

- Customer Managed
 - kms_decrypt_policy

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:*:317185619046:key/*"
  }
}
```

- In Resource section add account id of staging account.

2. Create lambda in prod account and add below policies in that lambda role. (**Note:** Production account lambda has been created by cfn template initially. "AWS Mangaed" policies also created by template, so only create "Customer Managed" policies for this step.)

- Policy Used.
 - AWS Mangaed
 1. AmazonEventBridgeFullAccess
 2. AmazonS3FullAccess
 3. AmazonSageMakerFullAccess
 4. AWSCloudFormationFullAccess
 - Customer Managed
 1. cross-account-kms-decrypt-policy-1

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Encrypt",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:*:317185619046:key/*"
  }
}
```

- In resource section used account id of staging account
- 2. lambda_assume_cross_account_role_policy

```
{
  "Version": "2012-10-17",
```

```

    "Statement": [
      {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource":
"arn:aws:iam::317185619046:role/staging_S3_Sagemaker_policy_role"
      }
    ]
  }

```

- In resource section added arn of role which assumed in staging account.
3. s3-cross-account-policy-1

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "arn:aws:s3::sagemaker-project-p-69wapckwev1l/*"
    }
  ]
}

```

- In resource section used bucket name from staging account where all data is present.
4. access_staging_private_ecr_policy(To access ECR Image from staging account)

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchCheckLayerAvailability"
      ],

```

```

        "Resource": "arn:aws:ecr:us-east-1:345594592951:repository/sagemaker-inference"
      }
    ]
  }

```

- In Resource section add arn of repo where inference image is present.

3. Create one Sagemaker execution role in production account.

- Add below policy in the role.
 - AWS Managed
 1. AmazonS3FullAccess
 2. AmazonSageMakerFullAccess
 - Customer Managed
 - access_ecr_stage_policy(To access ECR Image from staging account)

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchCheckLayerAvailability"
      ],
      "Resource": "arn:aws:ecr:us-east-1:345594592951:repository/sagemaker-inference"
    }
  ]
}

```

- In Resource section add arn of repo where inference image is present.

4. Create one role in staging account which have permission to access the buckets which work as assumed role for role of lambda from prod account.

- Role: staging_S3_Sagemaker_policy_role
- Add below policy in that.
 - AmazonSageMakerFullAccess
 - AmazonS3FullAccess

- Add below in trust relationship section.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::654654428824:role/service-
role/pull-artifact-trigger-template-role-xqd25oyu"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- In principal section added arn of lambda role from prod account.
- Add below policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": [
        "arn:aws:s3:::axcess-devst-sagemaker/*",
        "arn:aws:s3:::sagemaker-us-east-1-345594592951/*",
        "arn:aws:s3:::axcess-devst-sagemaker",
        "arn:aws:s3:::sagemaker-us-east-1-345594592951"
      ]
    }
  ]
}
```

- In Resource section add arn of s3 buckets which has models and other files available.

5. In cross account deployment Lambda function from prod account need to access the ECR repository from staging account where Inference images are present.

- Go to the staging account Amazon ECR console.
- Click on the repository where inference image are present.
- Click on Permissions in left side.
- Click on Edit policy JSON and add below policy in that.

```

{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AllowProdLambdaPull",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::361769565206:role/LambdaExecutionRole-
prod-account",
          "arn:aws:iam::361769565206:role/AmazonSageMakerServiceCatalogProductsU
seRoleMultiModelTB"
        ]
      },
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ]
    }
  ]
}

```

- In Principal section add arn of Lambda execution role from prod account and Sagemaker execution role from prod account.

6. Create new stage after staging stage in code pipeline. In that add staging account Lambda function as action provider and other details and click on Done. (**Note:** This point i.e 6'th number: This step has been created while launching custom project. So you can skip the 6'th point.)

Edit action

Action name
Choose a name for your action
Push events from codepipeline to lambda
No more than 100 characters

Action provider
AWS Lambda

Region
US East (N. Virginia)

Input artifacts
Choose an input artifact for this action. [Learn more](#)
BuildArtifact
Add
No more than 100 characters

Function name
Choose a function that you have already created in the AWS Lambda console. Or create a function in the AWS Lambda console and then return to this task.
xfactor-test-pull-artifact-from-codepipeline
Function name contains only letters, numbers, hyphens, or underscores with no spaces. This does not include the function alias or function ARN.

User parameters - optional
This string will be used in the event data parameter passed to the handler in AWS Lambda.
Type a value if required by the function

Variable namespace - optional
Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Output artifacts
Choose a name for the output of this action.
Add
No more than 100 characters

Cancel Done

- In this section choose lambda function name which we created in staging account.

1. Now whenever pipeline will run it push the events to the lambda which we have created in staging account.
2. From the event we can get s3 path of zip files.

Step-2

1. Create custom event bus in prod account. (**Note:** No need to create event bus in prod account, already created initially while launching resources in prod account using cfn template. Need to complete other permissions related and otehr activities.)
 - Click on event bus and click on create event bus and choose below steps which given in images.

The screenshot displays the 'Create event bus' interface in the Amazon EventBridge console. The 'Event bus detail' section contains a 'Name' field with the value 'my-event-bus' and a 'Description - optional' field. The 'Encryption info' section provides details on server-side encryption and offers two options for the AWS Key Management Service (KMS) key: 'Use AWS owned key' (selected) and 'Choose a different AWS KMS key (advanced)'. The 'Resource-based policy' section includes a 'Load template' button. The bottom section shows optional features: 'Archives' (Enabled), 'Schema discovery' (Enabled), and 'Tags' (No tags associated). The 'Create' button is highlighted in orange at the bottom right.

- Click on Create.
2. Create one Eventbridge rule in default event bus in staging account. (**Note:** No need to create eventbridge rule in staging account, already created while creating custom project. Need to complete other permissions related and otehr activities.)

- Click on create rule.

Amazon EventBridge > Rules > x_factor_cross_account_event_rule > Edit rule

Step 1
Define rule detail

Step 2
[Build event pattern](#)

Step 3
[Select target\(s\)](#)

Step 4 - optional
[Configure tags](#)

Step 5
[Review and update](#)

Define rule detail [Info](#)

Rule detail

Name
x_factor_cross_account_event_rule
Maximum of 64 characters consisting of numbers, lower/upper case letters, -, _.

Description - optional

Event bus [Info](#)
Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.
default

☒ **Enable the rule on the selected event bus**

Rule type [Info](#)

☒ **Rule with an event pattern**
A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.

☐ **Schedule**
A rule that runs on a schedule.

Cancel **Next**

Amazon EventBridge > Rules > x_factor_cross_account_event_rule > Edit rule

Step 1
[Define rule detail](#)

Step 2
Build event pattern

Step 3
[Select target\(s\)](#)

Step 4 - optional
[Configure tags](#)

Step 5
[Review and update](#)

Build event pattern [Info](#)

Event source

Event source
Select the event source from which events are sent.

☐ **AWS events or EventBridge partner events**
Events sent from AWS services or EventBridge partners.

☒ **Other**
Custom events or events sent from more than one source, e.g. events from AWS services and partners.

☐ **All events**
All events sent to your account.

Sample event - optional
You don't have to select or enter a sample event, but it's recommended so you can reference it when writing and testing the event pattern, or filter criteria.

You can reference the sample event when you write the event pattern, or use the sample event to test if it matches the event pattern. Find a sample event, enter your own, or edit a sample event below. [Learn more about the required fields in a sample event.](#)

Sample event type

☒ **AWS events** ☐ **EventBridge partner events** ☐ **Enter my own**

Sample events
Filter by event source and type or by keyword.

Select

1

Enter the event JSON

Copy

Creation method

Method

☐ Use schema
Use an Amazon EventBridge schema to generate the event pattern.

☐ Use pattern form
Use a template provided by EventBridge to create an event pattern.

☒ Custom pattern (JSON editor)
Write an event pattern in JSON.

Event pattern [Info](#)

Event pattern

Write an event pattern in JSON. You can test the event pattern against the sample event. You can also go to pre-defined pattern.

Prefix matching

Insert

Content-based filter syntax

```

1 {
2   "source": ["*factor-sagemaker-lambda-events"],
3   "detail-type": ["sagemaker-myDetailType"]
4 }

```

☒ JSON is valid

Copy

Prettify

Event pattern form

Test pattern

Cancel

Previous

Next

Amazon EventBridge > Rules > x_factor_cross_account_event_rule > Edit rule

Step 1

Define rule detail

Step 2

Build event pattern

Step 3

Select target(s)

Step 4 - optional

Configure tags

Step 5

Review and update

Select target(s)

Permissions

Note: When using the EventBridge console, EventBridge will automatically configure the proper permissions for the selected targets. If you're using the AWS CLI, SDK, or CloudFormation, you'll need to configure the proper permissions.

Target 1

Target types

Select an EventBridge event bus, EventBridge API destination (SaaS partner), or another AWS service as a target.

☒ EventBridge event bus
☐ EventBridge API destination
☐ AWS service

Target types

Select an EventBridge event bus, EventBridge API destination (SaaS partner), or another AWS service as a target.

☐ Event bus in the same account and Region
☒ Event bus in a different account or Region

Event bus as target

arn:aws:events:us-east-1:54654428824:event-bus/cross-account-bus

Execution role

EventBridge needs permission to send events to the event bus of the above AWS account. By continuing, you are allowing us to do so.

[EventBridge and AWS Identity and Access Management](#)

☒ Create a new role for this specific resource
☐ Use existing role

Role name

Amazon_EventBridge_Invoke_Event_Bus_717346451

Additional settings

Add another target

Cancel

Skip to Review and update

Previous

Next

- Refer Above Image: In this section choose eventbridge event bus in target type and add arn of prod account event bus. Also select on "Create a new role for this specific resources" option in Execution Role section and enable "Use execution role (recommended)" option.

Event pattern [Info](#)

```

1 {
2   "source": ["*factor-sagemaker-lambda-events"],
3   "detail-type": ["sagemaker-myDetailType"]
4 }

```

Copy

Step 3: Select target(s)

Edit

Targets

Details	Target Name	Type	Arn	Input	Role
<div> <div>am:aws:events:us-east-1:54654428824:event-bus/cross-account-bus</div> <div>Event bus in a different account or Region</div> </div>			<div> <div>arn:aws:events:us-east-1:54654428824:event-bus/cross-account-bus</div> <div>Matched event</div> </div>	<div> <div>Amazon_EventBridge_Invoke_Event_Bus_793415824</div> <div></div> </div>	

Input to target:

Matched event

Additional parameters:

--

Dead-letter queue (DLQ):

--

Step 4: Configure tag(s)

Edit

Tags (0)

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value
No tags associated with this resource.	

Cancel

Previous

Update rule

- Refer Above Image: Click on Update Rule after review all things.

3. Create evnetbridge rule in event bus which created in above steps in prod account. (**Note:** Eventbridge rule in prod account has been created by cfn template initially. No need to create rule but need to complete other permissions related and otehr activities.)

- choose the options which given in images.

The screenshot shows the 'Define rule detail' step in the Amazon EventBridge console. The breadcrumb navigation is 'Amazon EventBridge > Rules > cross-account-rule > Edit rule'. The left sidebar shows the steps: Step 1: Define rule detail (active), Step 2: Build event pattern, Step 3: Select target(s), Step 4 - optional: Configure tags, and Step 5: Review and update. The main content area is titled 'Define rule detail' with an 'Info' link. It contains a 'Rule detail' section with a 'Name' field set to 'cross-account-rule', a 'Description - optional' field, and an 'Event bus' dropdown set to 'cross-account-bus'. A warning message states: 'Schedule rule is not supported when custom or partner event bus is selected.' Below this, there is a checkbox 'Enable the rule on the selected event bus' which is checked. The 'Rule type' section has two options: 'Rule with an event pattern' (selected) and 'Schedule'. The 'Rule with an event pattern' option has a description: 'A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.' The 'Schedule' option has a description: 'A rule that runs on a schedule'. At the bottom right, there are 'Cancel' and 'Next' buttons.

The screenshot shows the 'Build event pattern' step in the Amazon EventBridge console. The breadcrumb navigation is 'Amazon EventBridge > Rules > cross-account-rule > Edit rule'. The left sidebar shows the steps: Step 1: Define rule detail, Step 2: Build event pattern (active), Step 3: Select target(s), Step 4 - optional: Configure tags, and Step 5: Review and update. The main content area is titled 'Build event pattern' with an 'Info' link. It contains an 'Event source' section with three radio button options: 'AWS events or EventBridge partner events', 'Other' (selected), and 'All events'. The 'Other' option has a description: 'Custom events or events sent from more than one source, e.g. events from AWS services and partners.' Below this is a 'Sample event - optional' section with a description: 'You don't have to select or enter a sample event, but it's recommended so you can reference it when writing and testing the event pattern, or filter criteria.' It includes a text area for 'You can reference the sample event when you write the event pattern, or use the sample event to test if it matches the event pattern. Find a sample event, enter your own, or edit a sample event below. Learn more about the required fields in a sample event.' Below this is a 'Sample event type' section with three radio button options: 'AWS events' (selected), 'EventBridge partner events', and 'Enter my own'. Below this is a 'Sample events' section with a filter by event source and type and by keyword. It includes a 'Select' dropdown menu and a list of sample events. At the bottom, there is a text area labeled 'Enter the event JSON'.

Creation method

Method

☐ Use schema
Use an Amazon EventBridge schema to generate the event pattern.

☐ Use pattern form
Use a template provided by EventBridge to create an event pattern.

☒ Custom pattern (JSON editor)
Write an event pattern in JSON.

Event pattern [Info](#)

Event pattern

Write an event pattern in JSON. You can test the event pattern against the sample event. You can also go to pre-defined pattern.

Prefix matching ▼

Insert

☒ Content-based filter syntax

```
1 {
2   "source": ["*factor-sagemaker-lambda-events"],
3   "detail-type": ["*sagemaker-anyDetailType"]
4 }
```

☒ JSON is valid

Copy

Prettify

Event pattern form

Test pattern

Cancel

Previous

Next

- o Refer Above Image: In this section add aws services as target type and in that add lambda function which created in prod account. Also select on "Create a new role for this specific resources" option in Execution Role section and enable "Use execution role (recommended)" option.

- Refer Above Image: Click on create.

4. Add Resource-based policy in the permission section of prod account event rule.

The screenshot shows the Amazon EventBridge console for the 'cross-account-bus' event bus. The left sidebar contains navigation links for Dashboard, Developer resources, Buses, Pipes, Scheduler, Integration, and Schema registry. The main content area displays the 'cross-account-bus' details, including its name, description, ARN, and discovery status. Below the details, the 'Permissions' tab is active, showing a resource-based policy. The policy is a JSON document that grants the 'events:PutEvents' action to the root user of the staging account (arn:aws:iam::317185619046:root) on the event bus resource (arn:aws:events:us-east-1:654654428824:event-bus/cross-account-bus).

Event bus details

Event bus name cross-account-bus	Schema discovery Not initiated	Created on Jun 25, 2024, 05:12 PM GMT+5:30
Description -	Discoverer ID No discoverer	Last modified Jun 25, 2024, 05:12 PM GMT+5:30
Event bus ARN arn:aws:events:us-east-1:654654428824:event-bus/cross-account-bus	Discoverer ARN No discoverer	

Resource-based policy

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [{
4     "Sid": "WebStoreCrossAccountPublish",
5     "Effect": "Allow",
6     "Principal": {
7       "AWS": "arn:aws:iam::317185619046:root"
8     },
9     "Action": "events:PutEvents",
10    "Resource": "arn:aws:events:us-east-1:654654428824:event-bus/cross-account-bus"
11  }]
12 }
```

[Manage permissions](#)

[Copy](#)

- In principal section used account id of staging account and in resource section added arn of event bus from prod account.