# Installation Guide - CAT-IM AWS Marketplace

The purpose of this guide is to offer a comprehensive walkthrough for deploying CAT-IM via the AWS Marketplace and integrating it seamlessly with AWS native services.



## 1. Prerequisites

- **EKS cluster**: operational cluster deployed onto an existing VPC/Subnets
- **Kubernets Ingress Controller:** Configured in cluster. The IntellectEU recommended controller is Traefik
- **Helm**: Helm v3 installed on your local machine
- **SFTP Broker:** operational broker, for file service interactions
- **AWS CLI:** v2 installed. Required for CloudFormation deployments
- **Route53 Zone:** already configured, so we can set custom DNS aliases for our services
- **VPC:** An existing VPC configured with private/public subnets

## 2. Provision RDS Postgres

Deploy an RDS Postgres Instance. This can be through various methods (OpenTofu/Terraform, AWS CLI, etc), but for ease of using AWS native solutions, this section will provision an RDS cluster through a Cloudformation template.

### 2.1 Create Cloudformation manifest

Create a file ( `rds-stack.yaml` ) with the contents below:

```
AWSTemplateFormatVersion: 2010-09-09
Description: Provision an RDS PostgreSQL instance in an existing VPC and subnet group
Parameters:
  VpcId:
    Type: AWS::EC2::VPC::Id
```

```
    Description: The ID of the existing VPC
  VpcCidr:
    Type: String
    Description: CIDR block of the VPC (used as default for SG ingress rules)
  SubnetIds:
    Type: List<AWS::EC2::Subnet::Id>
    Description: A list of existing Subnet IDs for the DB subnet group (must be in the same
VPC)
  DBName:
    Type: String
    Default: message
    Description: The name of the initial PostgreSQL database
  DBUsername:
    Type: String
    Default: masteruser
    Description: The master username for PostgreSQL
  DBPassword:
    Type: String
    NoEcho: true
    Description: The master user password for PostgreSQL
  DBAllocatedStorage:
    Type: Number
    Default: 20
    Description: The size of the database (GB)
  DBInstanceClass:
    Type: String
    Default: db.t3.micro
    AllowedValues:
      - db.t3.micro
      - db.t3.small
      - db.t3.medium
      - db.t3.large
      - db.m5.large
      - db.m5.xlarge
    Description: The instance class for the RDS instance
  DBEngineVersion:
    Type: String
    Default: "14.17"
    Description: PostgreSQL engine version
  HostedZoneId:
    Type: String
    Description: The Route53 Hosted Zone ID where you want to create the record
  RecordName:
    Type: String
    Description: The DNS record name (e.g. db.example.com)

Resources:
  DBSubnetGroup:
    Type: AWS::RDS::DBSubnetGroup
    Properties:
      DBSubnetGroupDescription: Subnet group for PostgreSQL RDS instance
      SubnetIds: !Ref SubnetIds
  DBSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Security group for PostgreSQL RDS
      VpcId: !Ref VpcId
      SecurityGroupIngress:
```

```
        - IpProtocol: tcp
          FromPort: 5432
          ToPort: 5432
          CidrIp: !Ref VpcCidr
  PostgresDB:
    Type: AWS::RDS::DBInstance
    Properties:
      DBName: !Ref DBName
      AllocatedStorage: !Ref DBAllocatedStorage
      DBInstanceClass: !Ref DBInstanceClass
      Engine: postgres
      EngineVersion: !Ref DBEngineVersion
      MasterUsername: !Ref DBUsername
      MasterUserPassword: !Ref DBPassword
      VPCSecurityGroups:
        - !Ref DBSecurityGroup
      DBSubnetGroupName: !Ref DBSubnetGroup
      PubliclyAccessible: false
      BackupRetentionPeriod: 7
      DeletionProtection: false
      StorageEncrypted: true
      MultiAZ: false

  DBRecordSet:
    Type: AWS::Route53::RecordSet
    Properties:
      HostedZoneId: !Ref HostedZoneId
      Name: !Ref RecordName
      Type: CNAME
      TTL: 300
      ResourceRecords:
        - !GetAtt PostgresDB.Endpoint.Address
```

### 2.2 Apply Cloudformation stack

With the use of AWS CLI, deploy the Cloudformation stack, ensuring you replace placeholders with relevant values:

```
aws cloudformation create-stack \
  --stack-name rds-stack \
  --template-body file://rds-stack.yaml \
  --capabilities CAPABILITY_NAMED_IAM \
  --parameters \
    ParameterKey=VpcId,ParameterValue={REPLACEME} \
    ParameterKey=SubnetIds,ParameterValue=\"{REPLACEME_WITH_COMMA_DELIMITED_SUBNET_IDS}"\" \
    ParameterKey=VpcCidr,ParameterValue={REPLACEME} \
    ParameterKey=DBUsername,ParameterValue={REPLACEME} \
    ParameterKey=DBPassword,ParameterValue={REPLACEME} \
    ParameterKey=HostedZoneId,ParameterValue={REPLACEME} \
    ParameterKey=RecordName,ParameterValue={REPLACEME}
```

## 3 Provision Cognito

### 3.1 Create Cloudformation manifest

Create a file ( `cognito-stack.yaml` ) with the contents below:

```
AWSTemplateFormatVersion: 2010-09-09
Description: Cognito User Pool with domain, groups and Resource server with custom scopes

Parameters:
  UserPoolName:
    Type: String
    Default: catim-pool
    Description: The name of the Cognito User Pool
  AppClientName:
    Type: String
    Default: catim-ui
    Description: App client for OAuth2 access
  DomainPrefix:
    Type: String
    Default: catim
    Description: Unique prefix for Cognito hosted UI domain
  CatimUiDomain:
    Type: String
    Default: catim-ui.catims.intellecteu.io
    Description: The CATIM UI Dns

Resources:
  CognitoTokenCustomizerLambda:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: CognitoTokenCustomizer
      Handler: index.handler
      Runtime: nodejs18.x
      Role: !GetAtt CognitoLambdaExecutionRole.Arn
      Code:
        ZipFile: |
          exports.handler = function(event, context) {

            const groups = event.request.groupConfiguration.groupsToOverride || [];

            // Group → scopes mapping
            const groupScopes = {
              "Admin": [
                  "Message:Repair:SendToQueue",
                  "Settings:MessageApproval:Update",
                  "Settings:MessageApproval:View",
                  "Settings:View",
                  "Message:Routing:View",
                  "Message:Metadata:View",
                  "Message:Routing:Export",
                  "Message:Routing:Import",
                  "Message:Template:View",
                  "Message:Repair:Edit",
                  "Message:Details:View",
                  "Message:Dashboard:Export",
                  "Message:Body:View",
```

```
                "Message:Intervention:View",
                "Message:Validation:Allow",
                "Message:Validation:Block",
                "Message:Validation:Details:View",
                "Message:Export",
                "Message:Create",
                "Message:Create:Dashboard:View",
                "Message:Create:SendToQueue",
                "Message:Create:Route",
                "Message:Template:Create",
                "Message:Draft:Create",
                "Message:Template:Update",
                "Message:Template:Delete",
                "Message:Draft:Update",
                "Message:Draft:View",
                "Message:Draft:Delete",
                "Message:Validation:Approve:1",
                "Message:Validation:Reject:1",
                "Message:Validation:Approve:2",
                "Message:Validation:Reject:2",
                "Message:Validation:Approve:3",
                "Message:Validation:Reject:3",
                "Message:Repair:Route",
                "Message:Repair:RepairStatus"
            ],
            "Viewer": [
              "Message:Details:View",
              "Message:Body:View",
              "Message:Intervention:View",
              "Message:Validation:Details:View",
              "Message:Create:Dashboard:View",
              "Message:Template:View"
            ],
            "Operator": [
              "Message:Template:View",
              "Message:Details:View",
              "Message:Dashboard:Export",
              "Message:Body:View",
              "Message:Intervention:View",
              "Message:Export",
              "Message:Create",
              "Message:Create:Dashboard:View",
              "Message:Create:Route",
              "Message:Template:Create",
              "Message:Draft:Create",
              "Message:Template:Update",
              "Message:Template:Delete",
              "Message:Draft:Update",
              "Message:Draft:View",
              "Message:Draft:Delete",
              "Message:Repair:Route",
              "Message:Repair:RepairStatus",
              "Message:Repair:SendToQueue"
            ],
            "L1": [
              "Message:Template:View",
              "Message:Repair:Edit",
              "Message:Details:View",
```

```
                "Message:Dashboard:Export",
                "Message:Body:View",
                "Message:Intervention:View",
                "Message:Export",
                "Message:Create",
                "Message:Create:Dashboard:View",
                "Message:Create:Route",
                "Message:Template:Create",
                "Message:Draft:Create",
                "Message:Template:Update",
                "Message:Template:Delete",
                "Message:Draft:Update",
                "Message:Draft:View",
                "Message:Draft:Delete",
                "Message:Repair:Route",
                "Message:Repair:RepairStatus",
                "Message:Repair:SendToQueue",
                "Message:Validation:Approve:1",
                "Message:Validation:Reject:1",
                "Message:Create:SendToQueue",
                "Message:Validation:Allow",
                "Message:Validation:Block",
                "Message:Validation:Details:View",
                "Message:Routing:View",
                "Message:Routing:Export",
                "Message:Routing:Import"
              ],
              "L2": [
                "Message:Template:View",
                "Message:Details:View",
                "Message:Dashboard:Export",
                "Message:Body:View",
                "Message:Intervention:View",
                "Message:Export",
                "Message:Create",
                "Message:Create:Dashboard:View",
                "Message:Create:Route",
                "Message:Template:Create",
                "Message:Draft:Create",
                "Message:Template:Update",
                "Message:Template:Delete",
                "Message:Draft:Update",
                "Message:Draft:View",
                "Message:Draft:Delete",
                "Message:Validation:Approve:2",
                "Message:Validation:Reject:2",
                "Message:Create:SendToQueue",
                "Message:Validation:Allow",
                "Message:Validation:Block",
                "Message:Validation:Details:View",
                "Message:Routing:View"
              ],
              "L3": [

                "Message:Template:View",
                "Message:Details:View",
                "Message:Dashboard:Export",
                "Message:Body:View",
```

```
                "Message:Intervention:View",
                "Message:Export",
                "Message:Create",
                "Message:Create:Dashboard:View",
                "Message:Create:Route",
                "Message:Template:Create",
                "Message:Draft:Create",
                "Message:Template:Update",
                "Message:Template:Delete",
                "Message:Draft:Update",
                "Message:Draft:View",
                "Message:Draft:Delete",
                "Message:Validation:Approve:3",
                "Message:Validation:Reject:3",
                "Message:Create:SendToQueue",
                "Message:Validation:Allow",
                "Message:Validation:Block",
                "Message:Validation:Details:View",
                "Message:Routing:View"
            ]
        };

        // Collect scopes for all groups the user belongs to
        let scopesToAdd = [];
        groups.forEach(group => {
            if (groupScopes[group]) {
                scopesToAdd = scopesToAdd.concat(groupScopes[group]);
            }
        });

        // remove duplicates
        scopesToAdd = [...new Set(scopesToAdd)];

          event.response = {
            "claimsAndScopeOverrideDetails": {
              "accessTokenGeneration": {
                "scopesToAdd": scopesToAdd,
                "scopesToSuppress": [
                  "phone_number",
                  "aws.cognito.signin.user.admin"
                ],
                "claimsToAddOrOverride": {
                  "preferred_username": event.request.userAttributes.email
                }
              }
            }
          };
          // Return to Amazon Cognito
          context.done(null, event);
        };

  CognitoLambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
```

```
            Principal:
              Service: lambda.amazonaws.com
            Action: sts:AssumeRole
      Policies:
        - PolicyName: CognitoLambdaExecutionPolicy
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - logs:CreateLogGroup
                  - logs:CreateLogStream
                  - logs:PutLogEvents
                Resource: "*"

  CognitoUserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      UserPoolName: !Ref UserPoolName
      AutoVerifiedAttributes:
        - email
      LambdaConfig:
        PreTokenGeneration: !GetAtt CognitoTokenCustomizerLambda.Arn
        PreTokenGenerationConfig:
          LambdaArn: !GetAtt CognitoTokenCustomizerLambda.Arn
          LambdaVersion: V3_0
      Schema:
        - Name: email
          AttributeDataType: String
          Required: true
          Mutable: true
      Policies:
        PasswordPolicy:
          MinimumLength: 8
          RequireUppercase: true
          RequireLowercase: true
          RequireNumbers: true
          RequireSymbols: true

  CognitoUserPoolClient:
    Type: AWS::Cognito::UserPoolClient
    Properties:
      ClientName: !Ref AppClientName
      UserPoolId: !Ref CognitoUserPool
      GenerateSecret: false
      AllowedOAuthFlows:
        - code
        - implicit
      AllowedOAuthScopes:
        - openid
        - email
        - profile
      AllowedOAuthFlowsUserPoolClient: true
      CallbackURLs:
        - !Sub "https://${CatimUiDomain}/"
      LogoutURLs:
        -  !Sub "https://${CatimUiDomain}/logout"
      SupportedIdentityProviders:
```

```
        - COGNITO

  CognitoGroupAdmin:
    Type: AWS::Cognito::UserPoolGroup
    Properties:
      GroupName: Admin
      UserPoolId: !Ref CognitoUserPool
  CognitoGroupViewer:
    Type: AWS::Cognito::UserPoolGroup
    Properties:
      GroupName: Viewer
      UserPoolId: !Ref CognitoUserPool
  CognitoGroupOperator:
    Type: AWS::Cognito::UserPoolGroup
    Properties:
      GroupName: Operator
      UserPoolId: !Ref CognitoUserPool
  CognitoGroupL1Compliance:
    Type: AWS::Cognito::UserPoolGroup
    Properties:
      GroupName: L1
      UserPoolId: !Ref CognitoUserPool
  CognitoGroupL2Compliance:
    Type: AWS::Cognito::UserPoolGroup
    Properties:
      GroupName: L2
      UserPoolId: !Ref CognitoUserPool
  CognitoGroupL3Compliance:
    Type: AWS::Cognito::UserPoolGroup
    Properties:
      GroupName: L3
      UserPoolId: !Ref CognitoUserPool

  CognitoUserPoolDomain:
    Type: AWS::Cognito::UserPoolDomain
    Properties:
      Domain: !Ref DomainPrefix
      UserPoolId: !Ref CognitoUserPool

  CognitoResourceServer:
    Type: AWS::Cognito::UserPoolResourceServer
    Properties:
      Identifier: message-api
      Name: "Message API Resource Server"
      UserPoolId: !Ref CognitoUserPool
      Scopes:
        - ScopeName: Message:Repair:SendToQueue
          ScopeDescription: Grants access to the Send to button when repairing a message,
allowing users to select a queue.
        - ScopeName: Message:Repair:AnyStatus
          ScopeDescription: Grants access to the repair button on message details on any
status.
        - ScopeName: Settings:MessageApproval:Update
          ScopeDescription: Grants access to edit/change all configs/toggles in the Message
Approval tab, inside Settings.
        - ScopeName: Settings:MessageApproval:View
          ScopeDescription: Grants access to consult all configs/toggles in the Message
Approval tab, inside Settings.
```

```
        - ScopeName: Settings:View
          ScopeDescription: Grants access to the settings dashboard.
        - ScopeName: Message:Routing:View
          ScopeDescription: Grants access to the Routing dashboard.
        - ScopeName: Message:Metadata:View
          ScopeDescription: Grants access to view the message metadata.
        - ScopeName: Message:Routing:Export
          ScopeDescription: Grants access to the Export button on the Routing dashboard.
        - ScopeName: Message:Routing:Import
          ScopeDescription: Grants access to the Import button on the Routing dashboard.
        - ScopeName: Message:Template:View
          ScopeDescription: Grants access to view available PUBLIC templates in the Message templates dashboard, and their details.
        - ScopeName: Message:Repair:Edit
          ScopeDescription: Grants access to edit the Application Header and Body on message repair.
        - ScopeName: Message:Details:View
          ScopeDescription: Grants access to view the message dashboard and details (the first part on the message details tab).
        - ScopeName: Message:Dashboard:Export
          ScopeDescription: Allows access to the export button on the messages dashboard.
        - ScopeName: Message:Body:View
          ScopeDescription: Grants access to view the message body (the second part is on the message details tab).
        - ScopeName: Message:Intervention:View
          ScopeDescription: Grants access to the interventions tab.
        - ScopeName: Message:Validation:Allow
          ScopeDescription: Grants access to the Allow button on the message validations tab.
        - ScopeName: Message:Validation:Block
          ScopeDescription: Grants access to the Block button on the message validations tab.
        - ScopeName: Message:Validation:Details:View
          ScopeDescription: Grants access to view validations dashboard and details (Info).
        - ScopeName: Message:Export
          ScopeDescription: Allows access to the export button on the message details for the body and/or details.
        - ScopeName: Message:Create
          ScopeDescription: Grants access to the Select button on the New FIN/MX message dashboard for creating a message.
        - ScopeName: Message:Create:Dashboard:View
          ScopeDescription: Grants access to visualize the New FIN/MX message dashboard.
        - ScopeName: Message:Create:SendToQueue
          ScopeDescription: Grants access to the Send to button, allowing users to select a queue and send messages for processing.
        - ScopeName: Message:Create:Route
          ScopeDescription: Grants access to the Route button, routing the message through a pre-defined workflow.
        - ScopeName: Message:Template:Create
          ScopeDescription: Grants access to the Template button on message creation for saving the message as a template.
        - ScopeName: Message:Draft:Create
          ScopeDescription: Grants access to the Draft button on message creation for saving the message as a draft.
        - ScopeName: Message:Template:Update
          ScopeDescription: Grants access to the Pen button on the Message templates dashboard to update a template.
        - ScopeName: Message:Template:Delete
```

```
          ScopeDescription: Grants access to the Trash button on the Message templates dashboard to delete a template.
        - ScopeName: Message:Draft:Update
          ScopeDescription: Grants access to the Pen button on the Message drafts dashboard to update a draft.
        - ScopeName: Message:Draft:View
          ScopeDescription: Grants access to view available drafts in the Message drafts dashboard.
        - ScopeName: Message:Draft:Delete
          ScopeDescription: Grants access to the Trash button on the Message drafts dashboard to delete a draft.
        - ScopeName: Message:Validation:Approve:1
          ScopeDescription: Provides access to the Approve button on the message validations tab, allowing decisions to be made at the First level.
        - ScopeName: Message:Validation:Reject:1
          ScopeDescription: Provides access to the Reject button on the message validations tab, allowing decisions to be made at the First level.
        - ScopeName: Message:Validation:Approve:2
          ScopeDescription: Provides access to the Approve button on the message validations tab, allowing decisions to be made at the Second level.
        - ScopeName: Message:Validation:Reject:2
          ScopeDescription: Provides access to the Reject button on the message validations tab, allowing decisions to be made at the Second level.
        - ScopeName: Message:Validation:Approve:3
          ScopeDescription: Provides access to the Approve button on the message validations tab, allowing decisions to be made at the Third level.
        - ScopeName: Message:Validation:Reject:3
          ScopeDescription: Provides access to the Reject button on the message validations tab, allowing decisions to be made at the Third level.
        - ScopeName: Message:Repair:Route
          ScopeDescription: Grants access to the Route button when repairing a message, routing the message through a pre-defined workflow.
        - ScopeName: Message:Repair:Metadata:Update
          ScopeDescription: Grants access to edit the message metadata during the repair.
        - ScopeName: Message:Repair:RepairStatus
          ScopeDescription: Grants access to the repair button on message details when messages are on the Repair status.

  LambdaInvokePermission:
    Type: AWS::Lambda::Permission
    Properties:
      Action: lambda:InvokeFunction
      FunctionName: !GetAtt CognitoTokenCustomizerLambda.Arn
      Principal: cognito-idp.amazonaws.com
      SourceArn: !Sub arn:${AWS::Partition}:cognito-idp:${AWS::Region}:${AWS::AccountId}:userpool/${CognitoUserPool}

Outputs:
  UserPoolId:
    Description: Cognito User Pool ID
    Value: !Ref CognitoUserPool

  UserPoolClientId:
    Description: App Client ID
    Value: !Ref CognitoUserPoolClient

  UserPoolDomain:
    Description: Cognito Hosted UI domain URL
```

```
    Value: !Sub "https://${DomainPrefix}.auth.${AWS::Region}.amazoncognito.com"

  ResourceServerIdentifier:
    Description: Resource Server Identifier
    Value: !Ref CognitoResourceServer
```

### 3.2 Apply Cloudformation stack

With the use of AWS CLI, deploy the Cloudformation stack, ensuring you replace placeholders with relevant values:

> Replace placeholder for CatimUiDomain, ensuring it is the URL you will associated with the CATIM Frontend

```
aws cloudformation create-stack \
  --stack-name CognitoStack \
  --template-body file://cognito-stack.yaml \
  --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND \
  --parameters \
    ParameterKey=CatimUiDomain,ParameterValue={REPLACEME}
```

### 3.3 Create Cognito user

1. Login to the AWS console and navigate to the Cognito User Pool you just created
2. Under *User Management > Users,* click *Create User*
3. Set a password (you will be asked to change it upon your first login)
4. After user creation, click on the user you created
5. Edit *User attributes* and add the *Optional attribute* ***preferred_username*** set to the user email

### 3.4 Add Cognito user to Admin group

1. In your Cognito user pool, navigate to *User Management > Groups > Admin*
2. In *Group Members,* add the new user you just created

## 4. Install & Configure Temporal

1. Add the temporalio Helm repo:

```
helm repo add temporalio https://temporalio.github.io/helm-charts
```

2. Add variables to your bash session, replacing placeholders:

```
WEB_URL=temporal.{REPLACEME} # the Ingress for your Temporal-web service
DB_HOST={REPLACEME} # The hostname for your Postgres RDS
DB_USERNAME={REPLACEME} # The RDS username
DB_PASSWORD={REPLACEME} # The RDS user password
```

3. Apply the Temporal helm values:

```
helm install -n catim --create-namespace temporal temporalio/temporal --version 0.62.0 -f
- <<EOF
web:
  image:
    repository: 709825985650.dkr.ecr.us-east-1.amazonaws.com/intellecteu/catim
    tag: temporalio-ui-2.37.1
    pullPolicy: IfNotPresent
  ingress:
    enabled: true
    hosts:
      - $WEB_URL
  additionalEnv:
    - name: TEMPORAL_CSRF_COOKIE_INSECURE
      value: "True"
server:
  image:
    repository: 709825985650.dkr.ecr.us-east-1.amazonaws.com/intellecteu/catim
    tag: temporalio-server-1.27.2
    pullPolicy: IfNotPresent
  internalFrontend:
    enabled: true
  config:
    namespaces:
      create: true
      namespace:
        - name: default
          retention: 3d
    persistence:
      default:
        driver: sql
        sql:
          driver: postgres12
          host: $DB_HOST
          port: 5432
          database: temporal
          user: $DB_USERNAME
          password: $DB_PASSWORD
          maxConns: 20
          maxConnLifetime: 1h
          tls:
            enabled: true
            enableHostVerification: false
      visibility:
        driver: sql
        sql:
          driver: postgres12
          host: $DB_HOST
          port: 5432
          database: temporal_visibility
          user: $DB_USERNAME
          password: $DB_PASSWORD
          maxConns: 20
          maxConnLifetime: 1h
          tls:
            enabled: true
            enableHostVerification: false
admintools:
  image:
```

```
    repository: 709825985650.dkr.ecr.us-east-1.amazonaws.com/intellecteu/catim
    tag: temporalio-admintools-1.28.1-tctl-1.18.4-cli-1.4.1
    pullPolicy: IfNotPresent
cassandra:
  enabled: false
mysql:
  enabled: false
postgresql:
  enabled: false
prometheus:
  enabled: false
grafana:
  enabled: false
elasticsearch:
  enabled: false
EOF
```

## 5. Install & Configure CAT-IM

1. Add variables to your bash session, replacing placeholders:

```
DOMAIN={REPLACEME} # the base custom domain/subdomain for your EKS cluster
DB_HOSTNAME={REPLACEME} # the RDS Postgresql DNS hostname
DB_USERNAME={REPLACEME} # The RDS Postgresql username
DB_PASSWORD={REPLACEME} # The RDS Postgresql password
OIDC_CLIENTID={REPLACEME} # The Cognito App Client ID
OIDC_AUTHORITY_URL=https://cognito-idp.{REPLACEME_AWSREGION}.amazonaws.com/{REPLACEME_USERPOOLID}
OIDC_AUTH_URL=https://{REPLACEME_COGNITODOMAINNAME}.auth.{REPLACEME_AWSREGION}.amazoncognito.com/oauth2/authorize
SFTP_HOSTNAME={REPLACEME}
SFTP_USERNAME={REPLACEME}
SFTP_PASSWORD={REPLACEME}
INGRESS_CLASS={REPLACEME}
```

2. Login to ECR registry:

```
aws ecr get-login-password \
    --region us-east-1 | helm registry login \
    --username AWS \
    --password-stdin 709825985650.dkr.ecr.us-east-1.amazonaws.com
```

3. Apply the CAT-IM helm values:

```
helm upgrade --install catim oci://709825985650.dkr.ecr.us-east-1.amazonaws.com/intellecteu/catim --namespace catim --version 4.0.12 \
    --set global.baseHostname=$DOMAIN \
    --set global.postgresql.auth.hostname=$DB_HOSTNAME \
    --set global.postgresql.auth.username=$DB_USERNAME \
    --set global.postgresql.auth.password=$DB_PASSWORD \
    --set global.oidc.clientId=$OIDC_CLIENTID \
    --set global.oidc.authorityUrl=$OIDC_AUTHORITY_URL \
    --set global.oidc.authUrl=$OIDC_AUTH_URL \
    --set global.sftp.hostname=$SFTP_HOSTNAME \
    --set global.sftp.username=$SFTP_USERNAME \
    --set global.sftp.password=$SFTP_PASSWORD \
```

```
12      --set global.ingressClassName=$INGRESS_CLASS
```