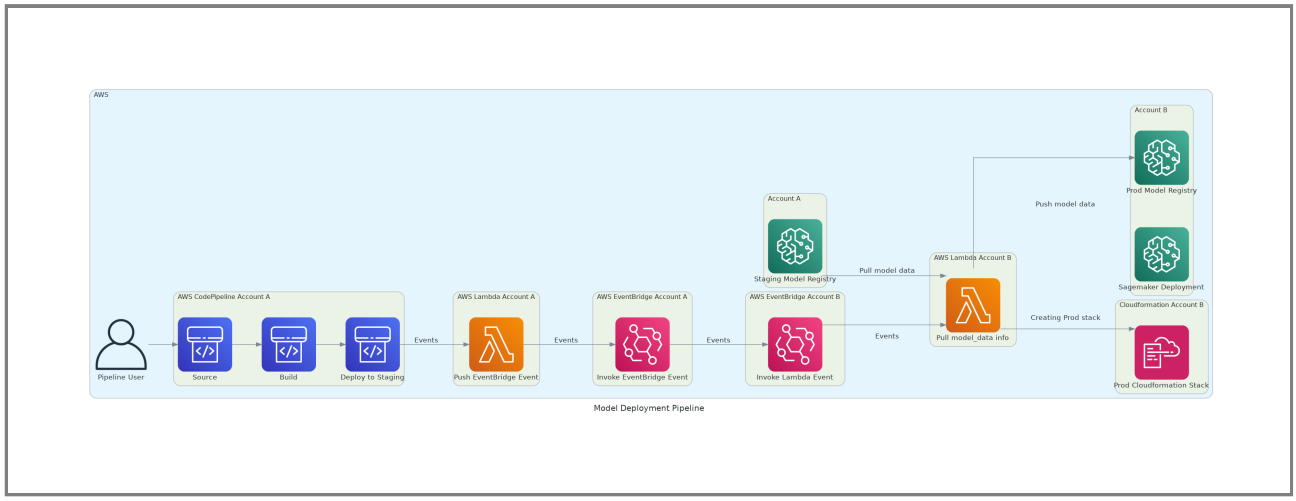
Cross-Account-Sagemaker-Model-Deployment

- Architecture Diagram.



- Steps
    1. Create two lambda functions, one in staging account and another is in prod account.
    2. In Staging account code-pipeline, after last stage add new stage and add target as lambda function.
    3. Create two event bridge rules one in staging account and one in prod account.
- Pre-requisite
    1. Cross account permission for eventbridge events.
    2. Cross account permission for S3 access.

Note: Resource in Staging account has been created while creating Sagemaker custom project so no need to create resource seperately in staging account. Only setting up permissions and other task need to complete, which are given below.

Note: Resource in Production account need create for cross account setup. For this CFN template has been provided. After launching resources in production account need to setting up permissions and other tasks, which are given below.

Get Started.

Step-1

1. Create one lambda function in staging account.(**Note:** This lambda function has been created by cfn template while creating custom project. "AWS Mangaed" polices also created by template, so only create "Customer Managed" policies for this step.)

- Add below polices to lambda role.
    - AWS Managed
        - AmazonEventBridgeFullAccess
        - AmazonS3FullAccess
        - AWSLambda_FullAccess

- 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 pilices in that lambda role.(**Note:** Production account lambda has been created by cfn template initially. "AWS Mangaed" polices also created by template, so only create "Customer Managed" policies for this step.)

- Ploicy 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 Mangaed
            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.

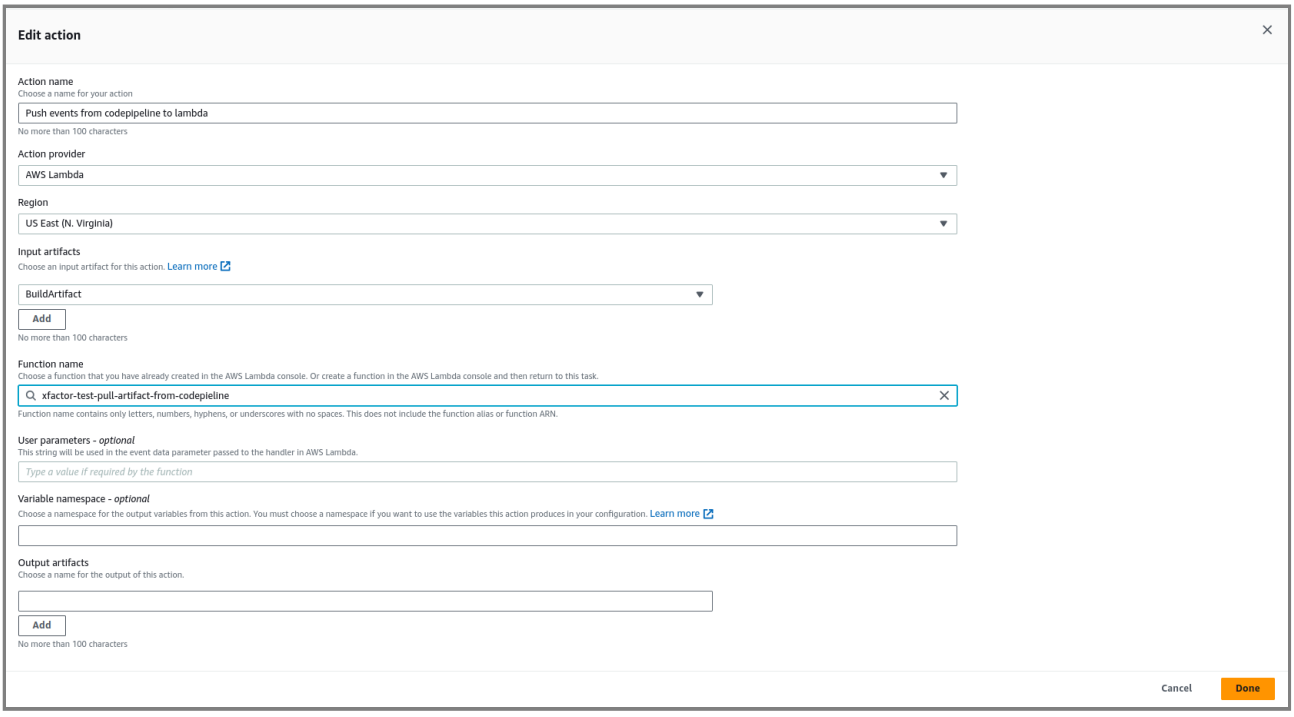
   - Go to the staging account Amazon ECR console.
   - Click on the repository where inference image are present.
   - Click on Permissions in left side.
   - Click on Edit policy JSON and add below policy in that.

```
    {
        "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.)

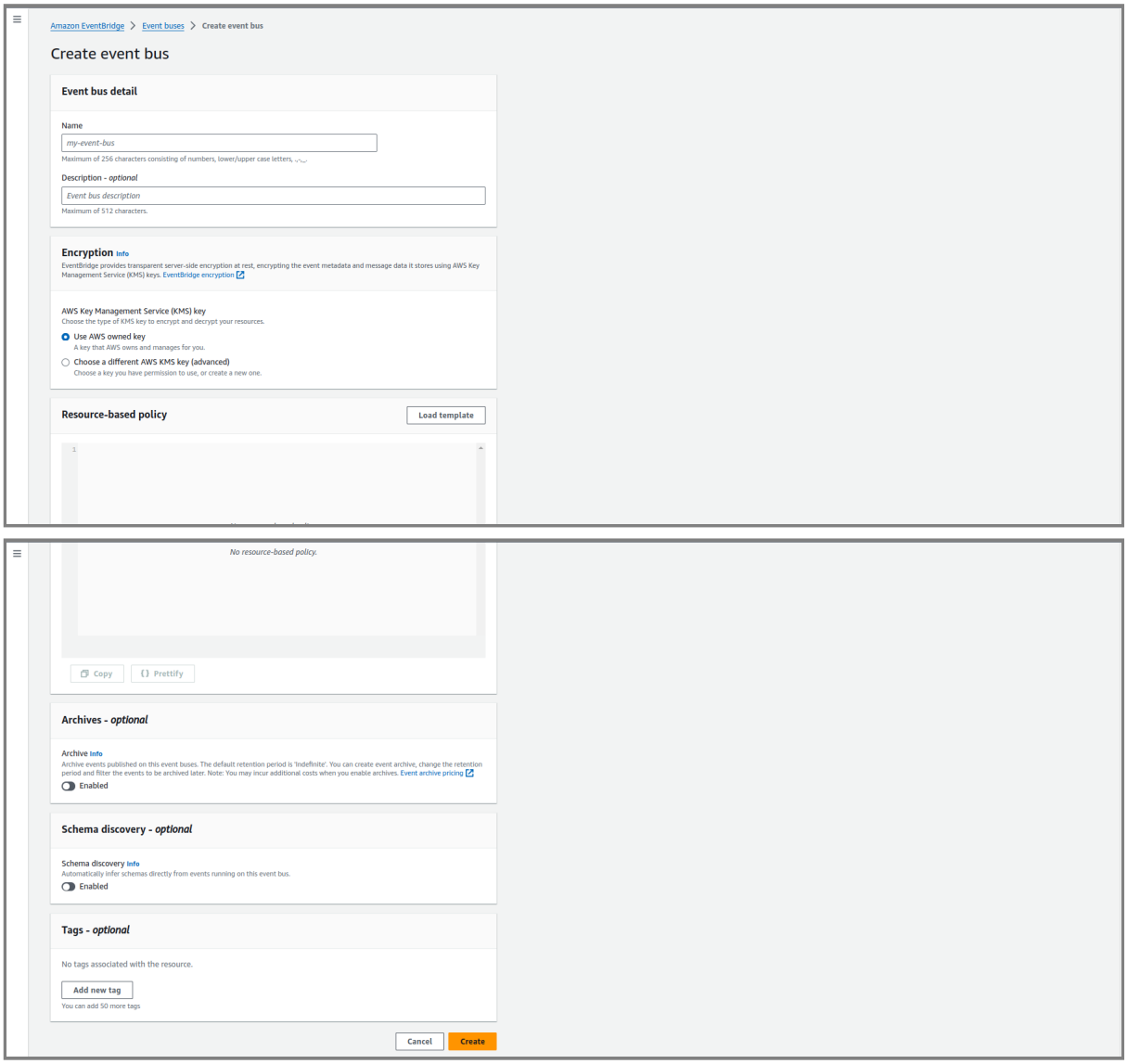


- In this section choose lambda function name which we created in staging account.

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

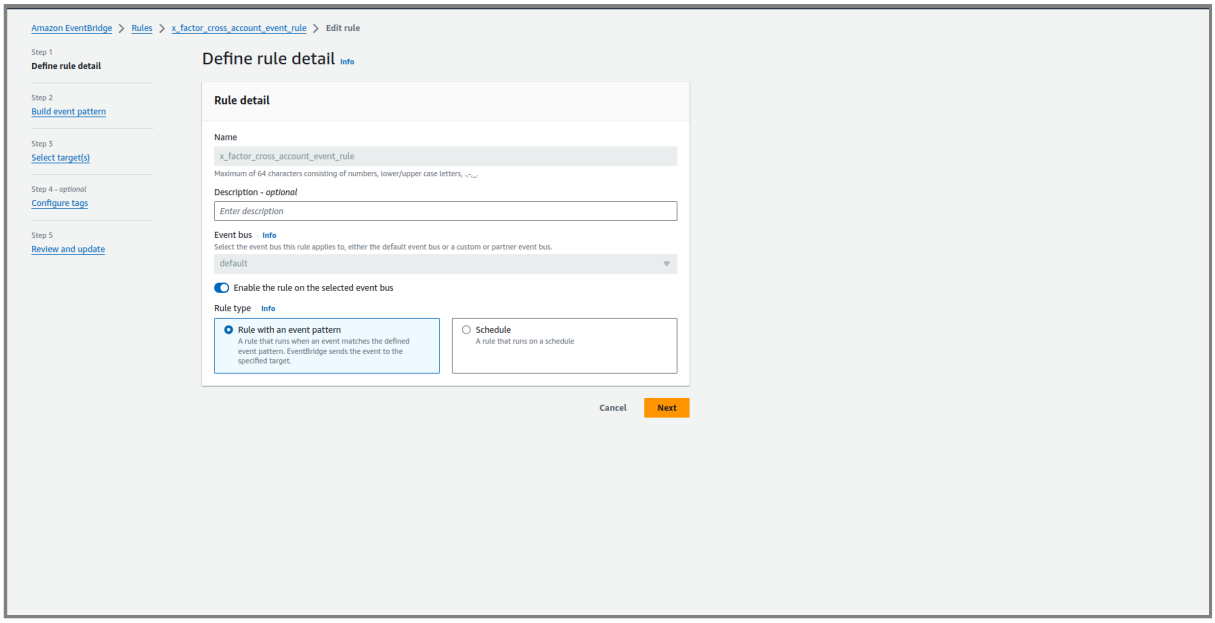
1. Create custom event bus in prod account.(**Note:** No need to create event bus in prod account, already created initially while launching resources in prod account using cfn template. Need to complete other permissions related and otehr activities.)
    - Click on event bus and click on create event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

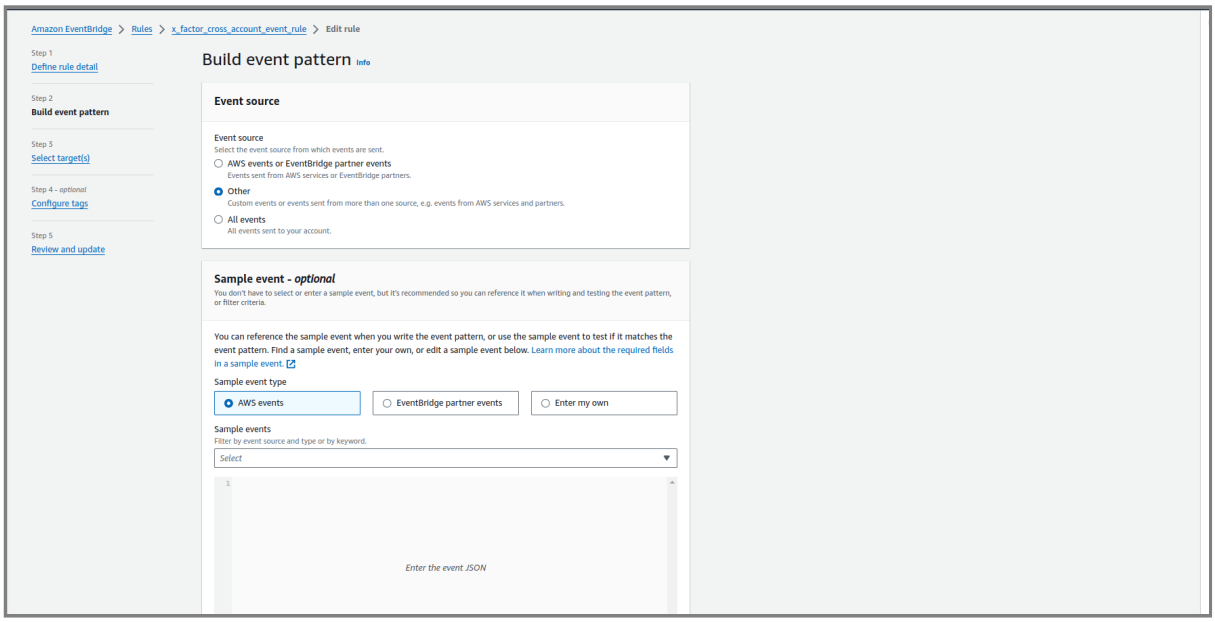


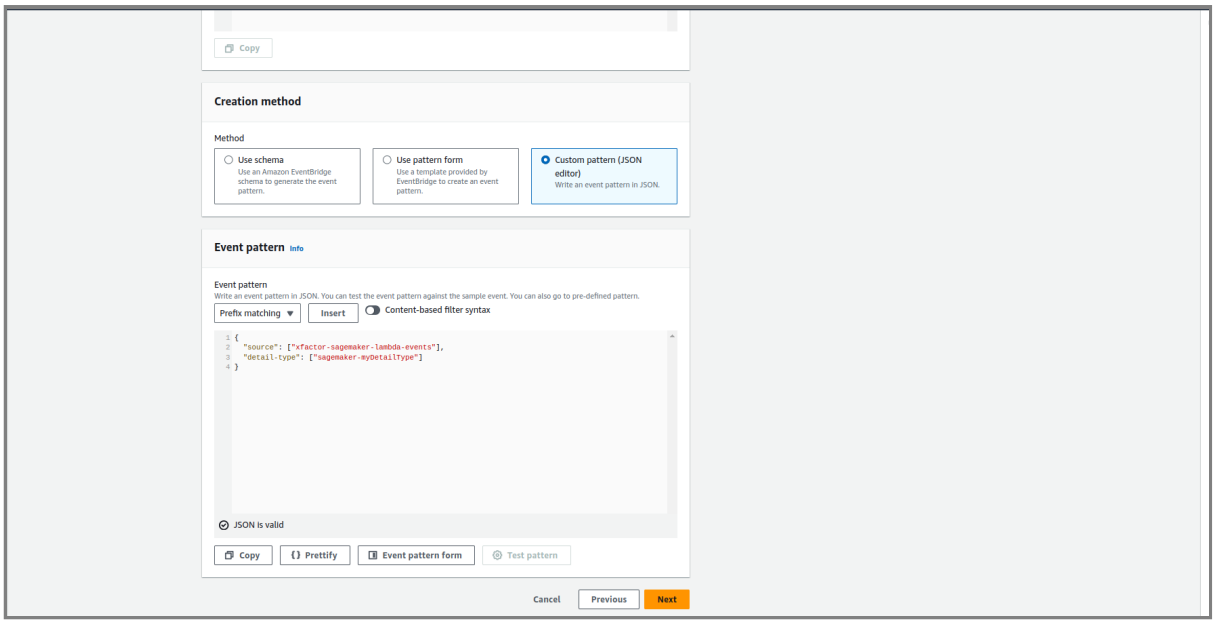
    - Click on Create.

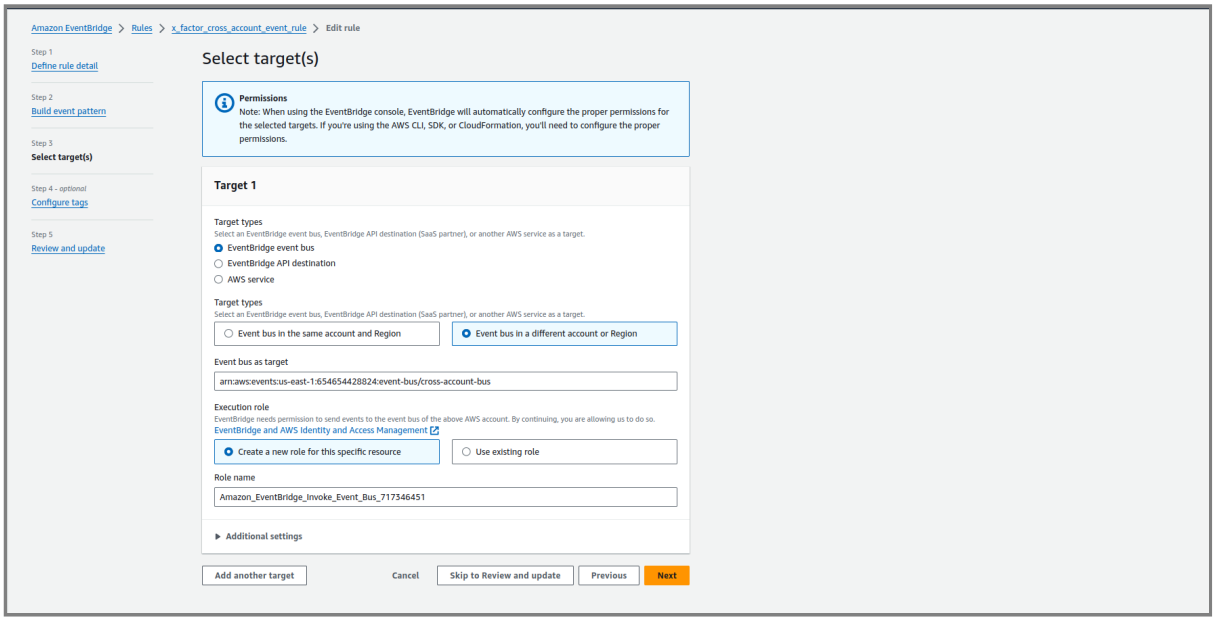
2. Create one Eventbridge rule in default event bus in staging account.(**Note:** No need to create eventbridge rule in staging account, already created while creating custom project. Need to complete other permissions related and otehr activities.)

- Click on create rule.

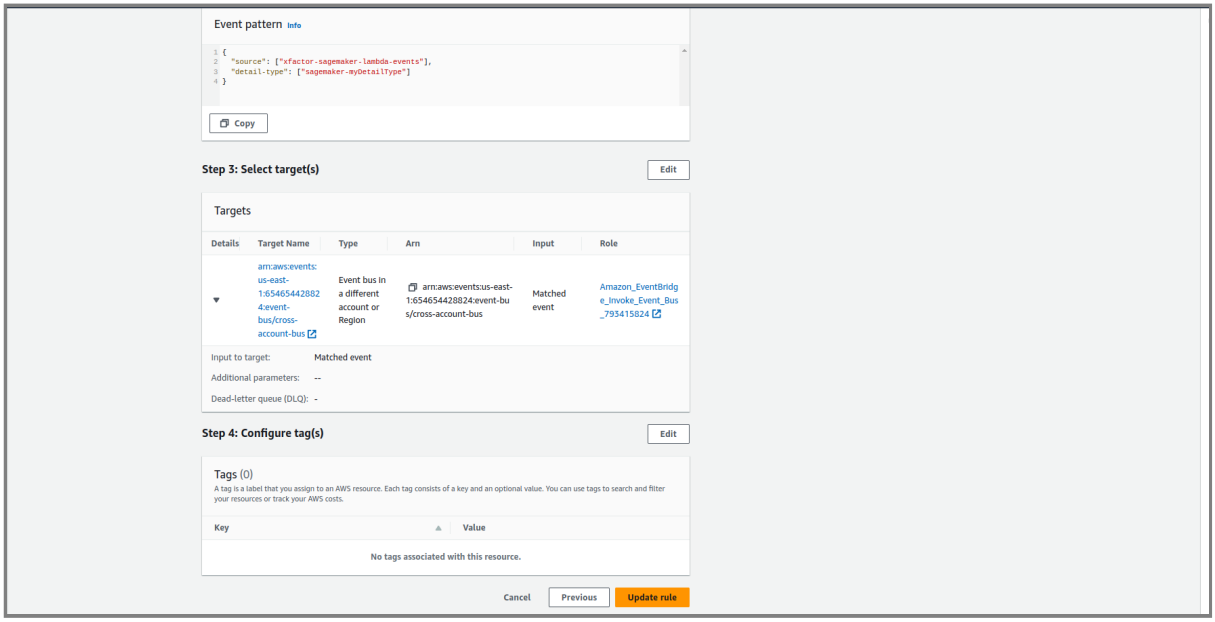








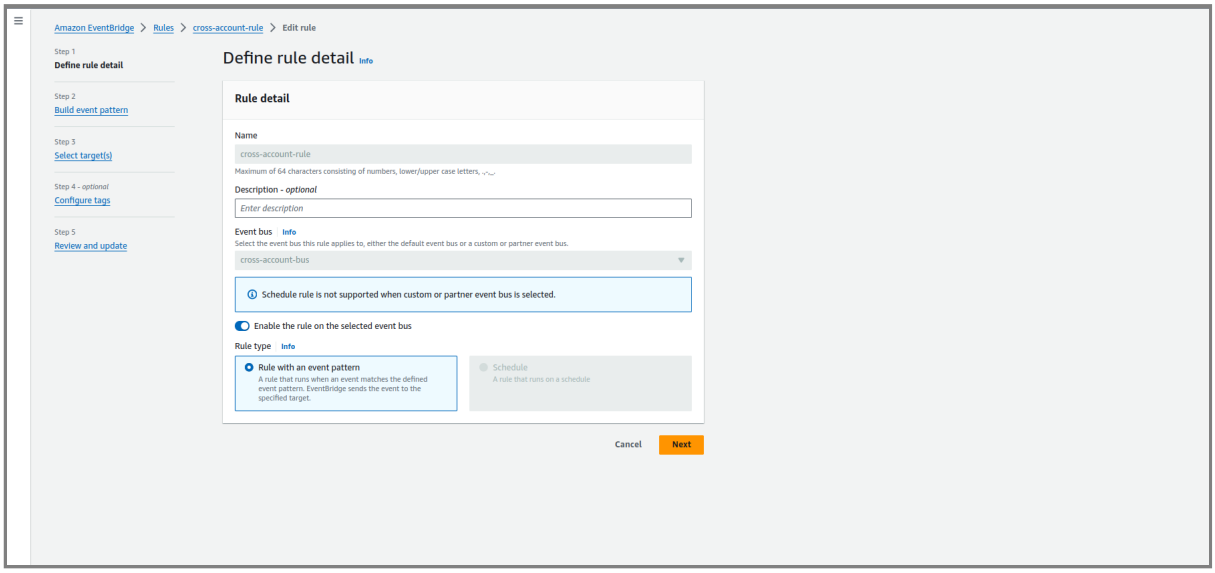
- Refer Above Image: In this section choose eventbridge event bus in target type and add arn of prod account event bus. Also select on "Create a new role for this specific resources" option in Execution Role section and enable "Use execution role (recommended)" option.

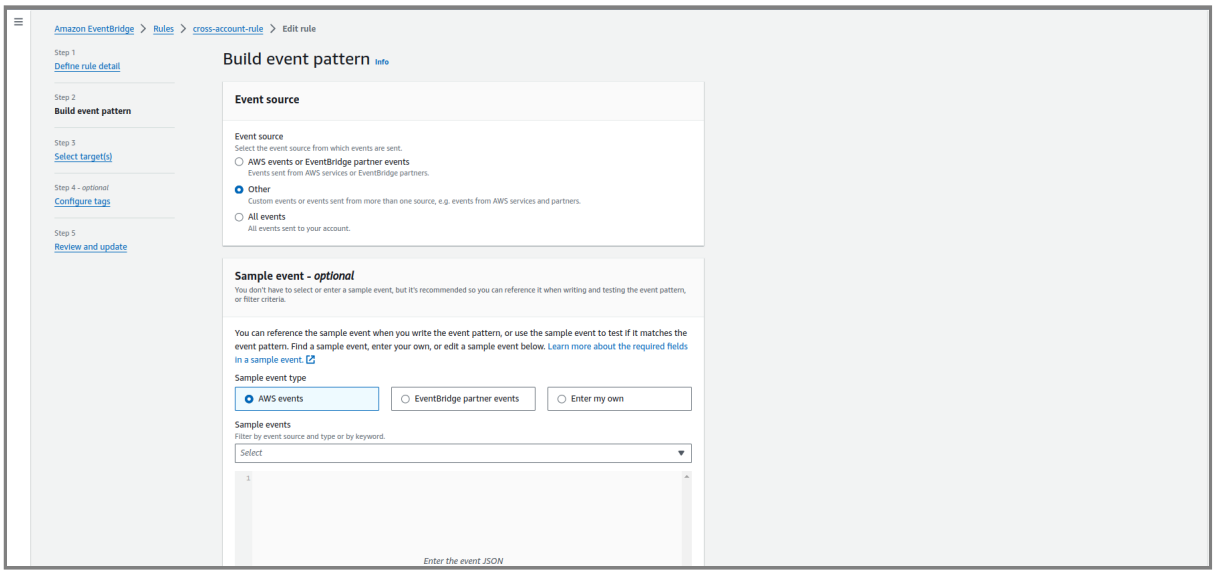


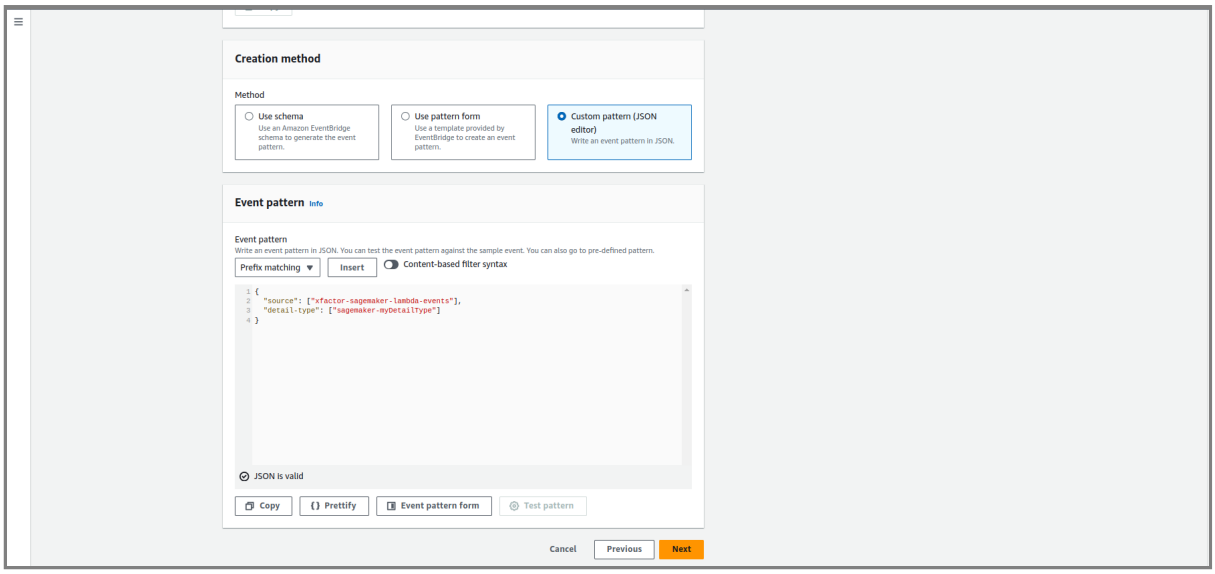
- Refer Above Image: Click on Update Rule after review all things.

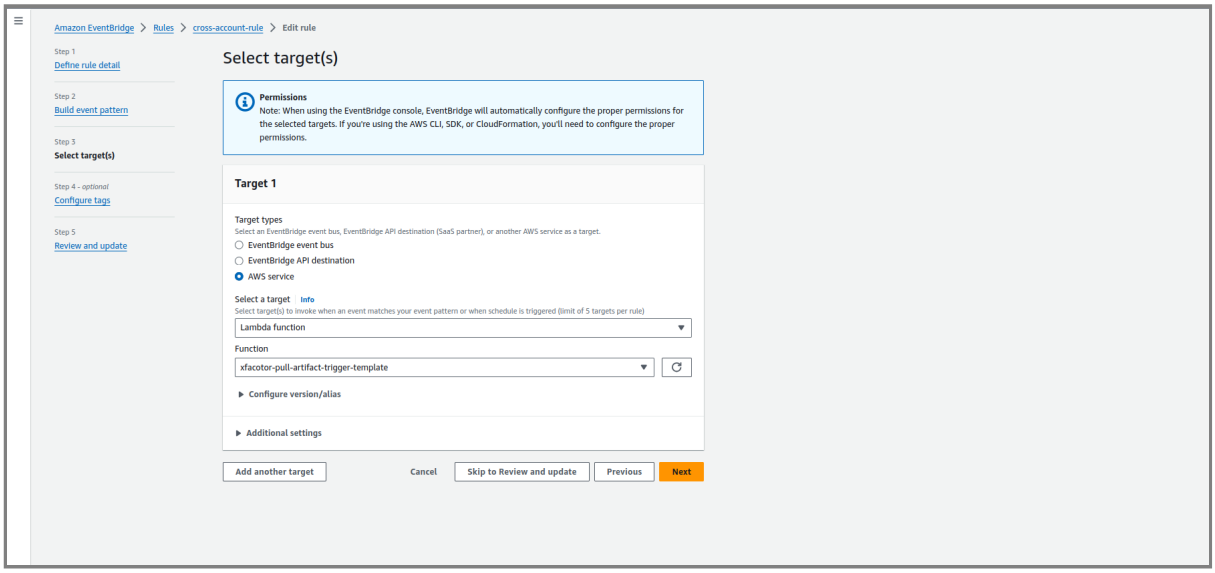
3. Create evnetbridge rule in event bus which created in above steps in prod account.(**Note:** Eventbridge rule in prod account has been created by cfn temlate initially. No need to create rule but need to complete other permissions related and otehr activities.)

    - choose the options which given in images.

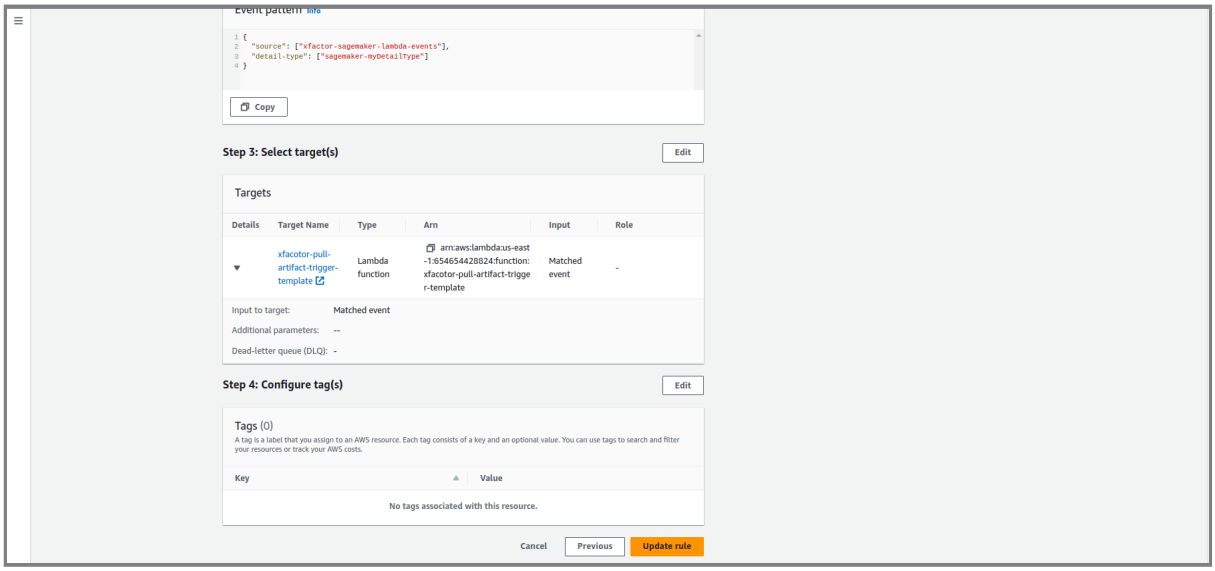






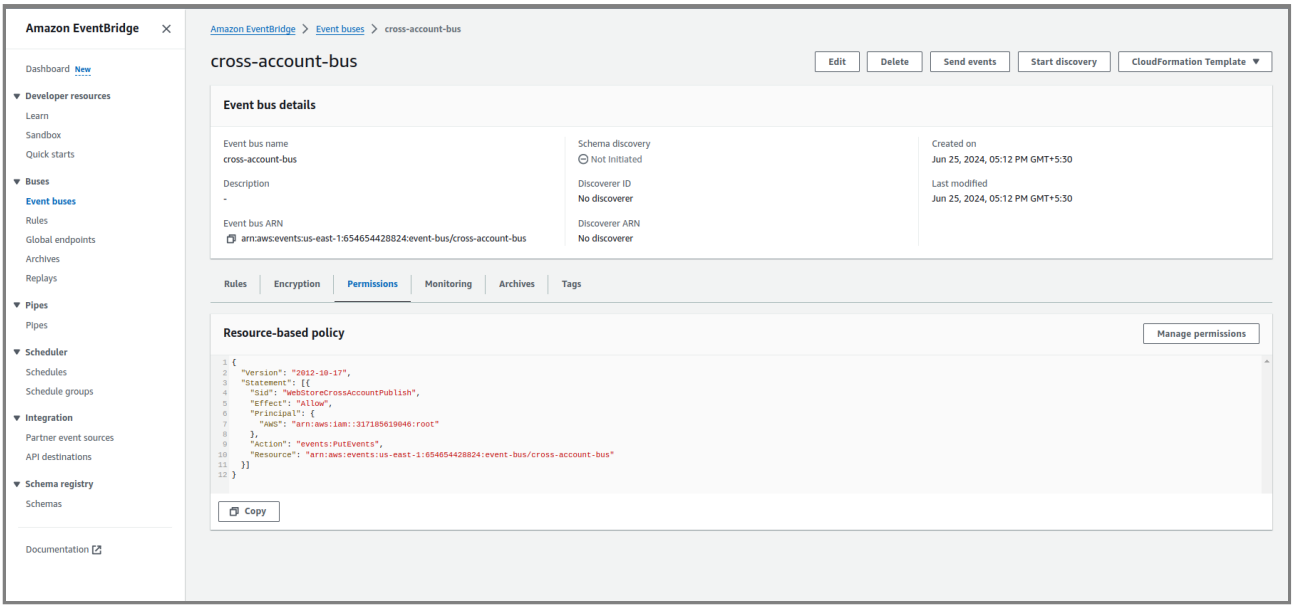


- Refer Above Image: In this section add aws services as target type and in that add lambda function which created in prod account. Also select on "Create a new role for this specific resources" option in Execution Role section and enable "Use execution role (recommended)" option.



- Refer Above Image: Click on create.

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




```
        {
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "WebStoreCrossAccountPublish",
        "Effect": "Allow",
        "Principal": {
        "AWS": "arn:aws:iam::317185619046:root"
        },
        "Action": "events:PutEvents",
        "Resource": "arn:aws:events:us-east-1:654654428824:event-
bus/cross-account-bus"
    }]
    }
```

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