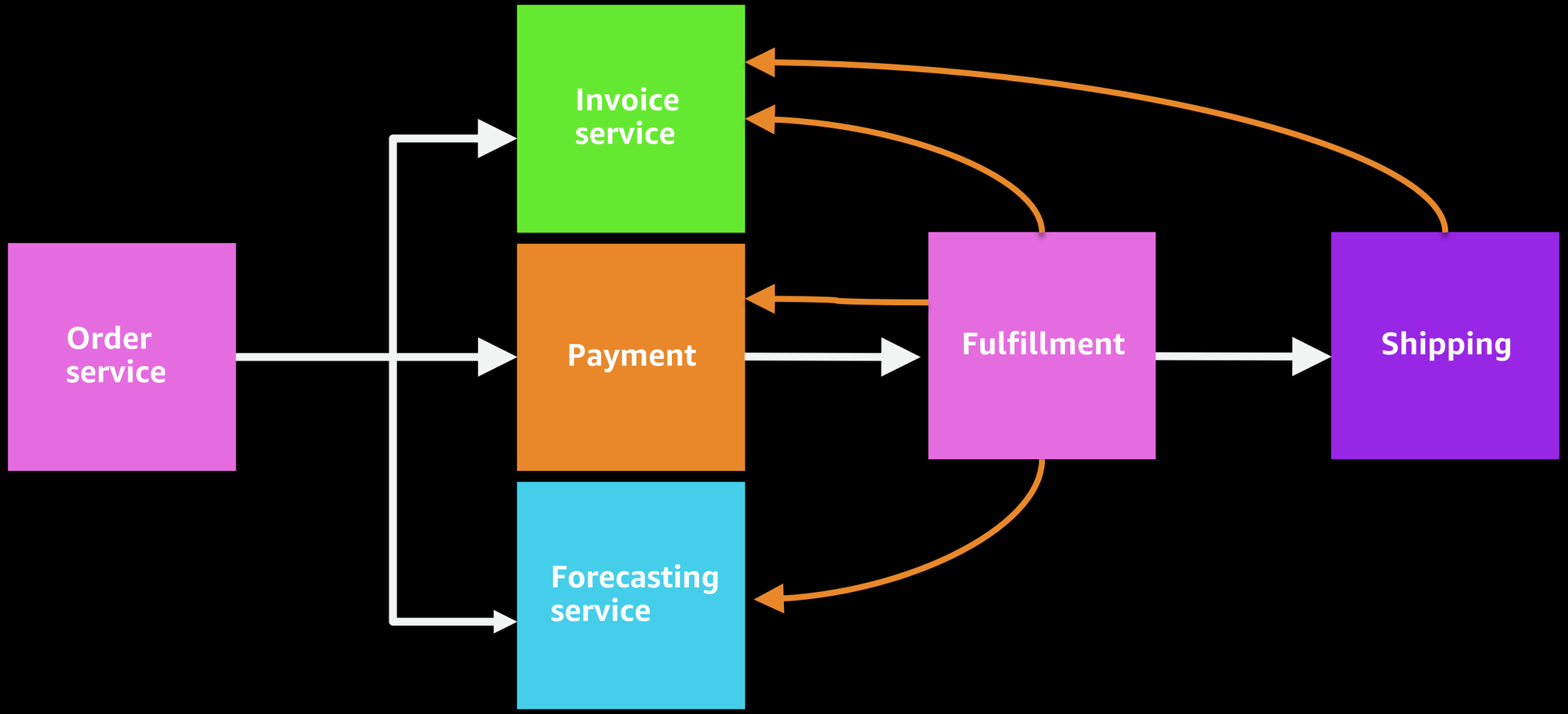
V3, V4, V5 . . . Now tightly coupled



The happy path works well



. . . but exception handling can be messy



What are the benefits?



Loose coupling



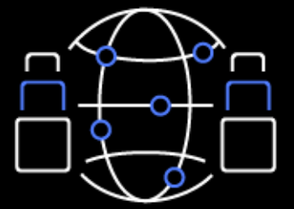
Scalability



Extensibility



Complexity



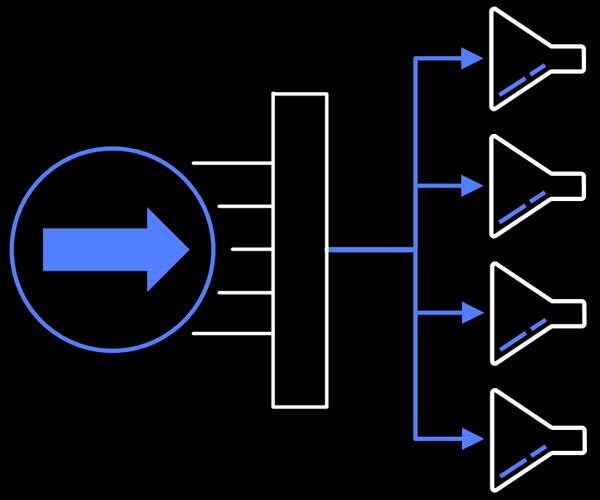
Availability



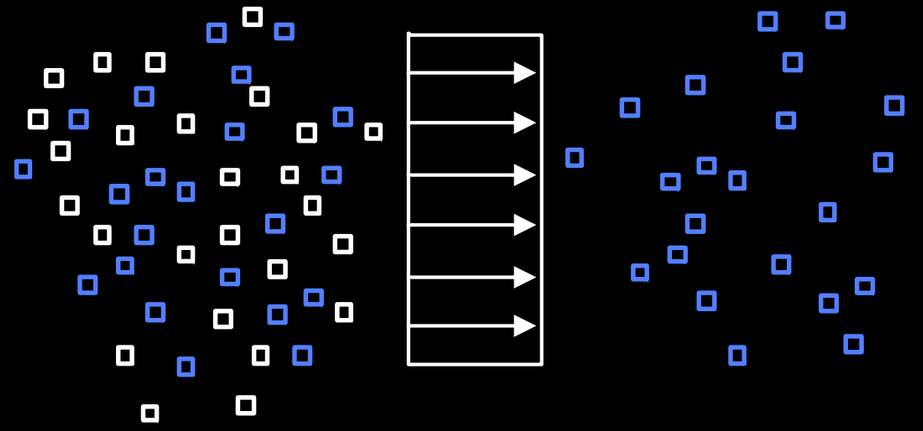
Amazon EventBridge

A serverless event bus service
for SaaS and AWS services

Key features of an event bus

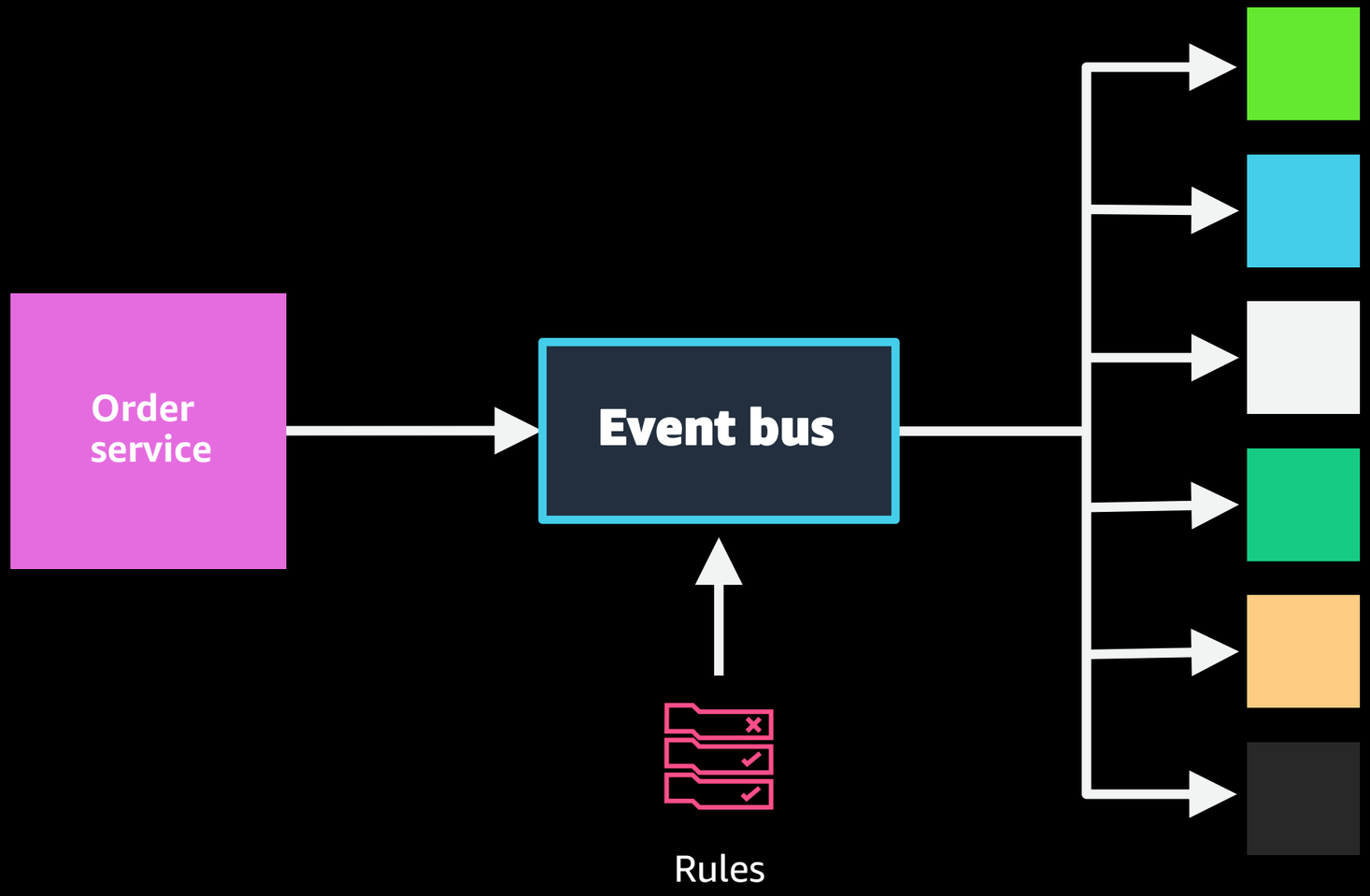


Decouples producers
from consumers



Filters and
routes events

How to use an event bus for decoupling



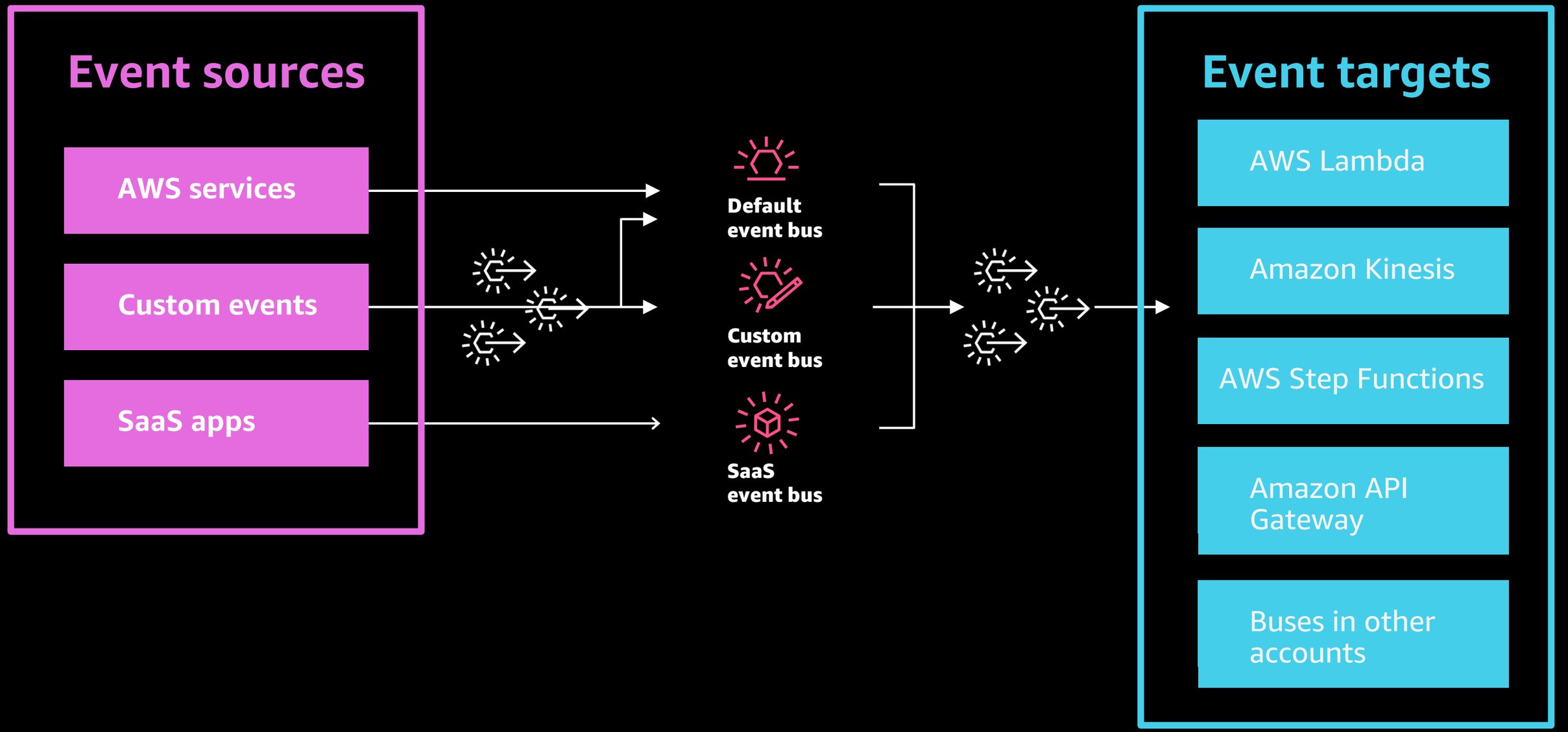
Integration patterns with Amazon EventBridge



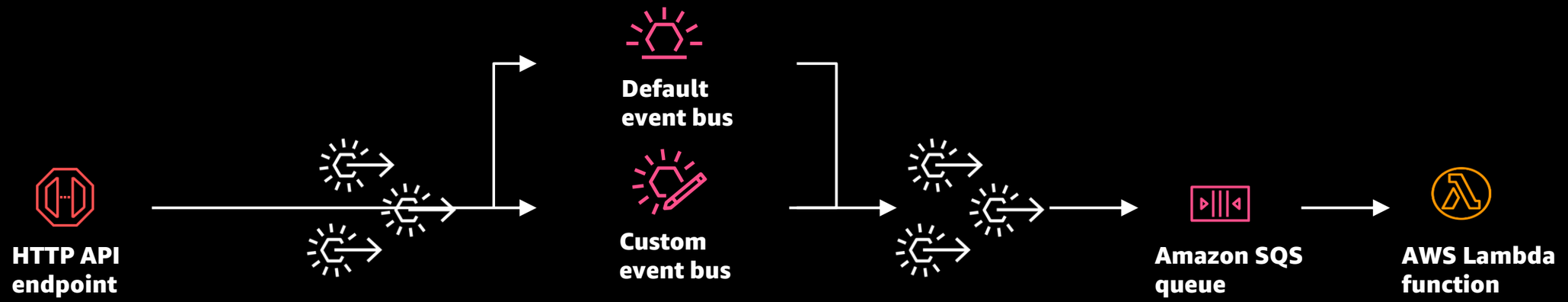
AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Sources and targets for Amazon EventBridge

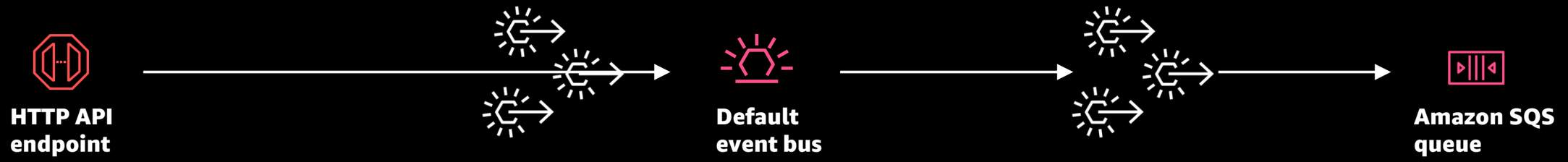


1. Building scalable webhooks



Download and deploy this code at
https://github.com/aws-samples/decoupling-with-amazon-eventbridge

1. Building scalable webhooks



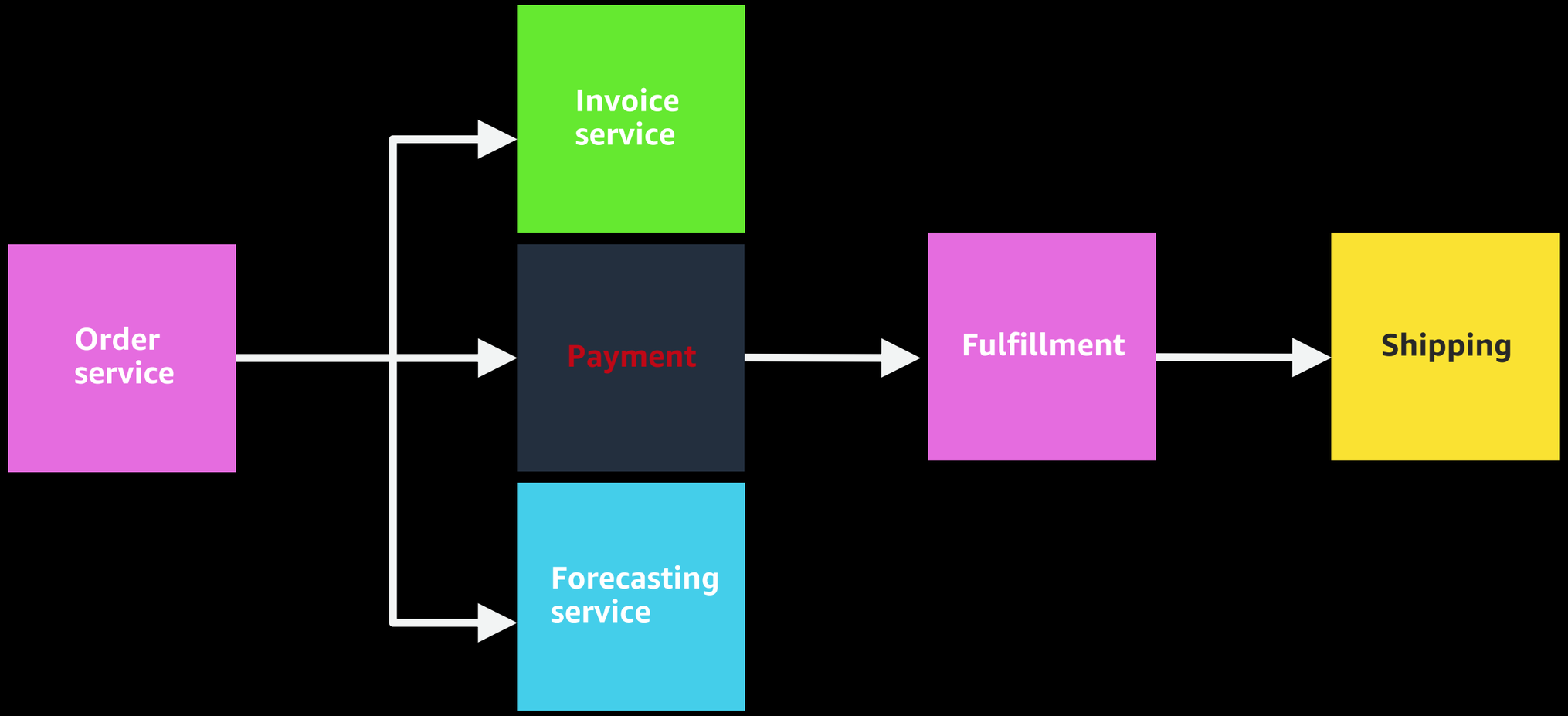

```
paths:
  /webhook:
    post:
       x-amazon-apigateway-integration:
        integrationSubtype: "EventBridge-
PutEvents"
        credentials:
          Fn::GetAtt: [MyHttpApiRole, Arn]
        requestParameters:
          Detail: "$request.body.Detail"
          DetailType: MyDetailType
          Source: myWebhook
```

```
EventRule:
  Type: AWS::Events::Rule
  Properties:
    EventPattern:
      source:
        - "myWebhook"
    Targets:
      - Arn: !GetAtt MySqsQueue.Arn
        Id: "SQSqueue"
```

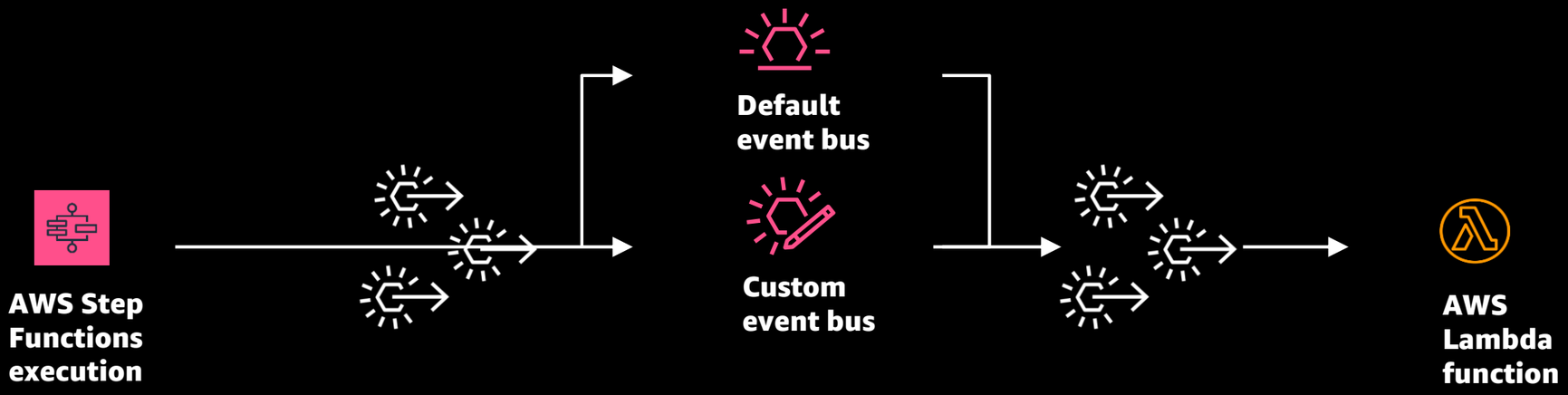


Download and deploy this code at
https://github.com/aws-samples/decoupling-with-amazon-eventbridge

2. Responding to workflows



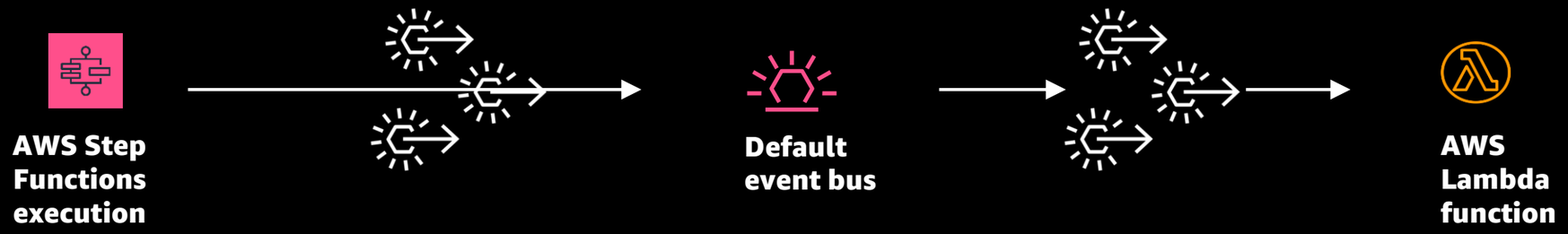
2. Responding to workflows





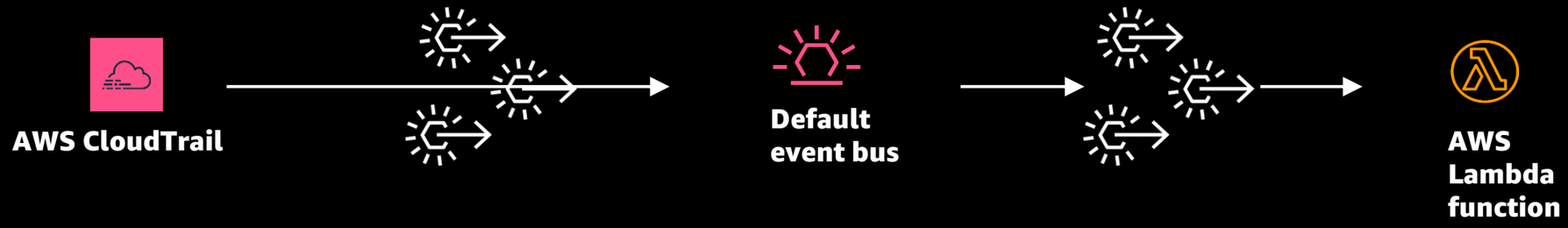
Download and deploy this code at
https://github.com/aws-samples/decoupling-with-amazon-eventbridge

2. Responding to workflows




```
{
  "detail-type": "Step Functions Execution Status Change",
  "source": "aws.states",
  "detail": {
      "executionArn": "arn:aws:states:us-east-2:123456789012:
execution:placeOrderStateMachine-o7Y0cFTSsUrH:fc76cfb4-1234-1234-f22f-
94188a350d38",
      "stateMachineArn": "arn:aws:states:us-east-
2:123456789012:stateMachine:v1StateMachine-1234FTSsUrH",
      "name": "fc76cfb4-1234-1234-1234-94188a350d12",
      "status": "RUNNING",
      "startDate": 1598889181228,
      "input": "{\"OrderID\": \"12345678\"}",
  }
}
```

3. Integrating with AWS CloudTrail

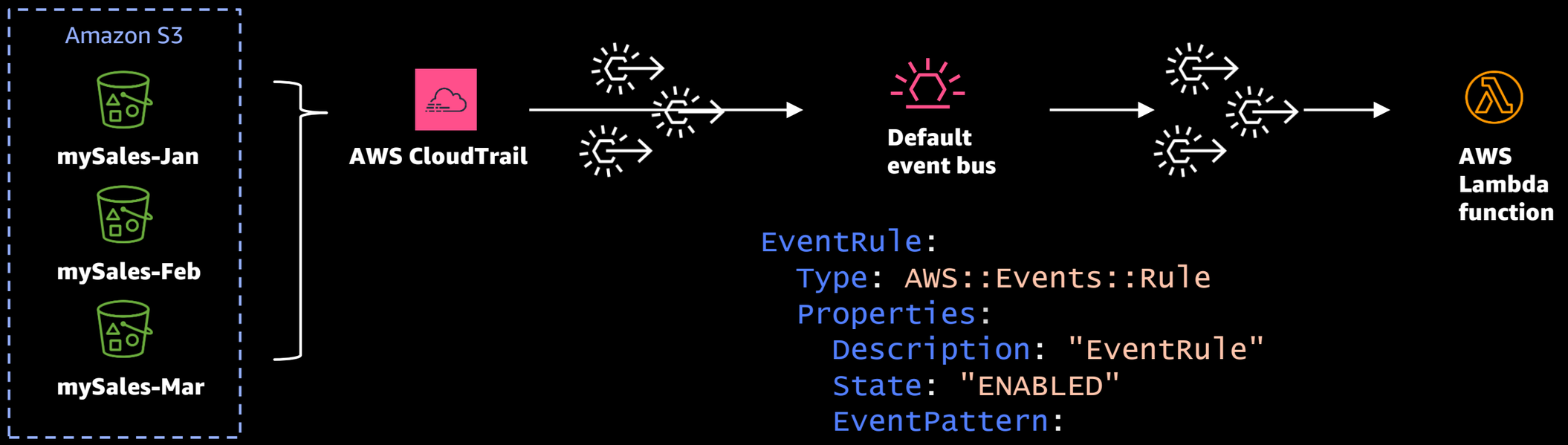


Any service that logs to AWS CloudTrail can send events to Amazon EventBridge

AWS CloudTrail logging must be configured before events will appear

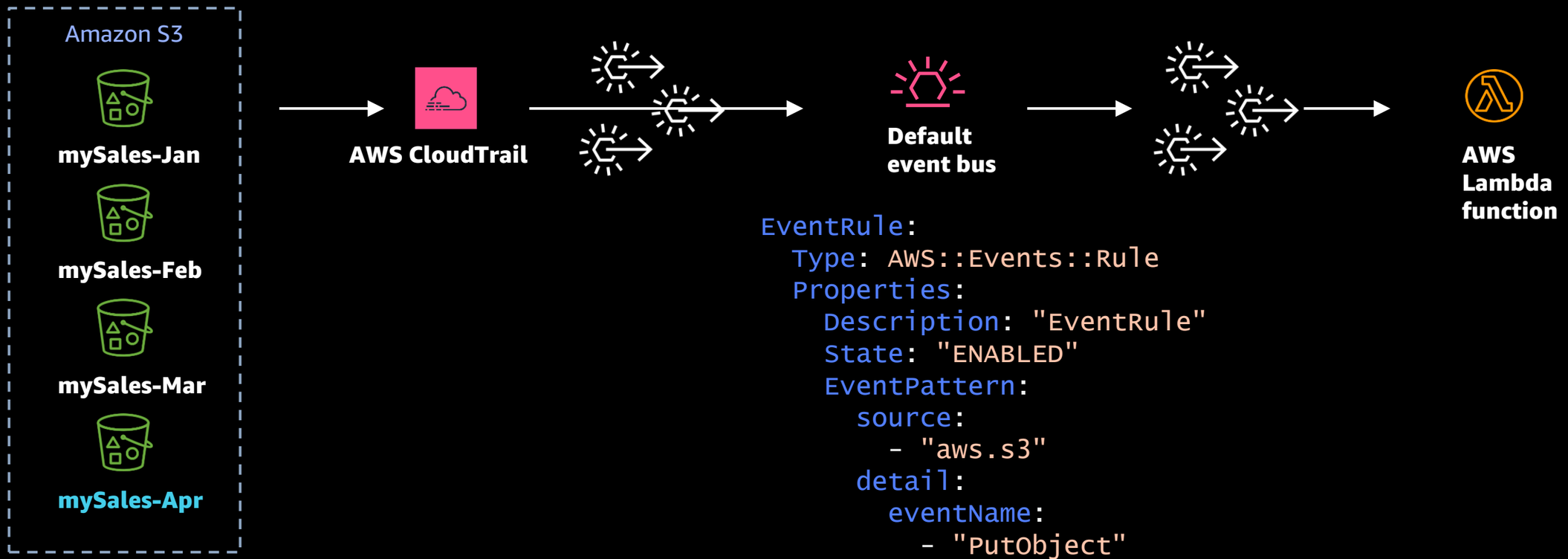
```
{
  "eventSource":"aws:s3",
  "eventName":"ObjectCreated:Put",
  "s3":{
     "bucket":{
        "name":"mybucket",
        "arn":"arn:aws:s3:::mybucket"
     },
     "object":{
        "key":"myImage.jpg",
        "size":1024,
        ...
```

Multiple S3 buckets as an event source




```
EventRule:
  Type: AWS::Events::Rule
  Properties:
    Description: "EventRule"
    State: "ENABLED"
    EventPattern:
      source:
        - "aws.s3"
      detail:
        eventName:
          - "PutObject"
        requestParameters:
          bucketName:
            - 'mySales-Jan'
            - 'mySales-Feb'
            - 'mySales-Mar'
```

Dynamic S3 buckets as an event source




```
EventRule:
  Type: AWS::Events::Rule
  Properties:
    Description: "EventRule"
    State: "ENABLED"
    EventPattern:
      source:
        - "aws.s3"
      detail:
        eventName:
          - "PutObject"
        requestParameters:
          bucketName:
            - "prefix": "mySales-"
```



Download and deploy this code at
https://github.com/aws-samples/decoupling-with-amazon-eventbridge

SaaS partner integration



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Integrating with SaaS providers



Expanding partner list

30 integrated partners with more to come



No polling

Replaces polling and basic webhooks



Partner event bus

Events appear directly in your AWS account



More secure

No over-the-internet API keys



Fully managed

Scales automatically

Resources for building solutions

E-commerce with Shopify
https://aws.amazon.com/eventbridge/integrations/shopify/

Customer Support with Zendesk
https://aws.amazon.com/eventbridge/integrations/zendesk/

Continuous integration (CI) / continuous delivery (CD) with Buildkite
https://aws.amazon.com/eventbridge/integrations/buildkite/

Controlling access to resources



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Identity-based policies with Identity and Access Management (IAM)

Uses familiar policy elements (Resource, Action, Effect, Role)

Attach permission to user / group

Attach permission to role

AWS managed policies (CloudWatchEventsFullAccess)

Amazon EventBridge API actions

```
events:DeleteRule
events:DescribeEventBus
events:DescribeRule
events:DisableRule
events:EnableRule
events:ListRuleNamesByTarget
events:ListRules
events:ListTagsForResource
events:ListTargetsByRule
events:PutEvents
events:PutPermission
events:PutRule
events:PutTargets
events:RemovePermission
events:RemoveTargets
events:TestEventPattern
```

Resource-based policies with IAM

Grant Amazon Simple Notification Service (SNS) topic access

```
{
    "Sid": "PublishEventsToMyTopic",
    "Effect": "Allow",
    "Principal": {
      "Service": "events.amazonaws.com"
    },
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:region:account-
id:topic-name",
   "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:events:regio
n:account-id:rule/rule-name"
      }
    }
  }
```

Grant schema access to an account

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Sid": "Test",
          "Effect": "Allow",
          "Action": "schemas:*"
          "Principal": {
              "AWS": [ "123456789012" ]
          },
          "Resource": [
              "arn:aws:schemas:us-east-
1:012345678901:registry/default",
              "arn:aws:schemas:us-east-
1:012345678901:schema/default*"
          ]
      }
  ]
}
```

Tag policies

Supports policies based on tags

This rule restricts access to Amazon EventBridge resources

It denies all resources that are tagged environment / production

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Deny",
          "Action": [
              "events:PutRule",
              "events:DescribeRule",
              "events:DeleteRule",
              "events:CreateEventBus",
              "events:DescribeEventBus"
              "events:DeleteEventBus"
          ],
          "Resource": "*",
          "Condition": {
              "StringEquals": {"aws:ResourceTag
/environment": "production"}
          }
      }
  ]
}
```

Fine-grained access control

To create a rule and consume events

- Role must have events:PutRule: action
- Cannot invoke targets with only this permission

Allow user to invoke targets

- Role must have events:PutTargets action

Fine-grained access control

Restrict types of events user can consume

- Use the events:source and events:detail-type condition keys

Restrict which resources can be added to a rule

- Use the events:TargetArn condition key

Restrict events originating from AWS CloudTrail

- Use the events:detail.userIdentity.principalId condition key

Amazon EventBridge and the AWS SAM



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Benefits of using Amazon EventBridge with AWS SAM



Simplicity

Develop smaller, simpler micro-services



Easier for teams

Each team can work on separate services



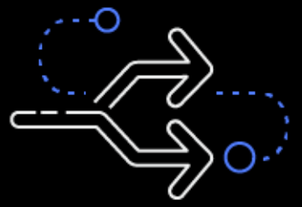
Faster to deploy

It's safer and easier to deploy



More extensible

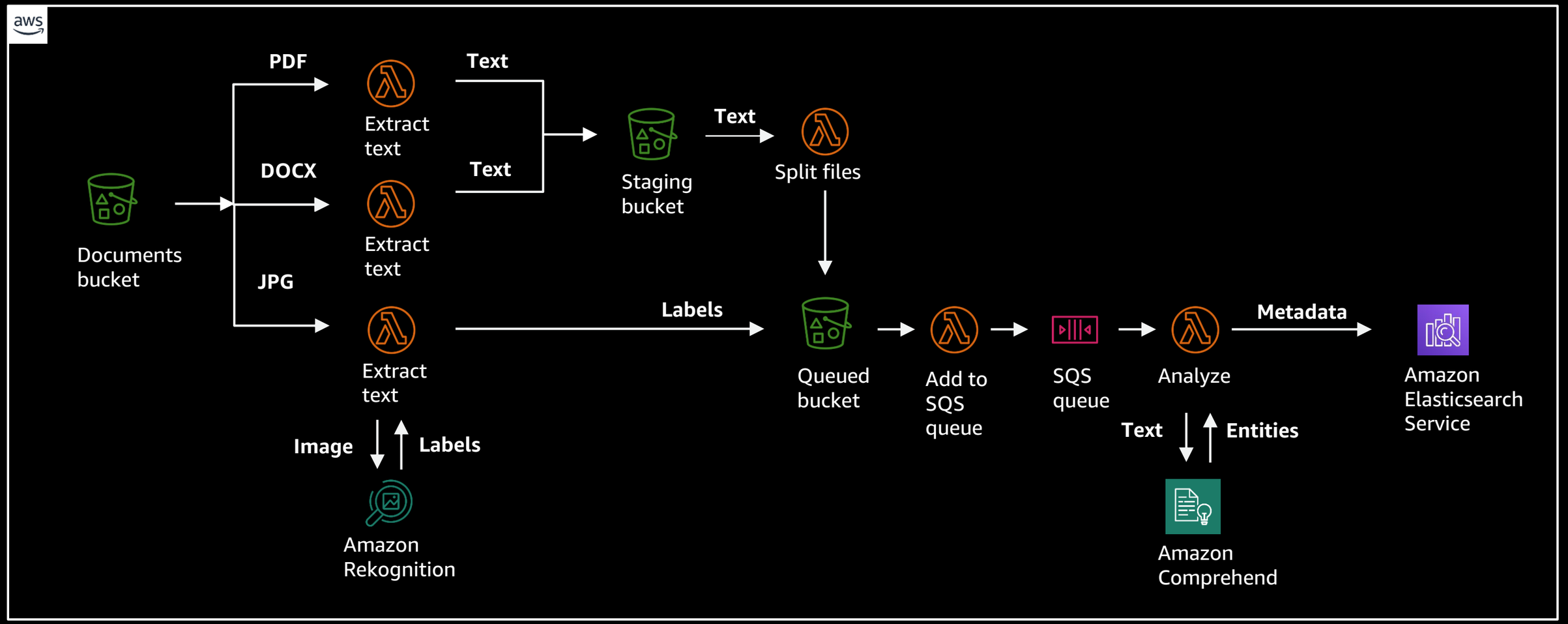
New features do not impact existing code



Easier to understand

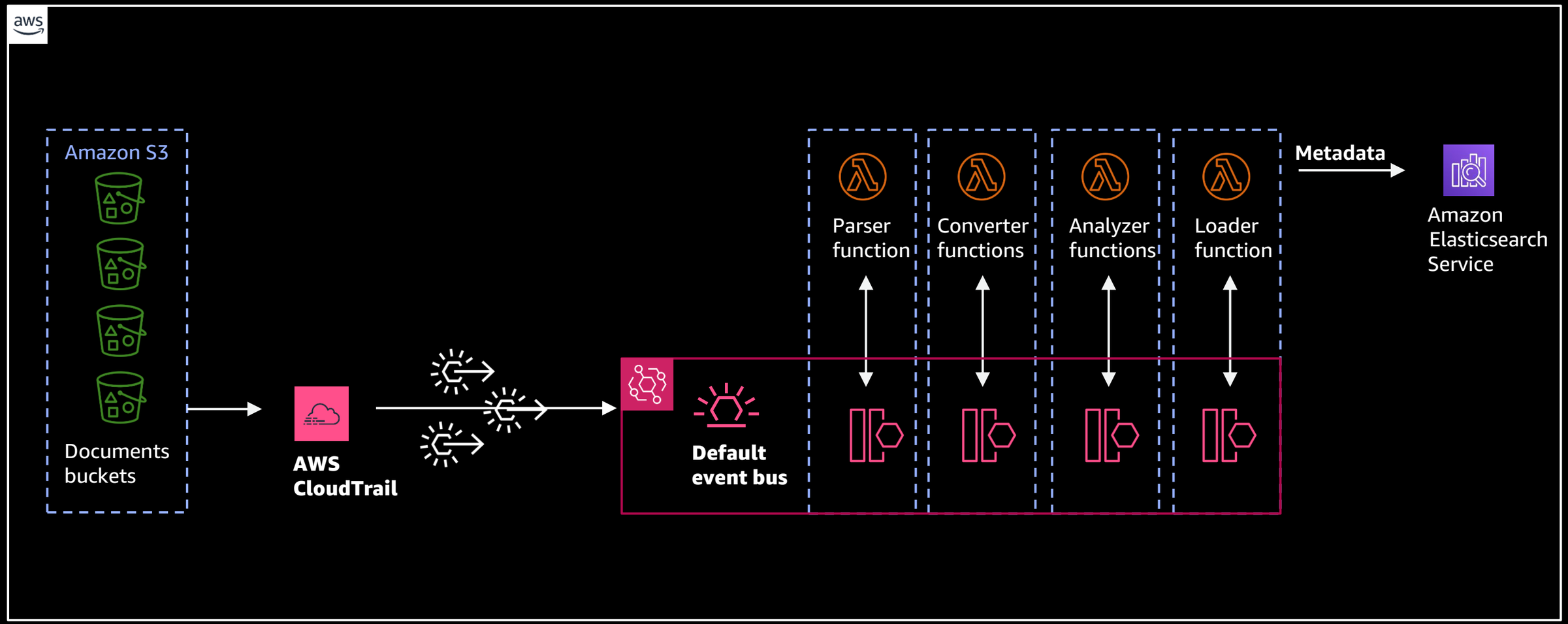
It's easier to see what each service is doing

Example: Document indexer application



Learn more about this example at
https://aws.amazon.com/blogs/compute/creating-a-searchable-enterprise-document-repository/

Example: Document Indexer application



Learn more about this example at
https://aws.amazon.com/blogs/compute/decoupling-larger-applications-with-amazon-eventbridge/

Using the Amazon EventBridge schema registry

Many sources of events

- AWS services
- SaaS partners
- Your custom applications

Schema registry keeps track of event structures (schema)

Auto-discovery of both AWS and custom schema

AWS Toolkit for Visual Studio code and JetBrains integrated development environments (IDEs)

For type-safe languages like Python and TypeScript

user

Delete all versions

Schema details

Edit description

Schema name	Last modified	Schema ARN
user	Aug 31, 2020, 04:55 PM EDT	arn:aws:schemas:us-east-2:763653534548:schema/myRegistry/user

Description

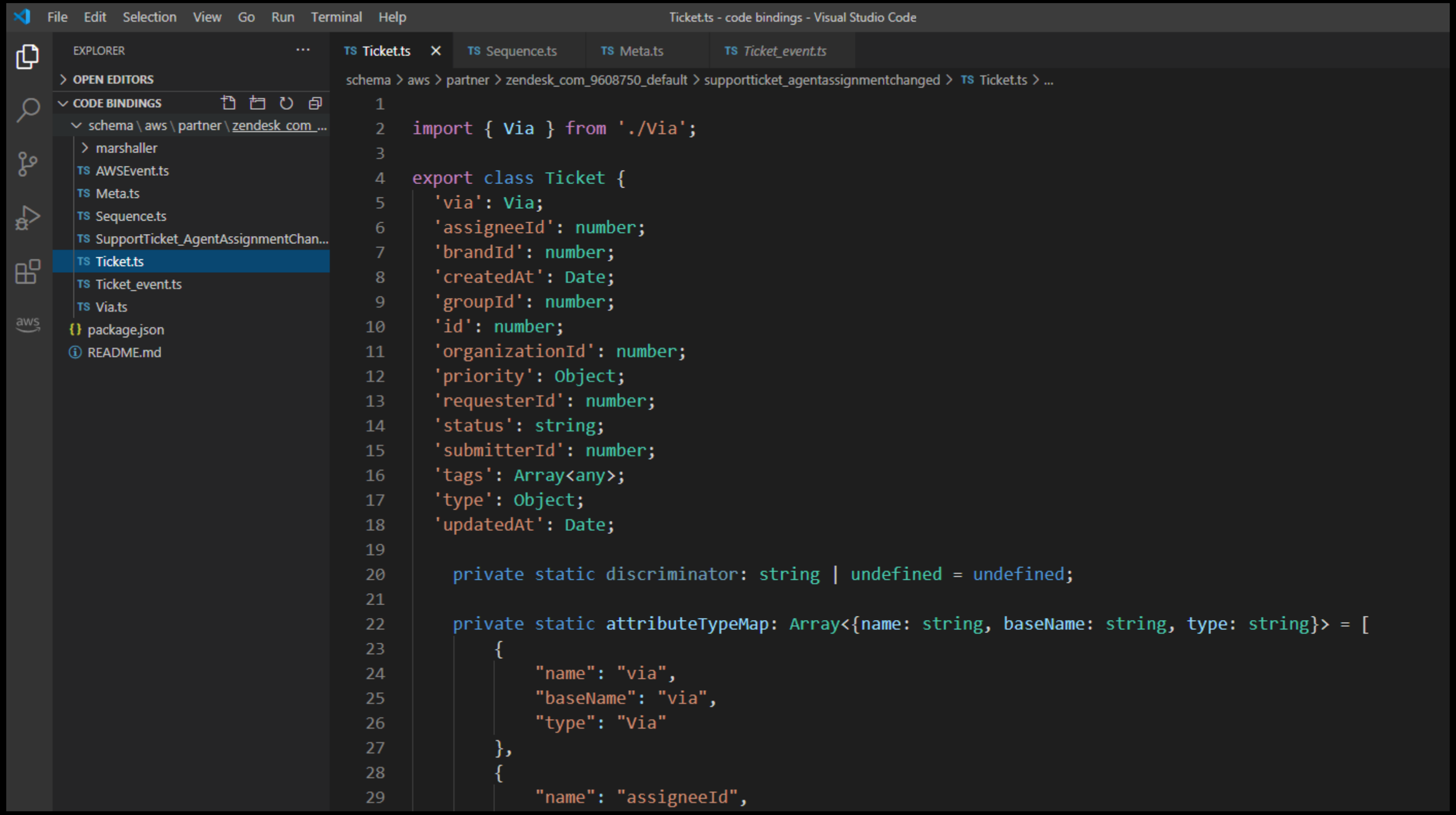
myApplication user profile

Schema registry	Number of versions	Schema type
myRegistry	1	OpenAPI 3.0

Version 1 Created on Dec 16, 2019, 10:50 AM EST

Version 1 | Save as a new version | Action | Download code bindings

```
{
  "openapi": "3.0.0",
  "info": {
    "version": "1.0.0",
    "title": "Event"
  },
  "paths": {},
  "components": {
    "schemas": {
      "Event": {
        "type": "object",
        "required": ["name", "id", "subscriptionRate"],
        "properties": {
          "id": {
            "type": "number"
```


File Edit Selection View Go Run Terminal Help
Ticket.ts - code bindings - Visual Studio Code
EXPLORER
OPEN EDITORS
CODE BINDINGS
schema \ aws \ partner \ zendesk_com ...
marshaller
AWSEvent.ts
Meta.ts
Sequence.ts
SupportTicket_AgentAssignmentChan...
Ticket.ts
Ticket_event.ts
Via.ts
package.json
README.md
Ticket.ts
Sequence.ts
Meta.ts
Ticket_event.ts
schema > aws > partner > zendesk_com_9608750_default > supportticket_agentassignmentchanged > Ticket.ts > ...
import { Via } from './Via';
export class Ticket {
'via': Via;
'assigneeId': number;
'brandId': number;
'createdAt': Date;
'groupId': number;
'id': number;
'organizationId': number;
'priority': Object;
'requesterId': number;
'status': string;
'submitterId': number;
'tags': Array<any>;
'type': Object;
'updatedAt': Date;
private static discriminator: string | undefined = undefined;
private static attributeTypeMap: Array<{name: string, baseName: string, type: string}> = [
{
"name": "via",
"baseName": "via",
"type": "Via"
},
{
"name": "assigneeId",

Conclusion



Decoupling

Amazon EventBridge can decouple your microservices



Scalability

Fully managed event scaling with minimal code



Extensibility

New event consumers are independent



Security

Granular access control to rules and events



Faster to deploy

Use AWS SAM to accelerate development

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.