AWS MARKETPLACE · CONTAINER PRODUCT

i2i Intelligence Platform Installation Guide

Step-by-step instructions for deploying the i2i platform into your own AWS account using the CloudFormation template included with your Marketplace subscription. Complete every prerequisite before clicking **Launch Stack**.

⏱ **Estimated time:** 30–60 minutes (one-time setup) 🚀 **Stack deploy time:** 5–10 minutes ECS Fargate

CloudFormation

🌐 **Supported regions:** us-east-1 · us-east-2 · us-west-2 · eu-west-1 · eu-central-1 · ap-southeast-1 · ap-northeast-1

🤖 **AI model:** Claude Sonnet 4.5 via Amazon Bedrock

CONTENTS

1 AWS Account & Region REQUIRED

Choose your deployment region and verify IAM permissions

Choose a region

All resources — VPC, subnets, S3 bucket, ACM certificate, Secrets Manager secrets, ECS cluster, and CloudWatch logs — must reside in the **same AWS region**. Mixing regions will cause the deployment to fail.

Geography	Supported Region IDs
United States	`us-east-1` `us-east-2` `us-west-2`
Europe	`eu-west-1` `eu-central-1`
Asia Pacific	`ap-southeast-1` `ap-northeast-1`

⚠️ **us-west-1 is not supported.** Select the nearest available region from the list above. Throughout this guide, your chosen region is referred to as `<YOUR-REGION>`.

Required IAM permissions

The IAM user or role used to launch the stack requires the following permissions. The simplest option is to attach the **AdministratorAccess** managed policy during initial deployment.

Service	Actions required
CloudFormation	`cloudformation:*`
IAM	`iam:CreateRole` · `iam:PutRolePolicy` · `iam:AttachRolePolicy` · `iam:PassRole`
EC2	`ec2:CreateSecurityGroup` · `ec2:AuthorizeSecurityGroupIngress`
ECS	`ecs:*`
Elastic Load Balancing	`elasticloadbalancing:*`
CloudWatch Logs	`logs:CreateLogGroup`
Secrets Manager	`secretsmanager:GetSecretValue`

2 Networking REQUIRED

VPC and subnets for ALB, ECS tasks, and MWAA

You need a single VPC containing both public subnets (for the ALB and ECS Fargate tasks) and private subnets (for Amazon MWAA). ECS tasks, although launched in public subnets, are protected by a security group that only permits traffic from the ALB.

VPC requirements

- [ ] VPC in `<YOUR-REGION>` with **DNS resolution** enabled (VPC → Actions → Edit DNS resolution → enable)

- [ ] VPC with **DNS hostnames** enabled (VPC → Actions → Edit DNS hostnames → enable)
- [ ] Note the **VPC ID** (format: `vpc-xxxxxxxxxxxxxxxxx`)
- [ ] At least **two public subnets** in different Availability Zones — used by the ALB and ECS tasks. Each must have an Internet Gateway route in its route table.
- [ ] At least **two private subnets** in different Availability Zones — used by MWAA workers. These require a NAT Gateway route for outbound internet access.
- [ ] Note all **subnet IDs** (format: `subnet-xxxx`, `subnet-yyyy`, ...)

> 💡 **Creating from scratch?** Go to AWS Console → VPC → *Create VPC* → choose *VPC and more*. Select 2 AZs, 2 public subnets, 2 private subnets, and 1 NAT Gateway (1 AZ is sufficient for evaluation).

---

3 Security Groups REQUIRED

Must be pre-created before launching the CloudFormation stack

Create the two security groups below and supply their IDs as CloudFormation parameters. This gives you explicit control over which IP addresses and services can reach the platform.

Security Group 1 — ALB `i2i-alb-sg`

Direction	Protocol	Port	Source / Destination	Purpose
Inbound	TCP	80	*Your users' IP CIDR*	HTTP access
Inbound	TCP	443	*Your users' IP CIDR*	HTTPS access
Outbound	All	All	0.0.0.0/0	Required to reach ECS tasks

Replace *your users' IP CIDR* with your office/VPN CIDR, e.g. `203.0.113.0/24`. Use `0.0.0.0/0` only for public internet access.

Security Group 2 — ECS Tasks `i2i-ecs-sg`

Direction	Protocol	Port	Source / Destination	Purpose
Inbound	TCP	80	ALB security group ID	ALB to UI container
Inbound	All	All	This security group (self)	Inter-service traffic
Outbound	All	All	0.0.0.0/0	Calls to Cube Cloud, Bedrock, S3

⚠️ For the ALB inbound rule on the ECS SG, specify the **security group ID** (sg-xxx) of the ALB SG — not a CIDR block.

CLI — create both security groups

BASH

```
# ── Step 1: Create ALB security group ──────────────────────
ALB_SG=$(aws ec2 create-security-group \
  --group-name i2i-alb-sg \
  --description "i2i ALB - inbound HTTP and HTTPS from users" \
  --vpc-id <your-vpc-id> \
  --region <YOUR-REGION> \
  --query GroupId --output text)

# Allow HTTP from your users
aws ec2 authorize-security-group-ingress \
  --group-id $ALB_SG --protocol tcp --port 80 \
  --cidr <your-cidr> --region <YOUR-REGION>

# Allow HTTPS from your users
aws ec2 authorize-security-group-ingress \
  --group-id $ALB_SG --protocol tcp --port 443 \
  --cidr <your-cidr> --region <YOUR-REGION>

echo "ALB SG ID: $ALB_SG"    # ← note this value

# ── Step 2: Create ECS tasks security group ────────────────
ECS_SG=$(aws ec2 create-security-group \
  --group-name i2i-ecs-sg \
  --description "i2i ECS tasks - inbound from ALB and inter-service" \
  --vpc-id <your-vpc-id> \
  --region <YOUR-REGION> \
  --query GroupId --output text)

# Allow ALB → ECS on port 80
aws ec2 authorize-security-group-ingress \
  --group-id $ECS_SG --protocol tcp --port 80 \
  --source-group $ALB_SG --region <YOUR-REGION>

# Allow ECS ↔ ECS (all ports, self-reference for inter-service calls)
aws ec2 authorize-security-group-ingress \
  --group-id $ECS_SG --protocol -1 \
  --source-group $ECS_SG --region <YOUR-REGION>

echo "ECS SG ID: $ECS_SG"    # ← note this value
```

- [ ] Note ALB Security Group ID → `ALBSecurityGroupId` CloudFormation parameter

- [ ] Note ECS Security Group ID → `ECSSecurityGroupId` CloudFormation parameter

---

4 TLS Certificate OPTIONAL

Strongly recommended for production; leave blank for HTTP-only evaluation

When provided, the ALB automatically redirects all HTTP traffic to HTTPS. If this parameter is left blank, the platform runs on HTTP only.

1 Request a public certificate in ACM

AWS Console → Certificate Manager → *Request* → *Request public certificate*.
Enter your domain (e.g. `i2i.yourcompany.com` or `*.yourcompany.com`). Choose **DNS validation** and add the supplied CNAME record to your DNS. Wait for status to show **Issued**.

2 Note the certificate ARN

Format: `arn:aws:acm:<YOUR-REGION>:<account-id>:certificate/<uuid>`

> 🚫 **Important:** ACM certificates are region-specific. The certificate *must* be requested in the same region as your deployment. Certificates from other regions cannot be attached to the ALB.

---

5 S3 Bucket REQUIRED

Stores all pipeline run outputs — charts, reports, metadata

i2i writes all pipeline results to a dedicated S3 bucket in your account. The bucket starts empty; the platform automatically creates the required folder structure and JSON schema files on first deployment.

```bash
# For all regions except us-east-1
aws s3api create-bucket \
  --bucket <your-unique-bucket-name> \
  --region <YOUR-REGION> \
  --create-bucket-configuration LocationConstraint=<YOUR-REGION>

# For us-east-1 only (omit --create-bucket-configuration)
aws s3api create-bucket \
  --bucket <your-unique-bucket-name> \
  --region us-east-1
```

- [ ] Bucket created in `<YOUR-REGION>`

- [ ] **Block Public Access** — all four settings enabled (this is the default; do not disable)
- [ ] Note the bucket name → `S3BucketName` CloudFormation parameter

> ℹ️ Do **not** upload anything to this bucket before launching the stack. The one-time setup task generates `metadata.json` and `new_test.json` automatically during deployment.

---

6 AWS Secrets Manager REQUIRED

Two secrets must be created — credentials are never stored in CloudFormation parameters

Secret 1 — Cube.js API Secret Token

```bash
aws secretsmanager create-secret \
  --region <YOUR-REGION> \
  --name "i2i/cubejs-api-secret" \
  --secret-string "<your-cubejs-api-secret-token>"
```

Secret 2 — Cube DB Password

```bash
aws secretsmanager create-secret \
  --region <YOUR-REGION> \
  --name "i2i/cube-db-password" \
  --secret-string "<your-cube-db-password>"
```

> ⚠️ Copy the full ARN from each command's response — **including the random suffix**. Example format: `arn:aws:secretsmanager:us-west-2:123456789012:secret:i2i/cubejs-api-secret-AbCdEf`. The ARN with suffix is required as the CloudFormation parameter value.

- [ ] Note secret ARN for `i2i/cubejs-api-secret` → `CubeJsApiSecretArn` parameter
- [ ] Note secret ARN for `i2i/cube-db-password` → `CubeDbPasswordSecretArn` parameter

---

7 Cube Cloud Connection Details REQUIRED

Collect from the Cube Cloud console before launching the stack

i2i connects to your Cube Cloud deployment as its semantic layer via both the REST API and the SQL API. All four values below are required CloudFormation parameters.

CloudFormation Parameter	Where to find it	Example value
`CubeApiUrl`	Cube Cloud → Deployment → Overview → *API URL*	`https://<deployment>.gcp-us-central1.cubecloudapp.dev/cubejs-api/v1`
`CubeDbName`	Subdomain part of your Cube deployment URL	`grateful-tahr`
`CubeDbUser`	Cube Cloud → Deployment → SQL API credentials	`cube` (default)
`CubeDbHost`	Cube Cloud → Deployment → Overview → *SQL API endpoint*	`<deployment>.sql.gcp-us-central1.cubecloudapp.dev`

ℹ The Cube.js API secret token and Cube DB password are supplied via Secrets Manager ARNs from Step 6 — not entered directly as parameters.

---

8 Amazon MWAA REQUIRED

Decide the environment name now — create the environment after CloudFormation completes

🚫 **Do not create the MWAA environment yet.** Only decide and record the name you intend to use (e.g. `i2i-prod`). The environment must be created *after* the CloudFormation stack reaches `CREATE_COMPLETE` — you will need output values from the stack when configuring it.

What to prepare now

- [ ] Decide the MWAA environment name (e.g. `i2i-prod`) → `MwaaEnvName` CloudFormation parameter
- [ ] Ensure private subnets from Step 2 are in the same VPC — MWAA will use them
- [ ] Have your MWAA DAGs S3 bucket ready (a separate bucket from the i2i working bucket in Step 5)

S3 bucket separation — important

Bucket	Purpose	Contents
MWAA DAGs bucket	Airflow file storage	`startup.sh`, `dags/airflow_backend.py`
i2i working bucket	Pipeline run outputs	Charts, EDA results, reports, `metadata.json`

MWAA execution role permissions

After creating the MWAA environment, add the following inline policy to its execution role so MWAA workers can read and write the i2i working bucket:

```
{
  "Effect": "Allow",
  "Action": ["s3:GetObject", "s3:PutObject", "s3:ListBucket"],
  "Resource": [
    "arn:aws:s3:::<your-i2i-working-bucket-name>",
    "arn:aws:s3:::<your-i2i-working-bucket-name>/*"
  ]
}
```

JSON

> When you create the MWAA environment later, place it in the **same VPC and region** as the ECS stack, using the **private subnets**. Enable **all log types at INFO level**. Leave the `MwaaRegion` CloudFormation parameter blank if MWAA is in the same region as the stack.

---

9 Amazon Bedrock Model Access REQUIRED

Access must be granted before the platform can process any request

Request access to the following model in `<YOUR-REGION>` :

```
us.anthropic.claude-sonnet-4-5-20250929-v1:0
```

MODEL ID

1. **Open the Bedrock console**

   AWS Console → **Amazon Bedrock** → *Model access* (left sidebar)

2. **Request access**

   Click *Manage model access* → locate **Anthropic / Claude Sonnet 4.5** → check the box → click *Request model access* → *Submit*.

3. **Wait for approval**

   Refresh until the model status shows **Access granted**. Anthropic models are approved within minutes in most regions.

> ⚠️ Bedrock model availability varies by region. If Claude Sonnet 4.5 is not listed in your chosen region, select the nearest supported region or contact i2i support for an alternative model ID.

---

10 Launch the CloudFormation Stack

Subscribe on AWS Marketplace, then fill in the parameter values collected above

CloudFormation parameters reference

Parameter	Value	Source
`VpcId`	`vpc-xxxxxxxxxxxxxxxxx`	Step 2
`PublicSubnetIds`	`subnet-xxxx,subnet-yyyy`	Step 2
`ALBSecurityGroupId`	`sg-xxxxxxxxxxxxxxxxx`	Step 3
`ECSSecurityGroupId`	`sg-xxxxxxxxxxxxxxxxx`	Step 3
`ACMCertificateArn`	`arn:aws:acm:…` or leave blank	Step 4
`S3BucketName`	Your bucket name	Step 5
`CubeJsApiSecretArn`	`arn:aws:secretsmanager:…`	Step 6
`CubeDbPasswordSecretArn`	`arn:aws:secretsmanager:…`	Step 6
`CubeApiUrl`	`https://….cubecloudapp.dev/cubejs-api/v1`	Step 7
`CubeDbName`	Deployment subdomain	Step 7
`CubeDbUser`	`cube`	Step 7
`CubeDbHost`	`….sql.….cubecloudapp.dev`	Step 7
`MwaaEnvName`	Your chosen MWAA environment name	Step 8
`MwaaRegion`	Leave blank if same region as stack	Step 8

Launch steps

1 Subscribe and open CloudFormation

After subscribing on AWS Marketplace, click **Launch CloudFormation**. You will land on the CloudFormation *Create Stack* wizard with the template pre-loaded.

2 Fill in parameters

Enter all values from the table above. Leave all parameters not listed above at their default values unless advised by i2i support.

3 Acknowledge IAM capabilities

On the final review page, check "*I acknowledge that AWS CloudFormation might create IAM resources with custom names*" and click **Create Stack**.

4 Wait for CREATE_COMPLETE

Stack deployment takes 5–10 minutes. During this time the one-time setup task runs, generating `metadata.json` and `new_test.json` in your S3 bucket. If the setup task fails, the stack rolls back with a clear error message in the Events tab.

5 Copy the stack outputs

Navigate to the **Outputs** tab and record the following values — you will need them in the next step:

- `ApplicationURL` — public URL for the i2i platform
- `I2IApiUrl` — internal Cloud Map DNS for the analytics engine
- `UtilsApiUrl` — internal Cloud Map DNS for the utils service

---

11 Post-Deployment: Configure MWAA

Complete these steps after the stack reaches CREATE_COMPLETE

✅ You now have the `ApplicationURL`, `I2IApiUrl`, and `UtilsApiUrl` from the CloudFormation Outputs tab. Keep them open — you will need them below.

a) Create and upload the startup script

Create a file named `startup.sh` using the template below. The two Cloud Map DNS values are fixed — do not change them. Set `S3_BUCKET_NAME` and `REGION_NAME` to your own values.

STARTUP.SH

```
#!/bin/bash
# i2i Intelligence Platform - MWAA Startup Script
# Upload to the ROOT of your MWAA DAGs bucket (not inside dags/).
# Then set the Startup script file path in the MWAA environment settings.

# Internal Cloud Map DNS names - DO NOT change these values
export UTILS_API_URL="http://utils.i2i.local:8950/"
export I2I_API_URL="http://i2i-svc.i2i.local:8085/"

# Set your deployment-specific values here
export S3_BUCKET_NAME="<your-i2i-working-bucket-name>"
export REGION_NAME="<YOUR-REGION>"
```

Upload the file to the **root** of your MWAA DAGs bucket:

```bash
aws s3 cp startup.sh s3://<your-mwaa-dags-bucket>/startup.sh
```

b) Create the MWAA environment

Create the Amazon MWAA environment using the name you recorded in Step 8. Set the **Startup script file path** to `s3://<your-mwaa-dags-bucket>/startup.sh` during the creation wizard — the environment will inject the variables automatically on every worker start.

- [ ] Use the **same name** entered as the `MwaaEnvName` CloudFormation parameter
- [ ] Place it in the **same VPC and region** as the ECS stack
- [ ] Use the **private subnets** from Step 2
- [ ] Configure the **DAGs S3 bucket** to the MWAA DAGs bucket (not the i2i working bucket)
- [ ] Enable **all log types at INFO level**

c) Allow MWAA → ECS traffic

After the MWAA environment is created and reaches **Available** state, add inbound rules to the ECS security group to permit MWAA workers to call the i2i services:

```bash
# Retrieve the MWAA environment security group ID
MWAA_SG=$(aws mwaa get-environment \
  --name <your-mwaa-env-name> \
  --region <YOUR-REGION> \
  --query "Environment.NetworkConfiguration.SecurityGroupIds[0]" \
  --output text)

# Allow MWAA workers → i2i analytics engine (port 8085)
aws ec2 authorize-security-group-ingress \
  --group-id <your-ecs-sg-id> \
  --protocol tcp --port 8085 \
  --source-group $MWAA_SG --region <YOUR-REGION>

# Allow MWAA workers → utils service (port 8950)
aws ec2 authorize-security-group-ingress \
  --group-id <your-ecs-sg-id> \
  --protocol tcp --port 8950 \
  --source-group $MWAA_SG --region <YOUR-REGION>
```

d) Upload the DAG file

```bash
aws s3 cp airflow_backend.py \
  s3://<your-mwaa-dags-bucket>/dags/airflow_backend.py
```

After 30–60 seconds, the `i2i_pipeline` DAG will appear as **Active** in the Airflow UI. If it shows a parse error, verify that all four values in `startup.sh` are correct and that the MWAA environment has finished restarting.

e) Verify Airflow environment variables

Confirm all four variables are present in: MWAA Console → your environment → Edit → *Airflow configuration options*

Key	Expected value	Set by
`S3_BUCKET_NAME`	Your i2i working bucket name	Your input in startup.sh
`REGION_NAME`	Your deployment region	Your input in startup.sh
`I2I_API_URL`	`http://i2i-svc.i2i.local:8085/`	Injected by startup.sh
`UTILS_API_URL`	`http://utils.i2i.local:8950/`	Injected by startup.sh

Deployment checklist

- [ ] `startup.sh` uploaded to `s3://<mwaa-dags-bucket>/startup.sh`
- [ ] Startup script file path configured in MWAA environment settings
- [ ] MWAA environment created with correct VPC, private subnets, and log levels
- [ ] ECS security group updated with MWAA inbound rules on ports 8085 and 8950
- [ ] `airflow_backend.py` uploaded to `s3://<mwaa-dags-bucket>/dags/`
- [ ] `i2i_pipeline` DAG shows as **Active** in the Airflow UI
- [ ] Open `ApplicationURL` in a browser — the i2i platform is ready

---

12 Attaching a Custom Domain OPTIONAL

Map a vanity domain to the ALB's auto-generated DNS name

Immediately after deployment the platform is reachable via the ALB's auto-generated DNS name (e.g. `i2i-platform-alb-123456789.<YOUR-REGION>.elb.amazonaws.com`). To serve it under your own subdomain:

Option A — Amazon Route 53 (recommended)

1 **Open your hosted zone**

Route 53 → Hosted zones → your domain

2 **Create an alias A record**

- Record name: your chosen subdomain (e.g. `i2i`)
- Record type: **A**
- Alias: **Yes** → Alias to Application and Classic Load Balancer
- Region: `<YOUR-REGION>`
- Load balancer: select `i2i-platform-alb-…`

3 **Save**

DNS propagates within 60 seconds for Route 53 alias records.

Option B — External DNS provider

Field	Value
Record type	CNAME
Name	Your chosen subdomain, e.g. `i2i`
Value	`ALBDNSName` from CloudFormation Outputs
TTL	300

⚠️ If you provided an ACM certificate in Step 4, the domain in the certificate must exactly match the subdomain you are pointing at the ALB, otherwise browsers will display a certificate mismatch error.

13

Managing Your Data in S3

S3 folder layout and routine maintenance tasks

Folder layout written by the platform

```
<your-i2i-bucket>/ ├── metadata.json ← Schema of your Cube data model (auto-generated on first deploy) ├── new_test.json ← Semantic layer metadata shown in the UI sidebar ├── qa_results.json ← Cumulative Q&A history across all sessions └── <run_id>/ ← One folder per pipeline run ├── eda/ ← Working files for the EDA stage ├── hypothesis/ ← Working files for the hypothesis stage ├── plot/ ← Working files for the chart planning stage └── outputs/ ├── eda_result_list.json ├── hypothesis_list.json ├── hypothesis_result_list.json ├── question_list.json ├── final_plot_list.json ├── insight_des.json ├── report.json └── summary.json
```

Routine maintenance

Refresh schema after Cube model changes

Re-run the CloudFormation stack update (or re-run `deploy.sh`). The one-time setup task overwrites `metadata.json` and `new_test.json` with the latest schema from your Cube deployment.

Remove old run data

```bash
# Delete a single run folder
aws s3 rm s3://<your-bucket>/<run-id>/ --recursive

# Or set an S3 Lifecycle rule to expire run folders after N days
# S3 Console → your bucket → Management → Lifecycle rules → Create rule
```

Inspect results manually

```bash
aws s3 cp s3://<your-bucket>/<run-id>/outputs/summary.json ./summary.json
```

Enable versioning for point-in-time recovery

```bash
aws s3api put-bucket-versioning \
  --bucket <your-bucket> \
  --versioning-configuration Status=Enabled
```