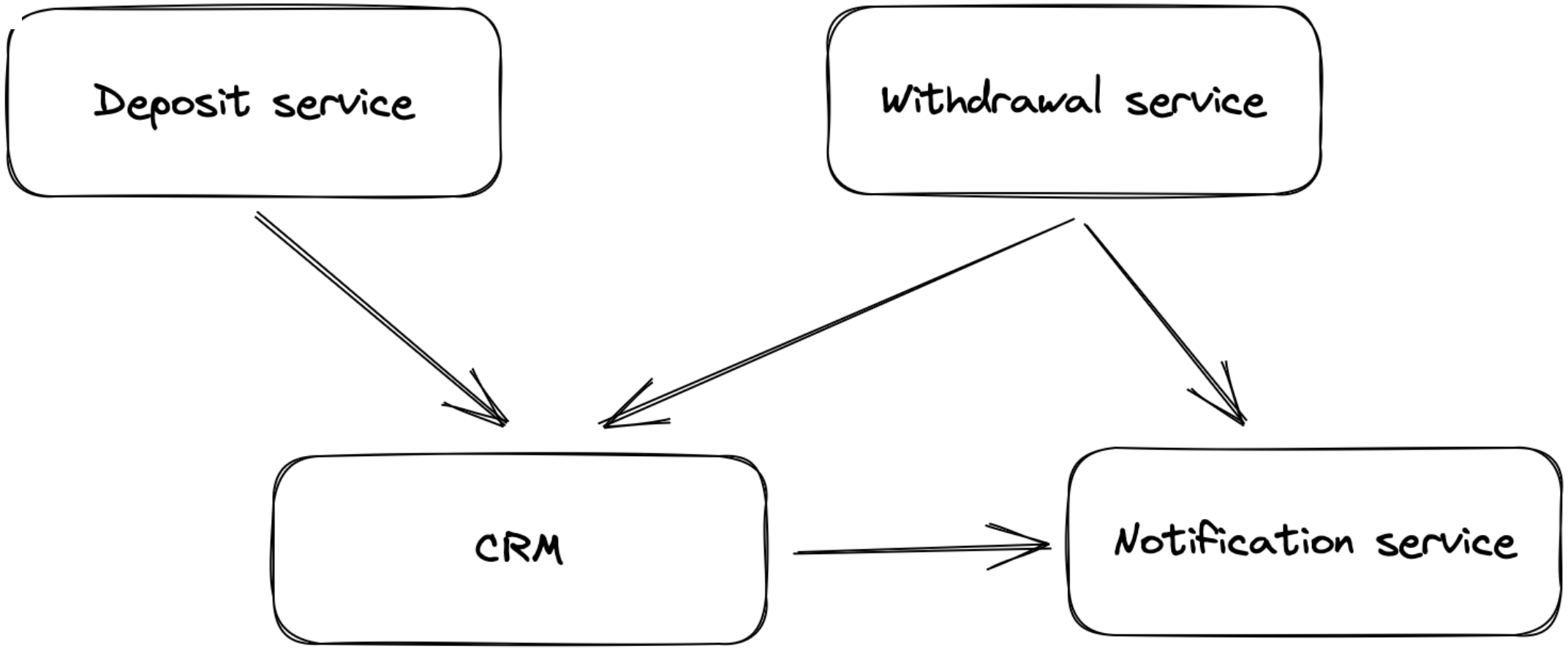
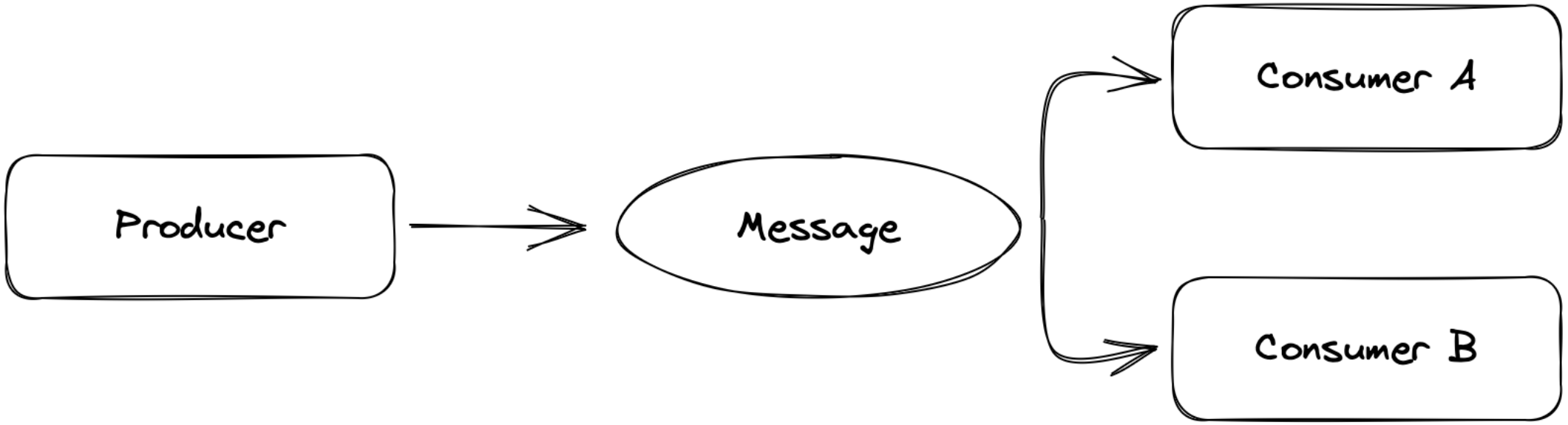
PURPLE
TECHNOLOGY

Decoupled architektúra s použitím AWS EventBridge

Pavol Jediný

Čo to je event-driven architektúra (EDA)


Deposit service
Withdrawal service
CRM
Notification service


Producer
Message
Consumer A
Consumer B

Výhody (EDA)

- Škálovanie systému

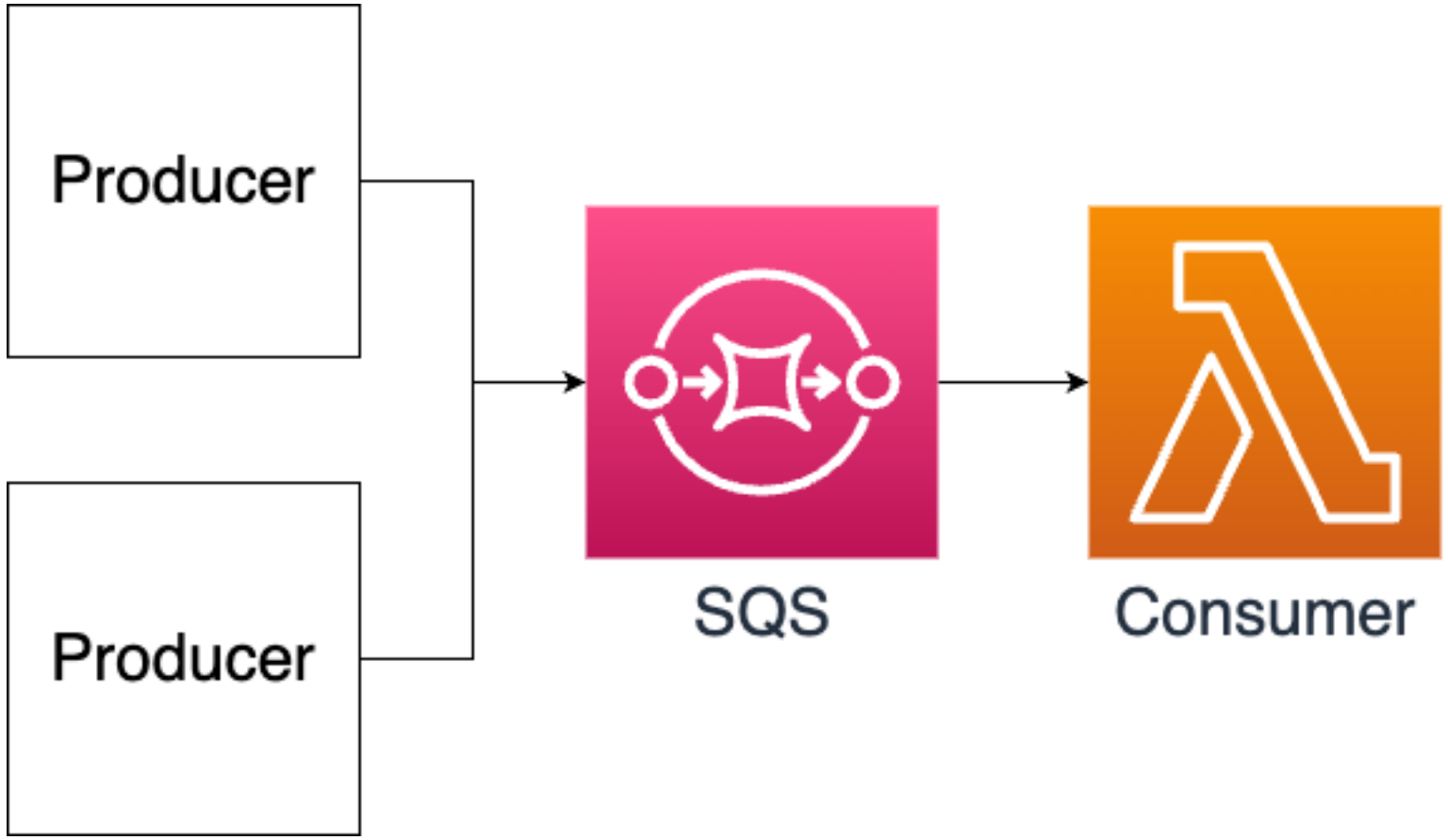
Výhody (EDA)

- Škálovanie systému
- Škálovanie teamu

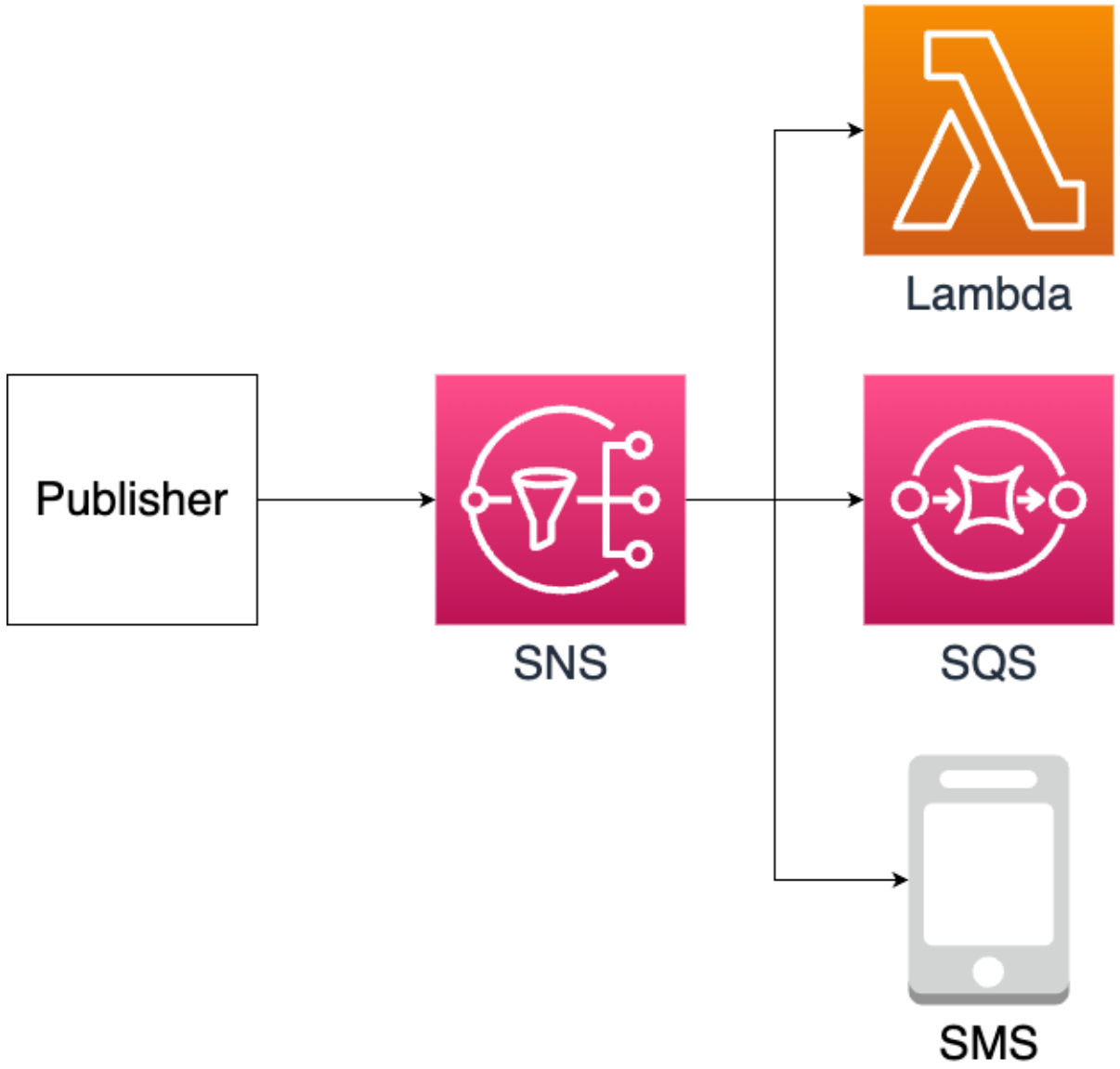
Výhody (EDA)

- Škálovanie systému
- Škálovanie teamu
- Odolnosť voči chybám

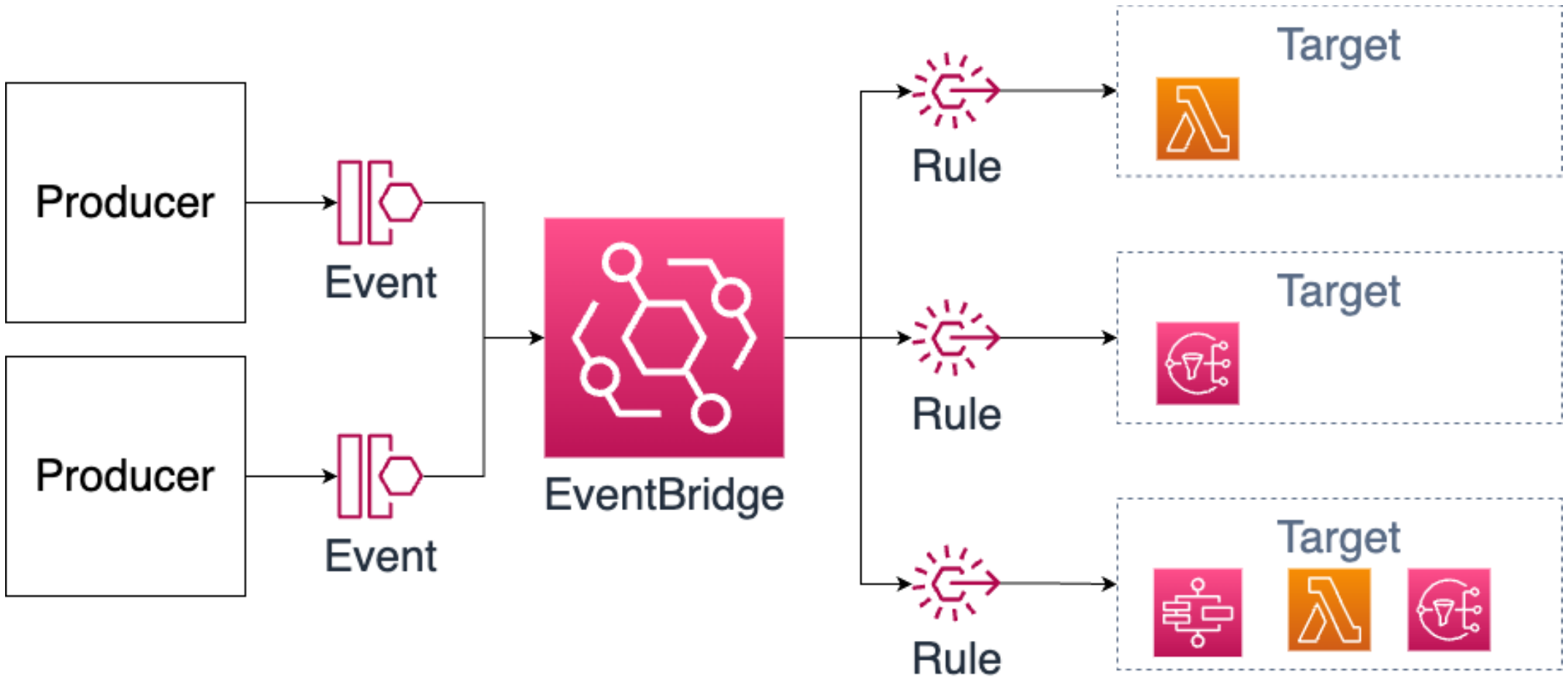
AWS služby - SQS



AWS služby - SNS



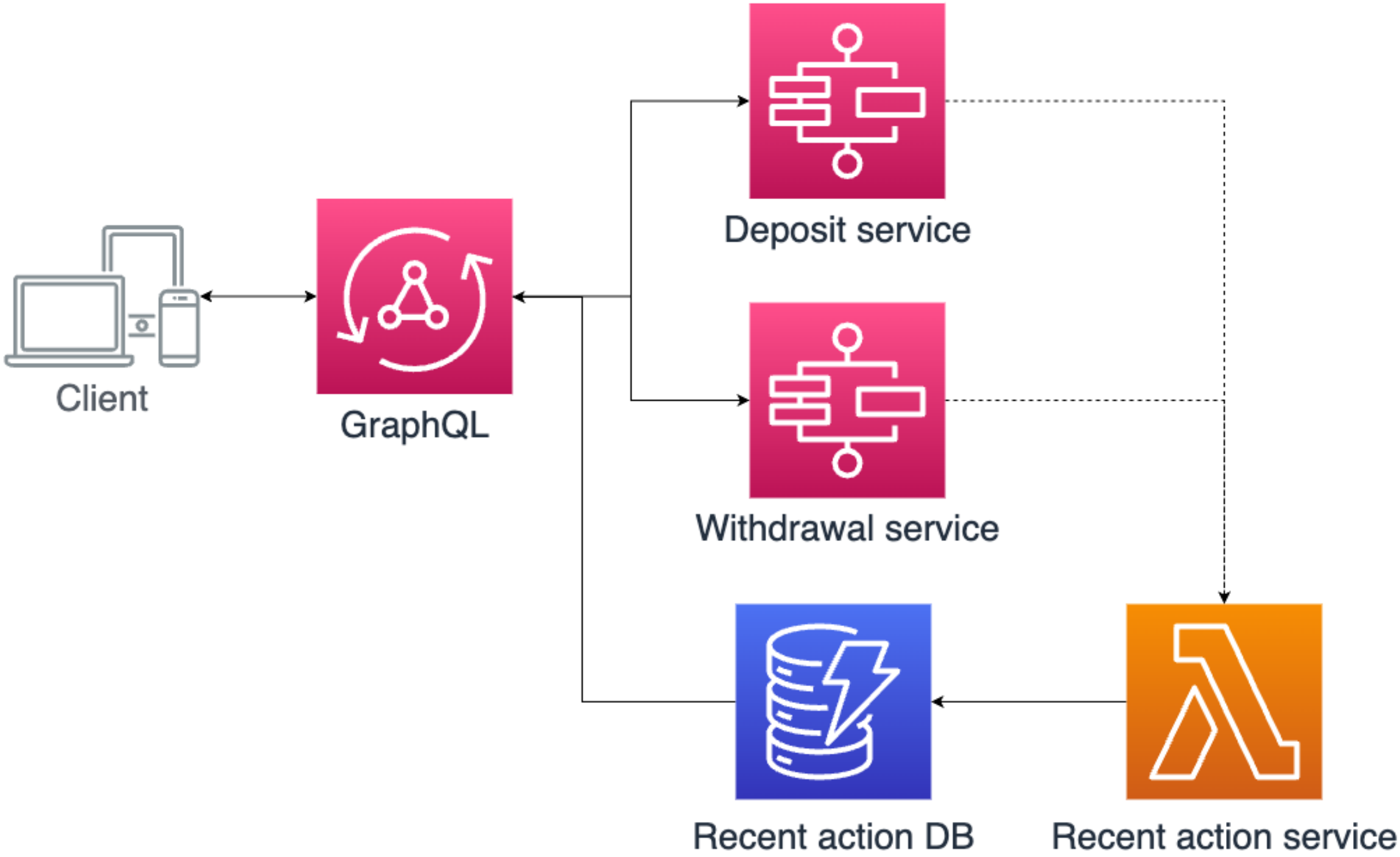
AWS služby - EventBridge



PURPLE TECHNOLOGY

Use case

Recent actions

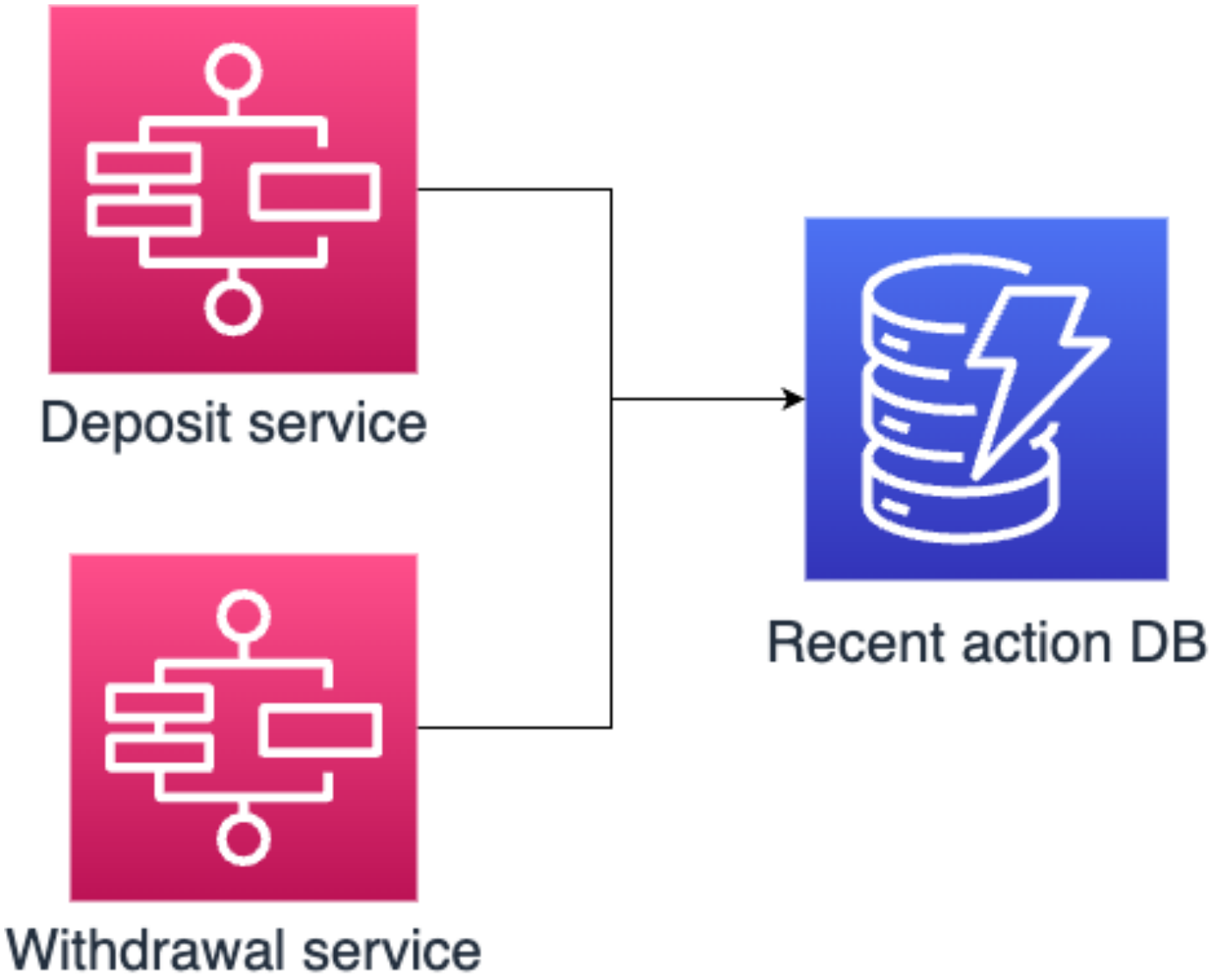

Client
GraphQL
Deposit service
Withdrawal service
Recent action DB
Recent action service

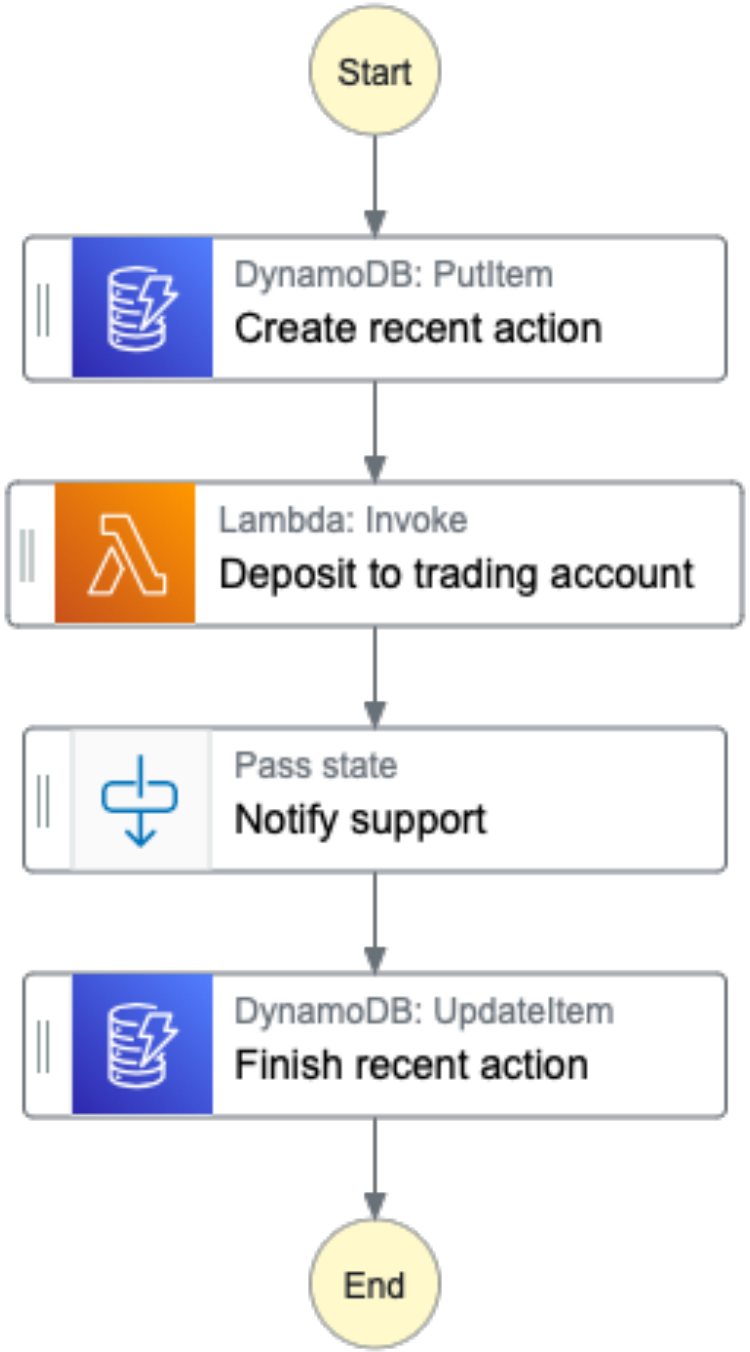
Recent Actions

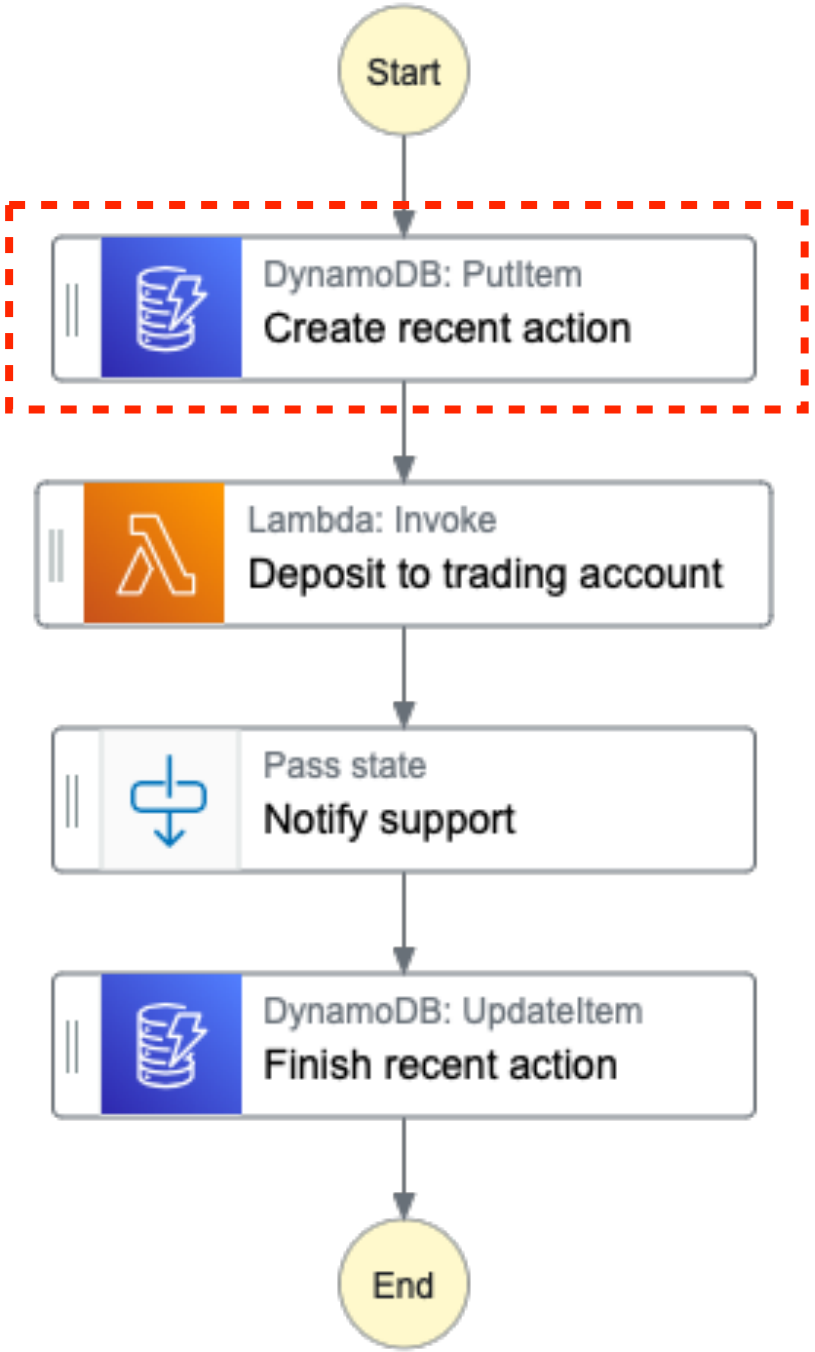
Show full history

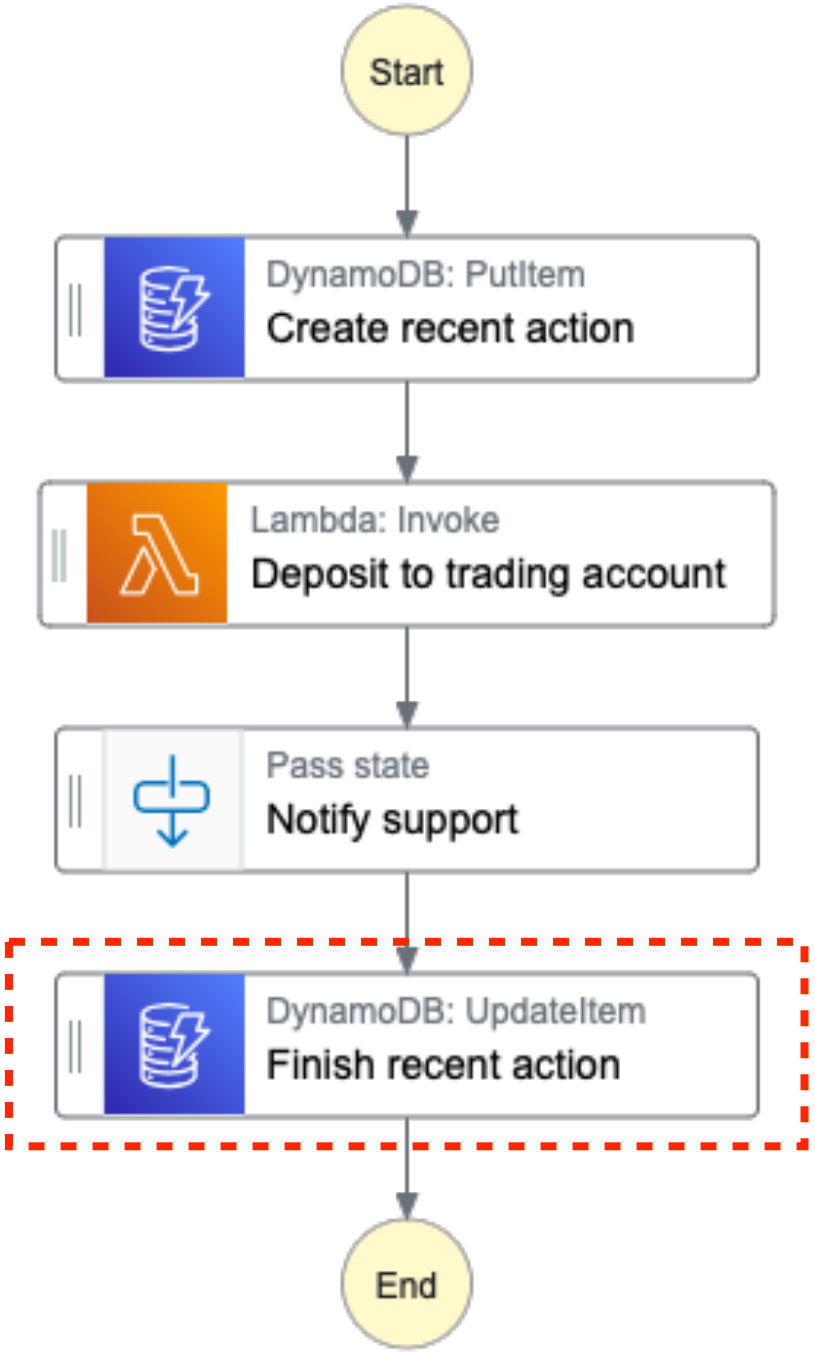
Completed	2022/12/31 07:14	New Live account		2000000026 | USD | TERA | LIVE
In progre...	2022/12/13 05:52	New Deposit - Sticpay	10 USD	1259982 | USD | STANDARD | LIVE
Completed	2022/12/12 23:36	New payment method details added - Sticpay		
Completed	2022/12/12 23:35	New payment method details added - Skrill		
Completed	2022/12/12 23:34	New Deposit - Sticpay	20 USD	1259982 | USD | STANDARD | LIVE

Návrh #1

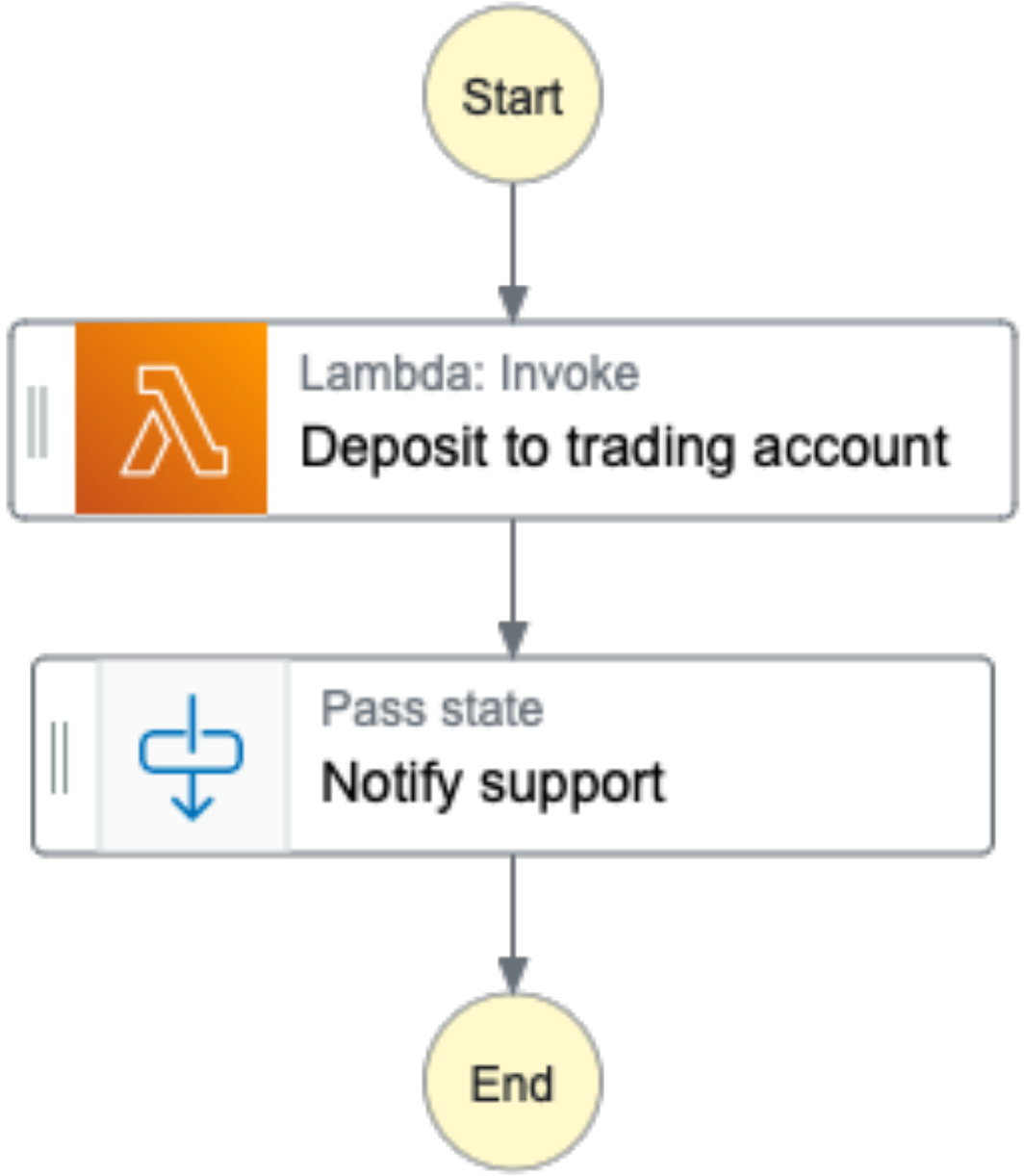

Deposit service
Withdrawal service
Recent action DB

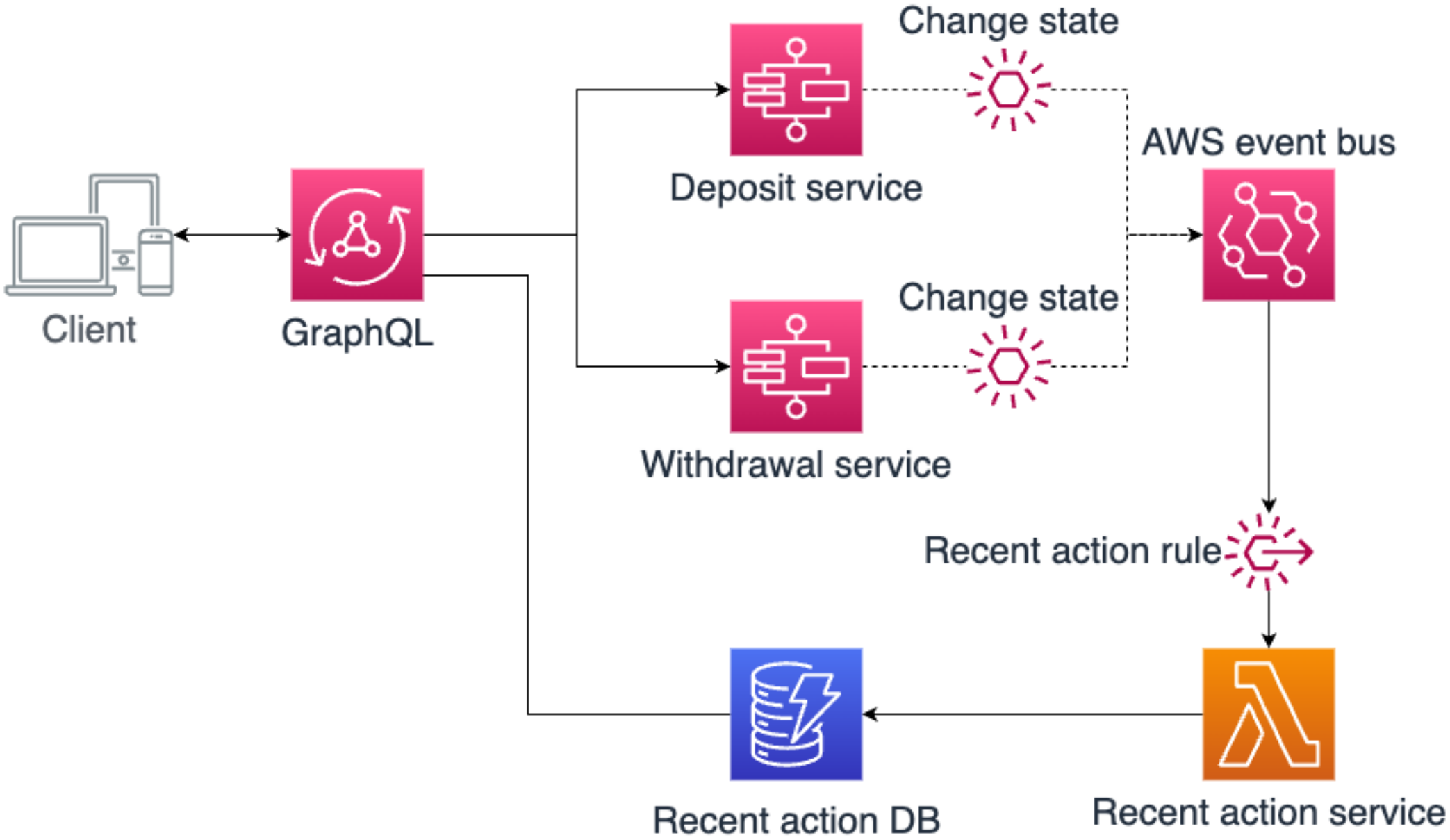

Start
DynamoDB: PutItem
Create recent action
Lambda: Invoke
Deposit to trading account
Pass state
Notify support
DynamoDB: UpdateItem
Finish recent action
End


Start
DynamoDB: PutItem
Create recent action
Lambda: Invoke
Deposit to trading account
Pass state
Notify support
DynamoDB: UpdateItem
Finish recent action
End

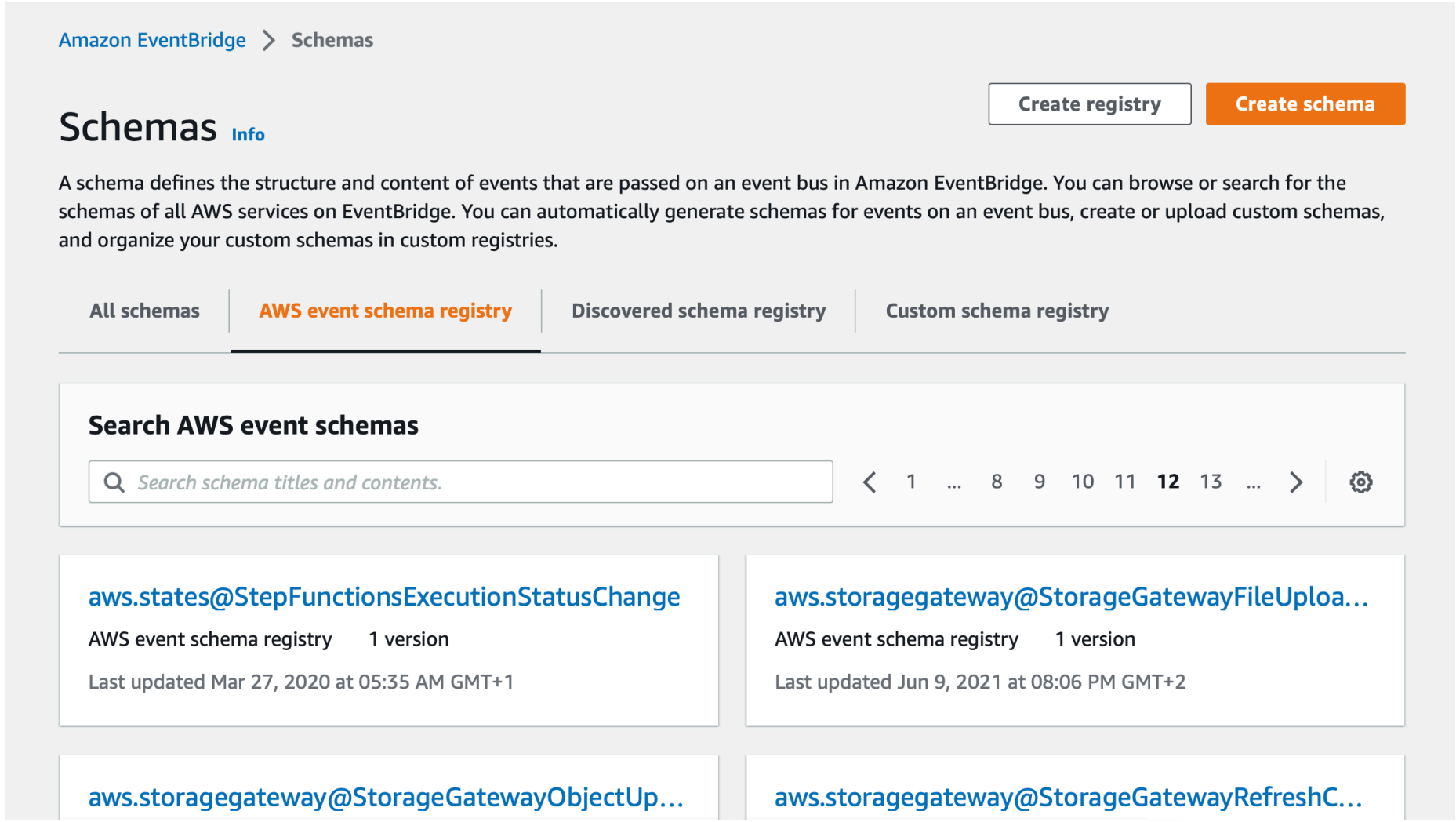

Start
DynamoDB: PutItem
Create recent action
Lambda: Invoke
Deposit to trading account
Pass state
Notify support
DynamoDB: UpdateItem
Finish recent action
End

Návrh #2


Start
Lambda: Invoke
Deposit to trading account
Pass state
Notify support
End

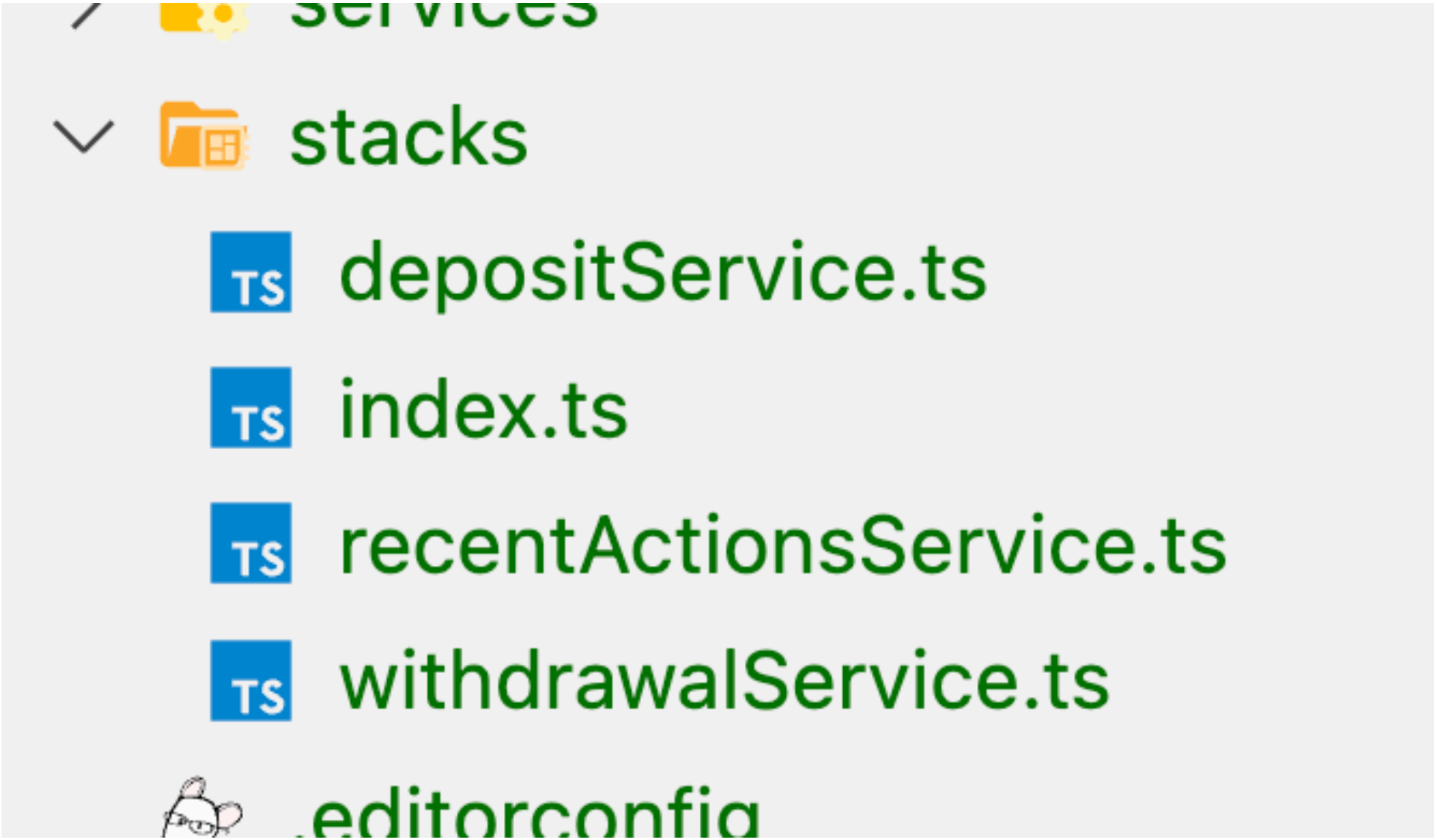

Change state
AWS event bus
Deposit service
Client
GraphQL
Change state
Withdrawal service
Recent action rule
Recent action DB
Recent action service

Zoznam AWS schém



SST & CDK

- CDK (Cloud Development Kit)
  - AWS framework
  - Definovanie infraštruktúry v kóde (IaC)
  - Deploy prostredníctvon AWS CloudFormation
- SST
  - CDK extension
  - Lepšie DX
  - https://sst.dev/


services
stacks
depositService.ts
index.ts
recentActionsService.ts
withdrawalService.ts
.editorconfig

```ts
export const depositServiceStack: FunctionalStack<
  DepositServiceStackOutput
> = ({ stack }) => {
  const app = stack.node.root as sst.App
  const { stage, name } = app

  const depositToTradingAccountStep = new tasks.LambdaInvoke(
    stack,
    'Deposit to trading account',
    {
      lambdaFunction: new sst.Function(stack, 'depositToTradingAccount', {
        handler: `services/deposit/src/depositToTradingAccount/index.handler`
      }),
      payloadResponseOnly: true
    }
  )
  const notifySupportStep = new Pass(stack, 'Notify support', {
    resultPath: JsonPath.DISCARD
  })

  const depositStateMachine = new sfn.StateMachine(stack, 'deposit', {
    stateMachineName: `${stage}-${name}-deposit`,
    definition: depositToTradingAccountStep.next(notifySupportStep)
  })

  return {
    depositStateMachine
  }
}
```

```
const depositToTradingAccountStep = new tasks.LambdaInvoke(
  stack,
  'Deposit to trading account',
  {
    lambdaFunction: new sst.Function(stack, 'depositToTradingAccount', {
      handler: `services/deposit/src/depositToTradingAccount/index.handler`
    }),
    payloadResponseOnly: true
  }
)
const notifySupportStep = new Pass(stack, 'Notify support', {
  resultPath: JsonPath.DISCARD
})

const depositStateMachine = new sfn.StateMachine(stack, 'deposit',
  stateMachineName: `${stage}-${name}-deposit`,
  definition: depositToTradingAccountStep.next(notifySupportStep)
})
```

```
const depositToTradingAccountStep = new tasks.LambdaInvoke(
  stack,
  'Deposit to trading account',
  {
    lambdaFunction: new sst.Function(stack, 'depositToTradingAccount', {
      handler: `services/deposit/src/depositToTradingAccount/index.handler`
    }),
    payloadResponseOnly: true
  }
)
const notifySupportStep = new Pass(stack, 'Notify support', {
  resultPath: JsonPath.DISCARD
})

const depositStateMachine = new sfn.StateMachine(stack, 'deposit', {
  stateMachineName: `${stage}-${name}-deposit`,
  definition: depositToTradingAccountStep.next(notifySupportStep)
})
```

A toto je cieľom mojej prezentácie

```
export function recentActionsStack({
  stack
}: StackContext): RecentActionsStackOutput {
  const executionsTable = new Table(stack, 'executions', {
    fields: {
      executionId: 'string',
      clientId: 'string'
    },
    primaryIndex: { partitionKey: 'executionId' }
  })

  const depositService = use(depositServiceStack)
  const withdrawalService = use(withdrawalServiceStack)

  const stateMachineHandler = new Function(stack, 'Function', {
    handler: 'services/recentActions/src/stateMachineHandler/index.handler',
    environment: {
      EXECUTIONS_TABLE: executionsTable.tableName
    }
  })
  stateMachineHandler.bind([executionsTable])

  new EventBus(stack, 'RecentActionsBus', {
    cdk: {
      eventBus: events.EventBus.fromEventBusName(
        stack,
        'ImportedBus',
        'default'
      )
    },
    rules: {
      stepFunctionsStatusChange: {
        pattern: {
          source: ['aws.states'],
          detailType: ['Step Functions Execution Status Change'],
          detail: {
            stateMachineArn: [
              depositService.depositStateMachine.stateMachineArn,
              withdrawalService.withdrawalStateMachine.stateMachir
            ]
          }
        },
        targets: {
          stateMachineHandler
        }
      }
    }
  })

  return {}
}
```

```
export function recentActionsStack({
  stack
}: StackContext): RecentActionsStackOutput {
  const executionsTable = new Table(stack, 'executions', {
    fields: {
      executionId: 'string',
      clientId: 'string'
    },
    primaryIndex: { partitionKey: 'executionId' }
  })

  const depositService = use(depositServiceStack)
  const withdrawalService = use(withdrawalServiceStack)
```

```
export function recentActionsStack({
  stack
}: StackContext): RecentActionsStackOutput {
  const executionsTable = new Table(stack, 'executions', {
    fields: {
      executionId: 'string',
      clientId: 'string'
    },
    primaryIndex: { partitionKey: 'executionId' }
  })

  const depositService = use(depositServiceStack)
  const withdrawalService = use(withdrawalServiceStack)
```

```
const depositService = use(depositServiceStack)
const withdrawalService = use(withdrawalServiceStack)

const stateMachineHandler = new Function(stack, 'Function', {
  handler: 'services/recentActions/src/stateMachineHandler/index.handler',
  environment: {
    EXECUTIONS_TABLE: executionsTable.tableName
  }
})
stateMachineHandler.bind([executionsTable])

new EventBus(stack, 'RecentActionsBus', {
  cdk: {
```

```
new EventBus(stack, 'RecentActionsBus', {
  cdk: {
    eventBus: events.EventBus.fromEventBusName(
      stack,
      'ImportedBus',
      'default'
    )
  },
  rules: {
    stepFunctionsStatusChange: {
      pattern: {
        source: ['aws.states'],
        detailType: ['Step Functions Execution Status Change'],
        detail: {
          stateMachineArn: [
            depositService.depositStateMachine.stateMachineArn,
            withdrawalService.withdrawalStateMachine.stateMachineArn
          ]
        }
      },
      targets: {
        stateMachineHandler
      }
    }
  }
})
```

```
new EventBus(stack, 'RecentActionsBus', {
  cdk: {
    eventBus: events.EventBus.fromEventBusName(
      stack,
      'ImportedBus',
      'default'
    )
  },
  rules: {
    stepFunctionsStatusChange: {
      pattern: {
        source: ['aws.states'],
        detailType: ['Step Functions Execution Status Change'],
        detail: {
          stateMachineArn: [
            depositService.depositStateMachine.stateMachineArn,
            withdrawalService.withdrawalStateMachine.stateMachineArn
          ]
        }
      },
      targets: {
        stateMachineHandler
      }
    }
  }
})
```

```
new EventBus(stack, 'RecentActionsBus', {
  cdk: {
    eventBus: events.EventBus.fromEventBusName(
      stack,
      'ImportedBus',
      'default'
    )
  },
  rules: {
    stepFunctionsStatusChange: {
      pattern: {
        source: ['aws.states'],
        detailType: ['Step Functions Execution Status Change'],
        detail: {
          stateMachineArn: [
            depositService.depositStateMachine.stateMachineArn,
            withdrawalService.withdrawalStateMachine.stateMachineArn
          ]
        }
      },
      targets: {
        stateMachineHandler
      }
    }
  }
})
```

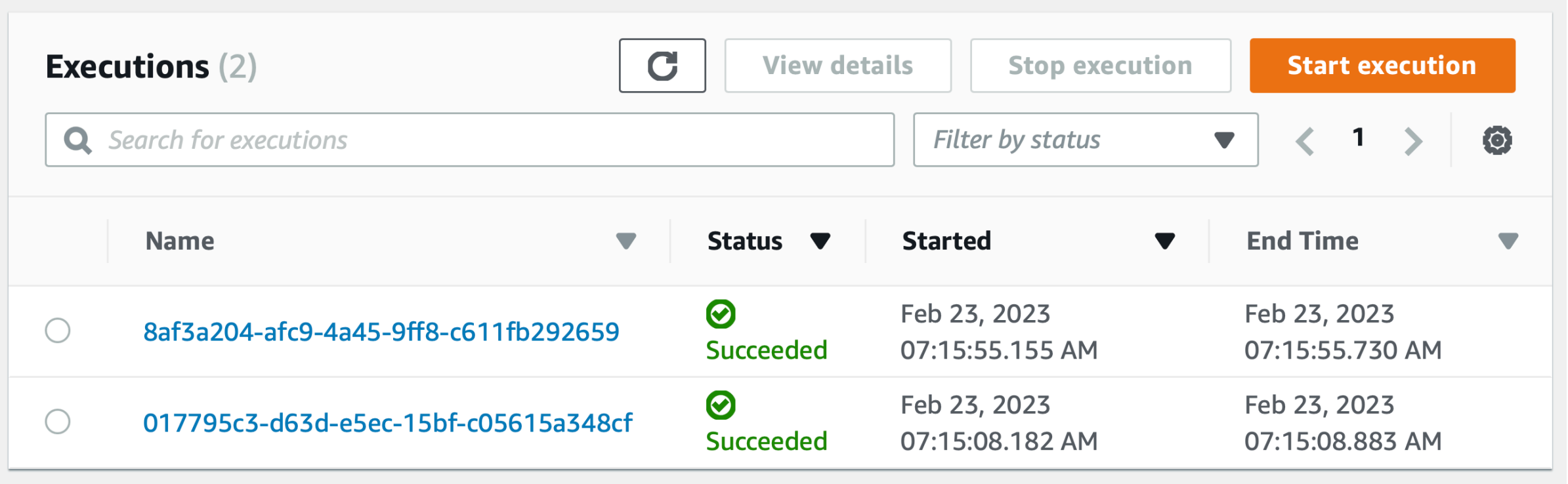
```
new EventBus(stack, 'RecentActionsBus', {
  cdk: {
    eventBus: events.EventBus.fromEventBusName(
      stack,
      'ImportedBus',
      'default'
    )
  },
  rules: {
    stepFunctionsStatusChange: {
      pattern: {
        source: ['aws.states'],
        detailType: ['Step Functions Execution Status Change'],
        detail: {
          stateMachineArn: [
            depositService.depositStateMachine.stateMachineArn,
            withdrawalService.withdrawalStateMachine.stateMachineArn
          ]
        }
      },
      targets: {
        stateMachineHandler
      }
    }
  }
})
```

```
{
  "version": "0",
  "id": "149ca895-6394-de0a-ccb5-33be1c57b2c5",
  "detail-type": "Step Functions Execution Status Change",
  "source": "aws.states",
  "account": "xxx",
  "time": "2023-02-21T06:11:21Z",
  "region": "eu-central-1",
  "resources": [
    "arn:aws:states:eu-central-1:xxx:execution:fMeetup-meetup3-deposit:1c55359b-d5a9-4a13-ba42-b15449ab2746"
  ],
  "detail": {
    "executionArn": "arn:aws:states:eu-central-1:xxx:execution:fMeetup-meetup3-deposit:1c55359b-d5a9-4a13-ba42-b15449ab2746",
    "stateMachineArn": "arn:aws:states:eu-central-1:xxx:stateMachine:fMeetup-meetup3-deposit",
    "name": "1c55359b-d5a9-4a13-ba42-b15449ab2746",
    "status": "SUCCEEDED",
    "startDate": 1676959878632,
    "stopDate": 1676959880695,
    "input": "{\"amount\": 50,\"accountNumber\": \"179\",\"clientId\": \"90876\"}",
    "output": "{\"amount\":50,\"accountNumber\":\"179\",\"clientId\":\"90876\",\"ticketNumber\":\"3b552eac-c5ec-4179-af80-9a72617fe618\"}",
    "inputDetails": {
      "included": true
    },
    "outputDetails": {
      "included": true
    },
    "error": null,
    "cause": null
  }
}
```

```ts
export const handler: Handler<Event, void> = async (event): Promise<void> => {
  const recentAction = parseRecentAction(event)

  const dynamodbClient = new DynamoDBClient({})
  await dynamodbClient.send(
    new PutItemCommand({
      TableName: process.env.EXECUTIONS_TABLE,
      Item: marshall(recentAction, { removeUndefinedValues: true })
    })
  )
}
```

Deposit state machine



Recent actions DB

Items returned (2)

Actions ▼ | Create item

1

executionId	clientId	input	output	status
arn:aws:states:eu-cen…	987654	{ "accountN…	{ "accountN…	SUCCEEDED
arn:aws:states:eu-cen…	123456	{ "accountN…	{ "accountN…	SUCCEEDED

Nevýhody EDA

- Debugging
- Onboarding nových developerov
- Dokumentácia systému
- Chýba vstupná validácia eventov

Ďakujem za pozornosť

Demo repozitár

Blog