AWS Compute Blog

Introducing cross-account targets for Amazon EventBridge Event Buses

by Chris McPeek | on 21 JAN 2025 | in Amazon EventBridge, Amazon Simple Notification Service (SNS), Amazon Simple Queue Service (SQS), AWS Identity and Access Management (IAM), AWS Lambda, AWS Serverless Application Model, Serverless | Permalink | Share

This post is written by Anton Aleksandrov, Principal Solutions Architect, Serverless and Alexander Vladimirov, Senior Solutions Architect, Serverless

Today, Amazon EventBridge is announcing support for cross-account targets for Event Buses. This new capability allows you to send events directly to targets, such as Amazon Simple Queue Service (Amazon SQS), AWS Lambda, and Amazon Simple Notification Service (Amazon SNS), located in other accounts.

Previously, EventBridge supported cross-account event delivery from an event bus in one account to an event bus in another account. This launch extends that capability and allows you to configure the source event bus to deliver events directly to all EventBridge supported targets in other accounts, not just event buses. This removes the need for an additional event bus in the target account.

Overview

Event-driven architectures built with EventBridge allow you to create solutions spanning many company departments and business domains, while remaining asynchronous and loosely coupled. As solutions grow, you may need to send events across account boundaries.

For example, you may have a set of event buses hosted in multiple accounts that are dispatching security-related events to an Amazon SQS queue hosted in a centralized account for further asynchronous processing and analysis.

Previously, EventBridge rules allowed you to define targets in the same account. The only target type that supported cross-account event delivery was another event bus. If you wanted to send events across account boundaries, you had to create event buses in both source and target accounts. After, you would configure a rule on the source event bus to send events to the target bus, and another rule on the target event bus to deliver the event to a desired target in the target account. Alternatively, a Lambda function or SNS topic could be used as a bridging mechanism to send events across accounts.

The following diagram illustrates what an architecture of cross-account event delivery looked like before the new capability. A “bridging” component, like another event bus, SNS topic, or Lambda function, was required to send events from one account to another.

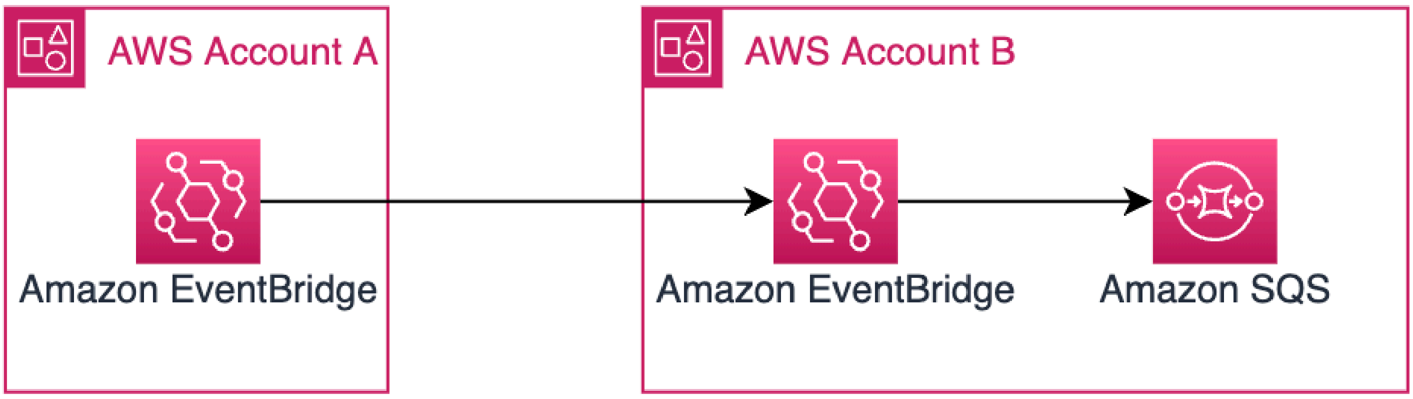


Figure 1: Delivering cross-account events from source bus to target bus

With this new EventBridge feature, you can deliver events from the source event bus to the desired targets in different accounts directly. This simplifies the architecture and persmission model. It also reduces latency in your event-driven solutions by having fewer components processing events along the path from source to targets.

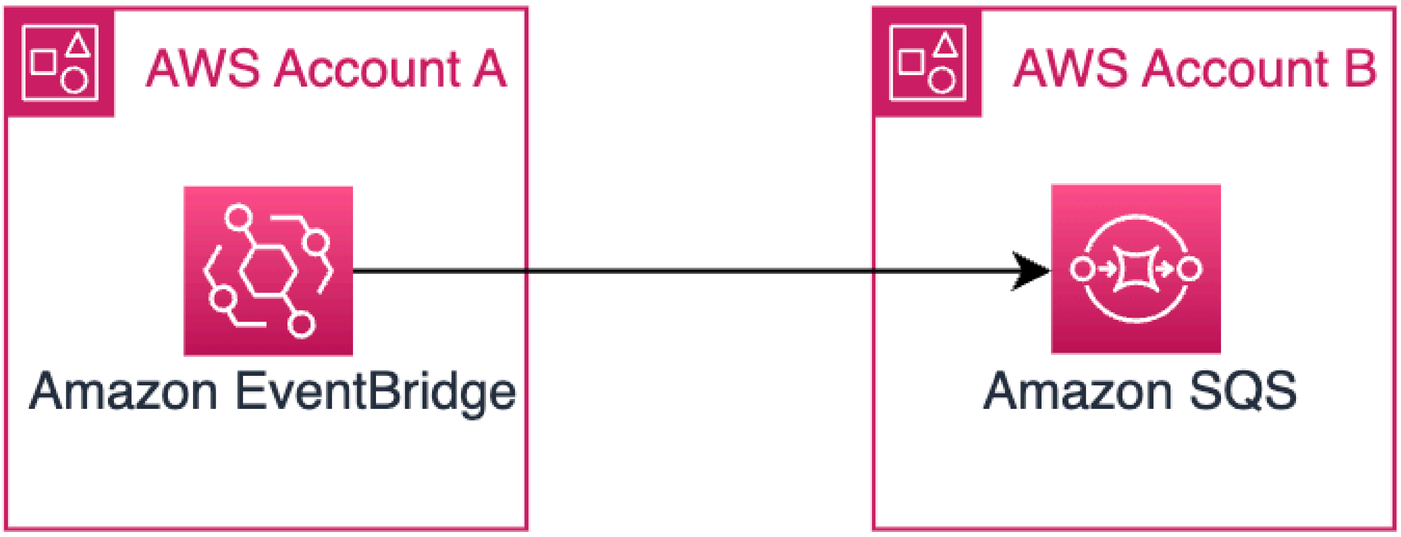


Figure 2: Delivering cross-account events to target directly

Configuring EventBridge delivery rule targets for cross-account event delivery

Enabling cross-account event delivery should be done with security in mind. You must establish mutual trust between the source and the target. Source event bus rules must have an AWS Identity and Access Management (IAM) role that allows them to send events to specific targets. This is achieved by attaching an execution role to the delivery rule targets.

Event delivery targets hosted in different accounts must have a resource access policy attached that explicitly allows receiving events from the execution role used in the source account. Due to this requirement, you can enable cross-account event delivery only for targets that support resource access policies, such as Amazon SQS queues, Amazon SNS topics, and AWS Lambda functions.

Having both an IAM role in the source account and resource policy in the target account allows you to have fine-grained control over which principals are allowed to use the PutEvents action and under which conditions. You can define service control policies (SCPs) to set organizational boundaries determining who can send and receive events in your organization.

As illustrated in the following diagram, consider Team A owns the source account (Account A). Team A is responsible for setting up the source event bus, its execution role, rules, and targets. Teams B and C own the target accounts (Account B and Account C, accordingly). Both teams manage their respective target accounts. For example, creating delivery targets, such as SQS queues, and granting permissions to accept events from the event bus in the source account. This approach enables Team A to manage the centralized source event bus for other teams, and Teams B and C to control who can send events to their targets. It provides high degree of overall control and governance.

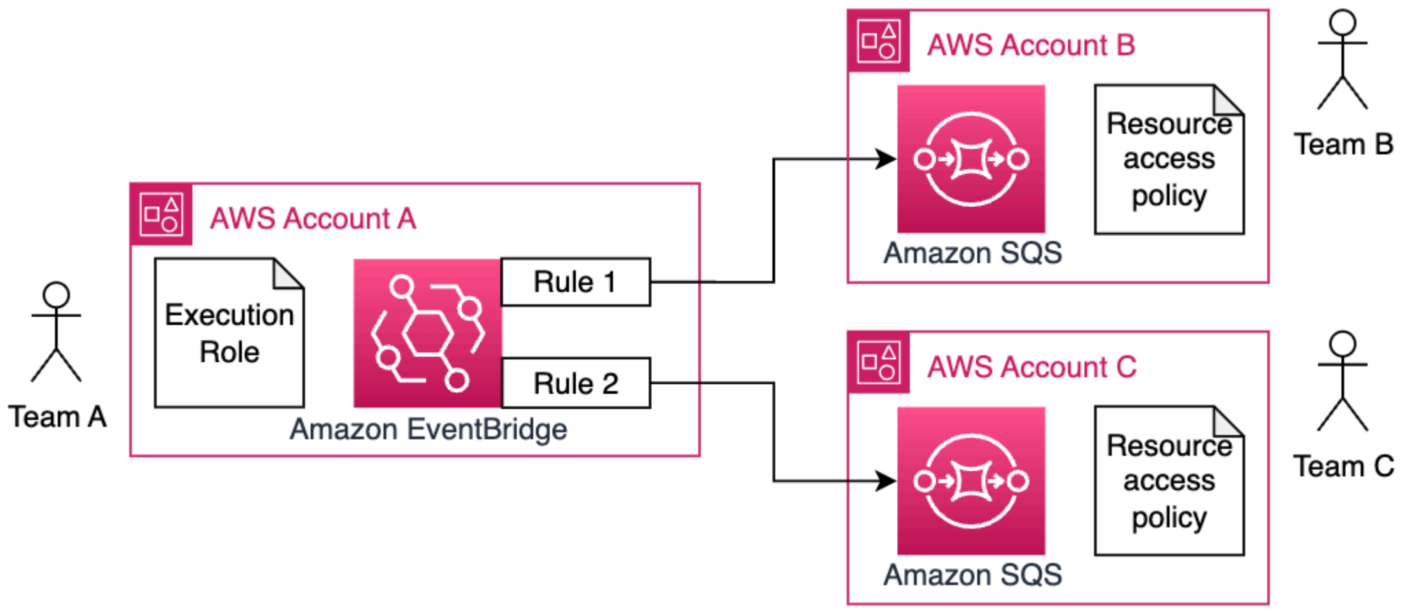


Figure 3: A cross-team collaboration sending events from source account to target account targets

The following example describes setting up cross-account event delivery to an SQS queue. You can apply the same technique to other target types as well, such as Lambda functions or SNS topics.

See the following diagram for a conceptual architectural layout and resource creation order.

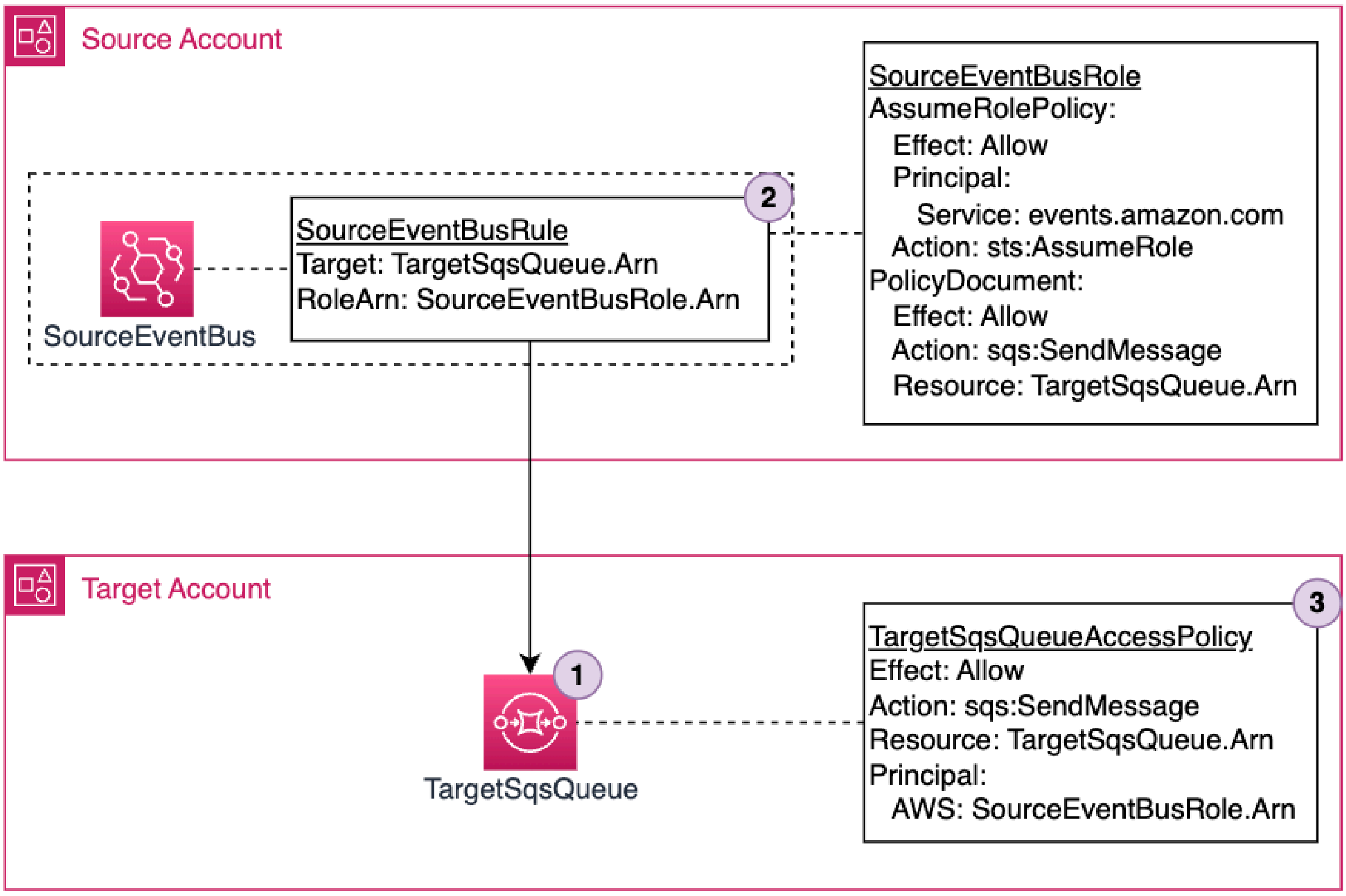


Figure 4: Permissions required for cross-account event delivery

Assuming the source event bus already exists, there are three general steps in setting up cross-account event delivery:

1. Target account – create the delivery target, such as an SQS queue.
2. Source account – configure a rule for cross-account event delivery. Set the target SQS queue ARN as rule target, and attach an execution role with permissions to send messages to the target SQS Queue.
3. Target account – apply a resource policy to the target SQS queue allowing the source event bus execution role to send events.

Showing cross-account delivery in action

Follow the instructions in this GitHub repository for provisioning the sample in your AWS accounts using AWS Serverless Application Model (AWS SAM). An event bus rule in the source account sends events directly to an SQS queue, a Lambda function, and an SNS topic in a target account. You must have two accounts for the sample to work.

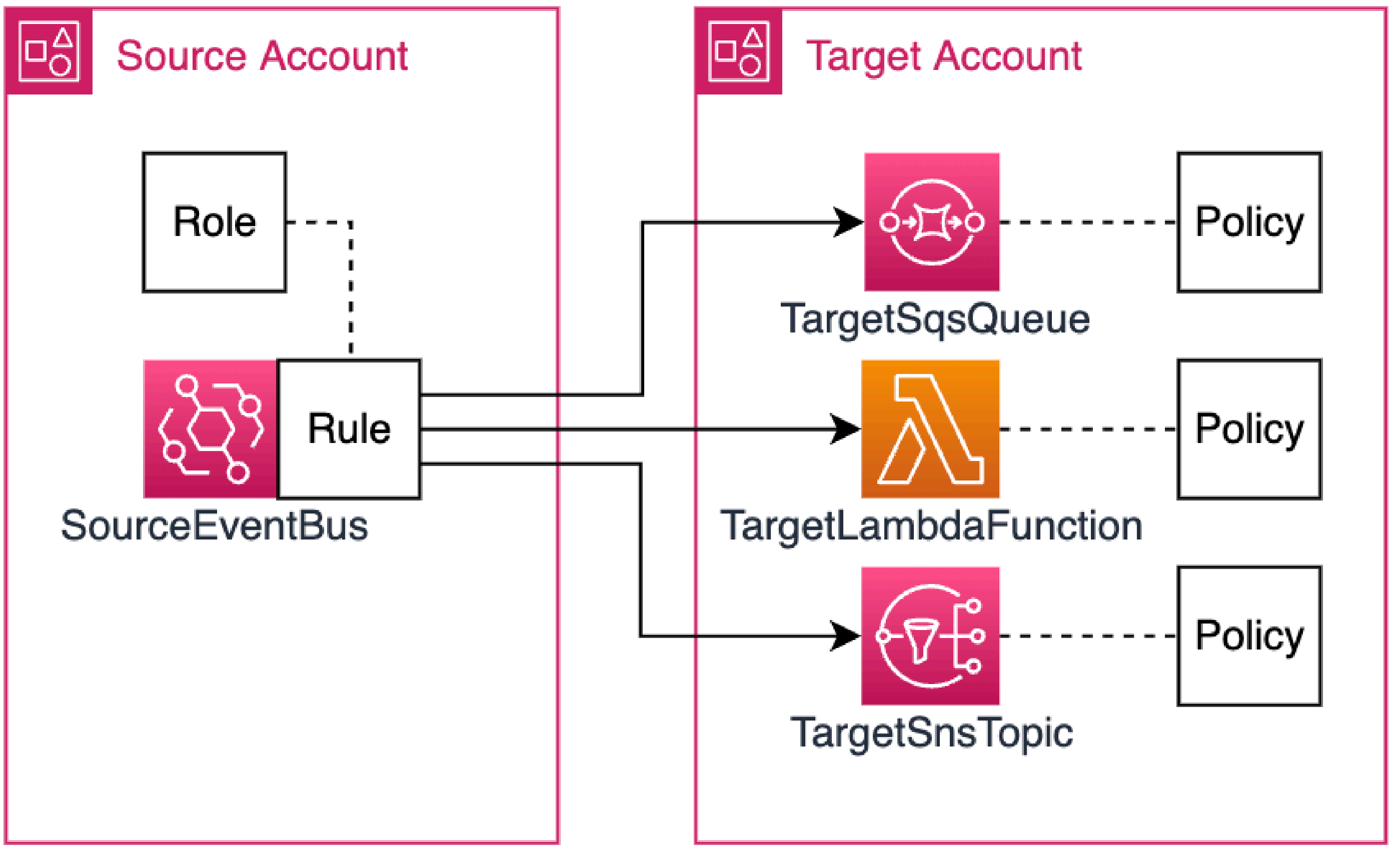
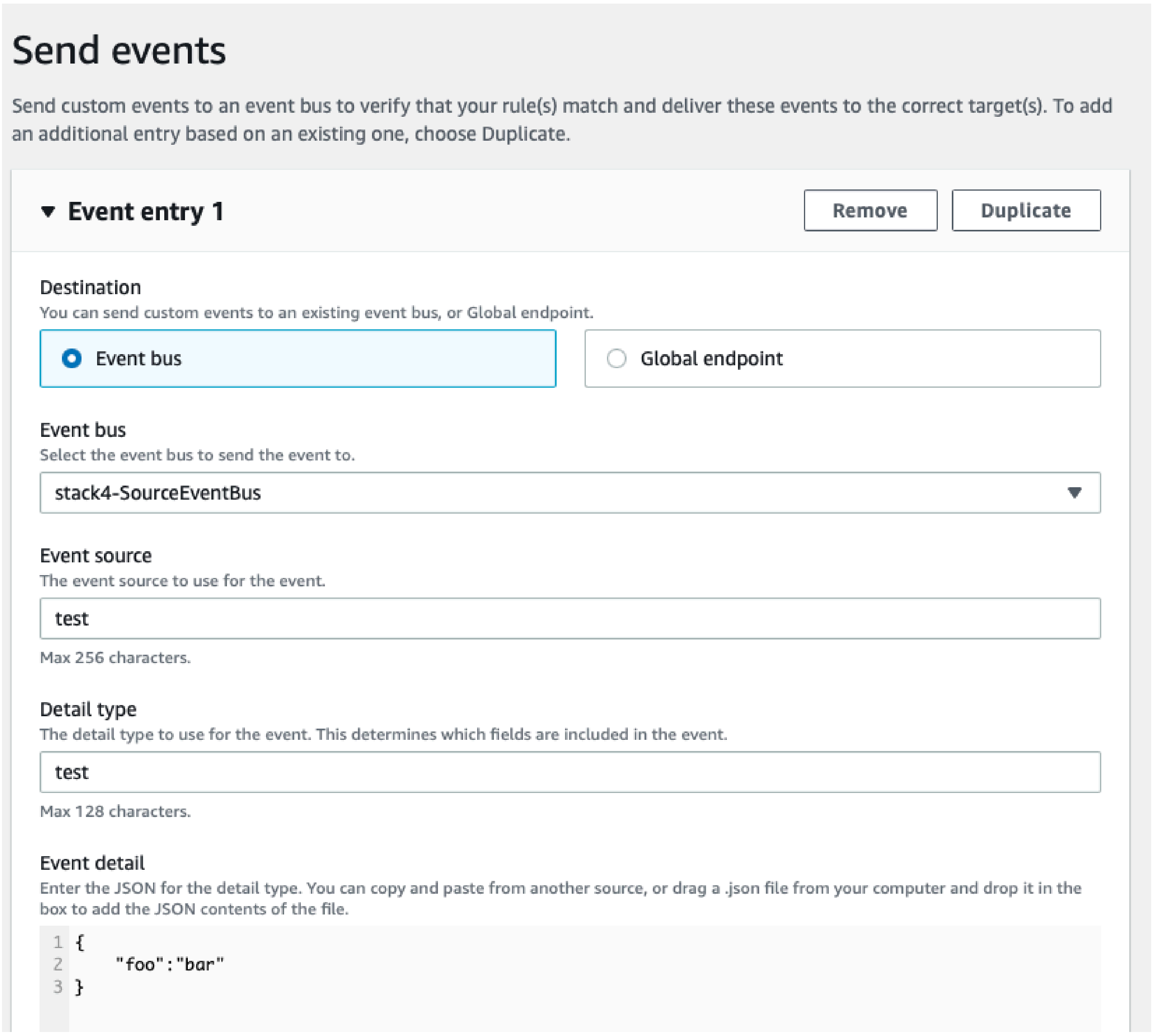


Figure 5: The sample project architecture, delivering events cross-account to Lambda, SQS, and SNS.

Make sure you enter a valid email address as SnsSubscriptionEmail value and confirm your email subscription once target stack is deployed.

After deployment, open the EventBridge console in the source account. Navigate to the newly created event bus, which has “SourceEventBus” in its name. Use the *Send Events* functionality to publish sample events, as shown in the following screenshot. Make sure that the *Source* of your events is set to “test”.


Send events

Send custom events to an event bus to verify that your rule(s) match and deliver these events to the correct target(s). To add an additional entry based on an existing one, choose Duplicate.

Event entry 1

Destination
You can send custom events to an existing event bus, or Global endpoint.

[✓ Event bus] Global endpoint

Event bus
Select the event bus to send the event to.
stack4-SourceEventBus

Event source
The event source to use for the event.
test
Max 256 characters.

Detail type
The detail type to use for the event. This determines which fields are included in the event.
test
Max 128 characters.

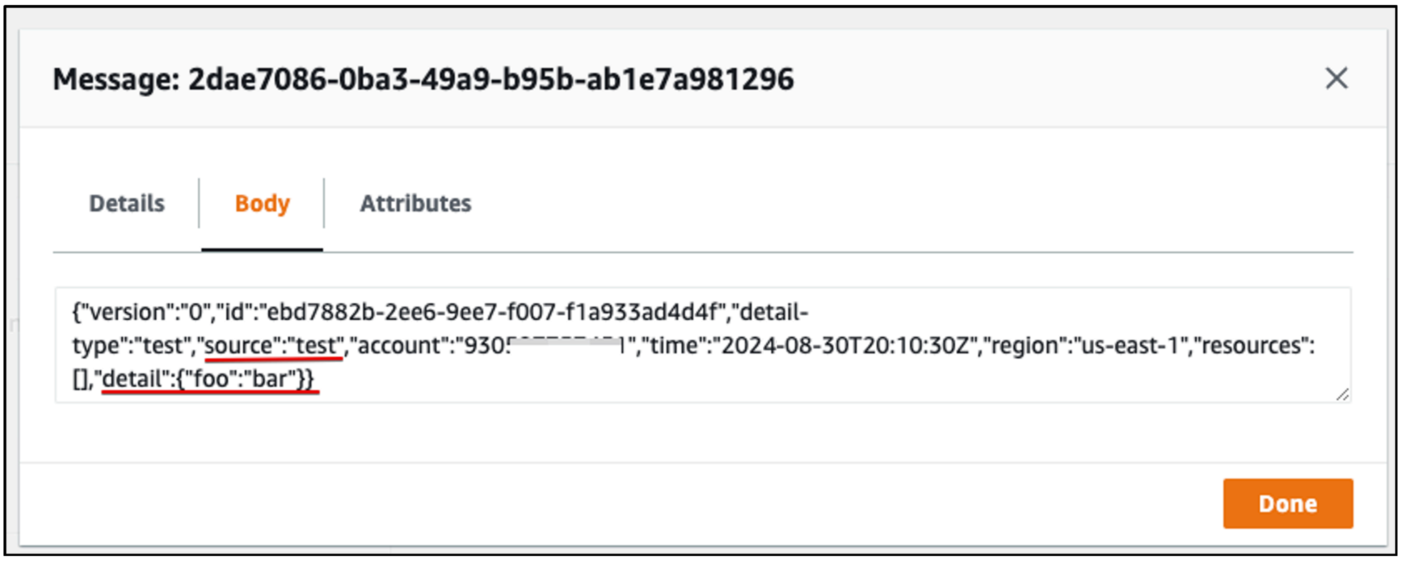
Event detail
Enter the JSON for the detail type. You can copy and paste from another source, or drag a .json file from your computer and drop it in the box to add the JSON contents of the file.

```
{
    "foo":"bar"
}
```

Figure 6: Sending test event

Validate that the events are successfully delivered to all three cross-account targets. Open the target account in a different browser or an incognito window:

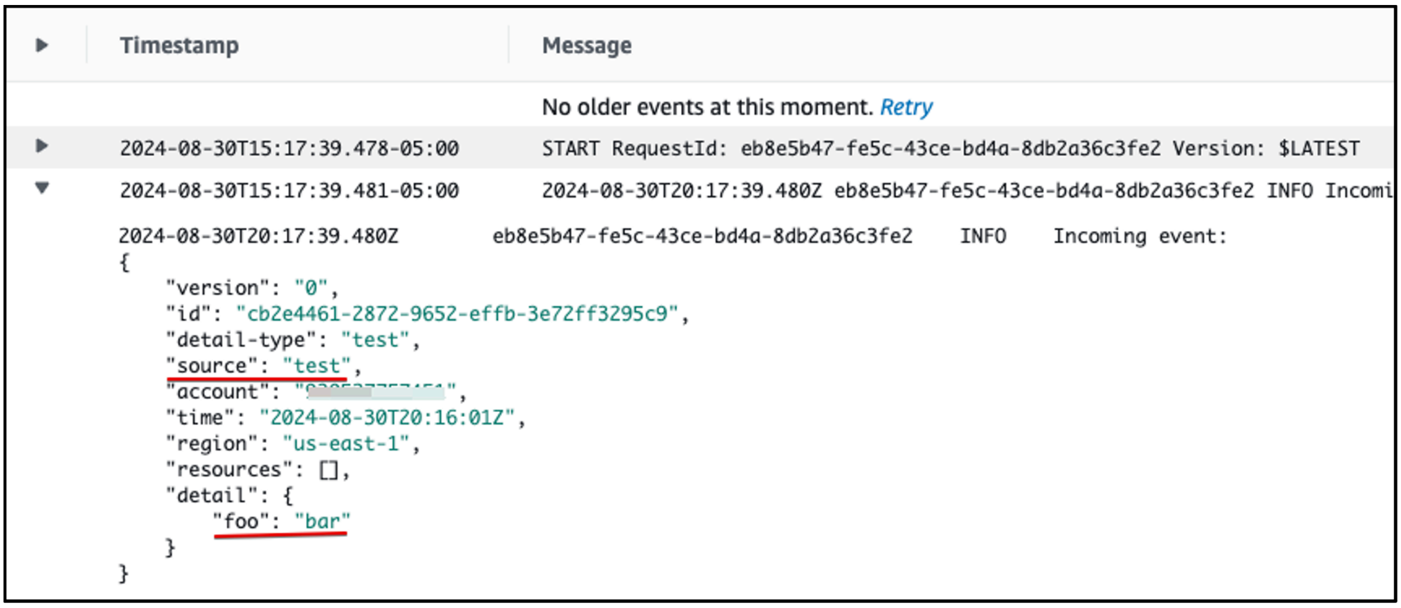
1. Navigate to the SQS console. Open the newly created queue, which has “TargetSqsQueue” in its name.
2. Choose **Send and Receive messages** then choose **Poll for messages**. You can see the events sent in the previous step.



```
{"version":"0","id":"ebd7882b-2ee6-9ee7-f007-f1a933ad4d4f","detail-type":"test","source":"test","account":"930[redacted]1","time":"2024-08-30T20:10:30Z","region":"us-east-1","resources":[],"detail":{"foo":"bar"}}
```

Figure 7: Receiving test event with SQS

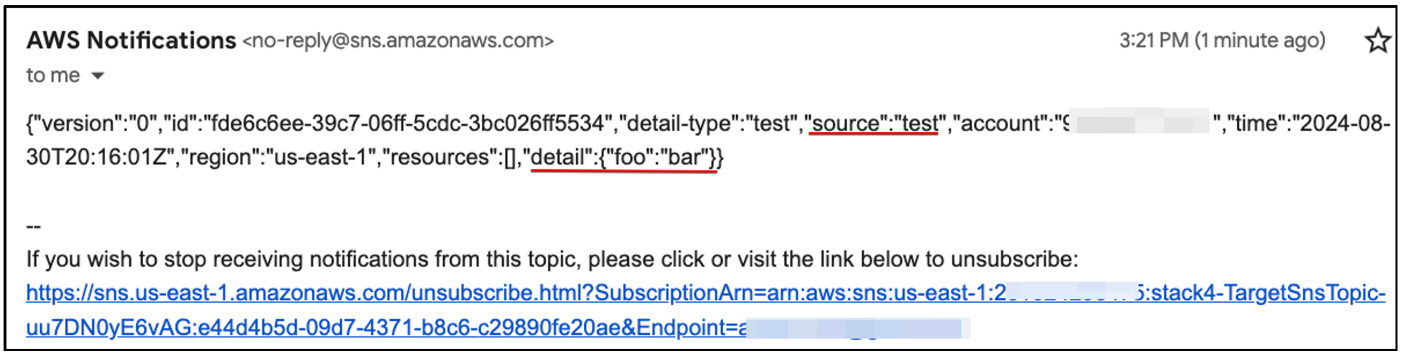
3. Navigate to Amazon CloudWatch Logs. Open the Log Group for the newly created Lambda function, which has “TargetLambdaFunction” in its name. It shows events sent in the previous step.



```
    Timestamp                        Message
                                     No older events at this moment. Retry
▶   2024-08-30T15:17:39.478-05:00    START RequestId: eb8e5b47-fe5c-43ce-bd4a-8db2a36c3fe2 Version: $LATEST
▼   2024-08-30T15:17:39.481-05:00    2024-08-30T20:17:39.480Z eb8e5b47-fe5c-43ce-bd4a-8db2a36c3fe2 INFO Incomi
    2024-08-30T20:17:39.480Z      eb8e5b47-fe5c-43ce-bd4a-8db2a36c3fe2    INFO    Incoming event:
    {
        "version": "0",
        "id": "cb2e4461-2872-9652-effb-3e72ff3295c9",
        "detail-type": "test",
        "source": "test",
        "account": "[redacted]",
        "time": "2024-08-30T20:16:01Z",
        "region": "us-east-1",
        "resources": [],
        "detail": {
            "foo": "bar"
        }
    }
```

Figure 8: Receiving test event with Lambda

4. Check your email. If you have confirmed the SNS topic subscription during the sample code deployment, it shows the events sent in the previous step.



AWS Notifications <no-reply@sns.amazonaws.com> 3:21 PM (1 minute ago)
to me

{"version":"0","id":"fde6c6ee-39c7-06ff-5cdc-3bc026ff5534","detail-type":"test","source":"test","account":"9[redacted]","time":"2024-08-30T20:16:01Z","region":"us-east-1","resources":[],"detail":{"foo":"bar"}}

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:2[redacted]5:stack4-TargetSnsTopic-uu7DN0yE6vAG:e44d4b5d-09d7-4371-b8c6-c29890fe20ae&Endpoint=a[redacted]

Figure 9: Receiving test event with SNS

Conclusion

The new EventBridge capability allows you to deliver events directly to targets across account boundaries. This capability helps to simplify your event-driven architectures, as well as improve latency by reducing the number of components processing your events as they travel from event buses to their destinations.

Refer to the EventBridge pricing page to learn more about cross-account events delivery costs.

For additional documentation, refer to Amazon EventBridge documentation. Get the sample code used in this blog from this GitHub repository.

For more serverless learning resources, visit Serverless Land.

TAGS: serverless