Industrial IoT Anomaly Detection — Operations Cookbook

Version: 1.0

Audience: DevOps, SRE, Cloud Operations Engineers

Table of Contents

Overview

This cookbook documents deployment, operation, monitoring and disaster recovery for the Industrial IoT Anomaly Detection platform. The system processes telemetry from edge devices, detects anomalies via ML models, and issues alerts. It covers both Primary AWS Region deployments and DR region replication.

Architecture

The architecture spans AWS IoT Core, SiteWise, Kinesis, DynamoDB, S3 Data Lake, SageMaker, SNS, and QuickSight with replication to a DR region.

Control machine runs the CI/CD tools (Rancher, ArgoCD, Semaphore) in Docker containers. This will be a local machine.

Replication uses S3 Cross-Region Replication and DynamoDB Global Tables.

Greengrass devices can dual-publish to Primary and DR endpoints.

IoT Core based things can also be the remote devices that send data directly to the IoT Core

We adopt CLI-First-Approach where all activities can be performed via commands. UI tools are for manual control and approvals.

Ansible Templates & playbooks are stored in a Git repository and cloned on control machine locally for deploying, updating or deleting system's architectural components on the cloud.

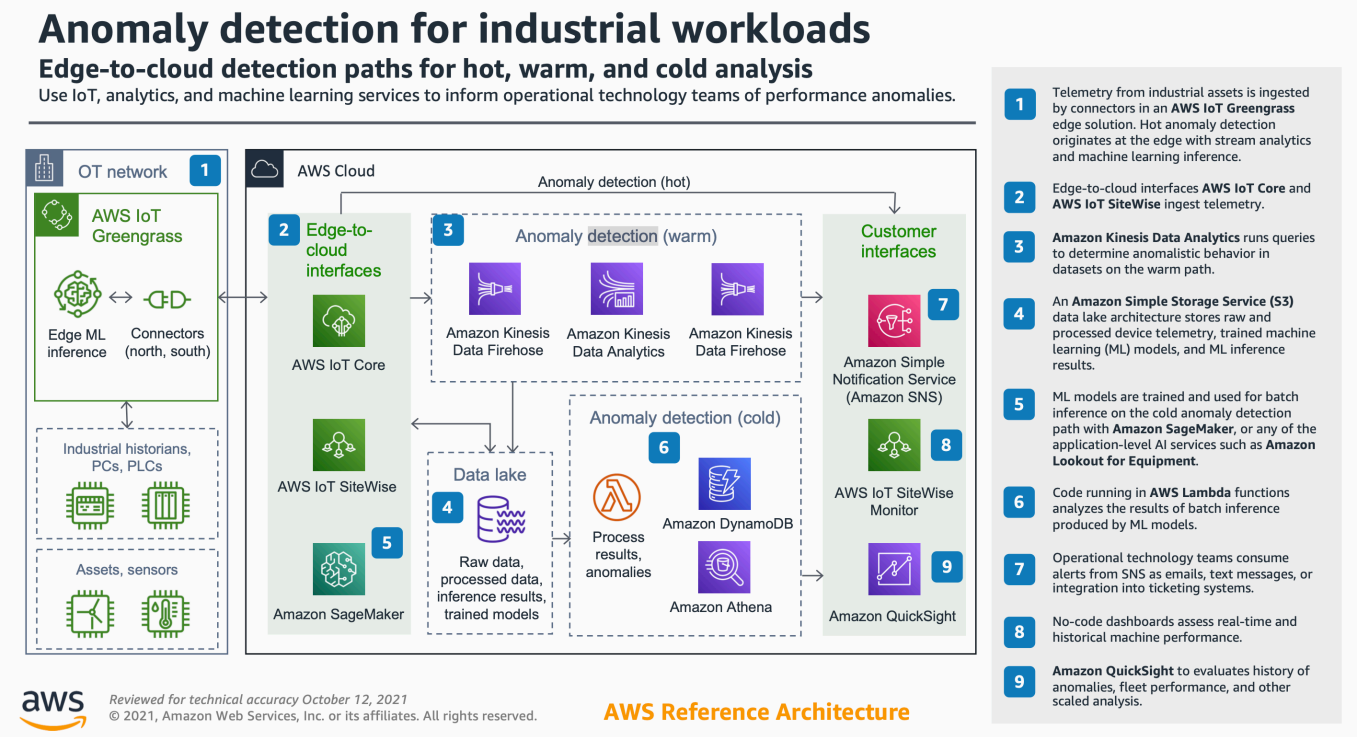
Reference Architecture

The Industrial IoT Anomaly Detection platform is built on AWS services for ingestion, processing, analytics, and alerting, with local DevOps tools for controlled deployment.

Components of Architecture

Layer	Service / Tool	Purpose
Smart Edge	IoT Greengrass	Runs local inference and forwards telemetry to AWS IoT Core.
Edge	IoT Core Thing Device	Sends raw telemetry data to AWS IoT Core
Ingestion	AWS IoT Core	Secure MQTT broker, receives device telemetry.
Data Routing	IoT Rules	Routes messages to Kinesis, SiteWise, S3, and SNS.
Time-series Data	IoT SiteWise	Models and stores asset telemetry for dashboards.
Stream Processing	Kinesis Data Streams / Analytics	Real-time processing, anomaly score computation.
Storage	S3 Data Lake	Raw + processed telemetry storage.
Storage	DynamoDB	Stores anomaly detection results and device states.
ML Inference	SageMaker Endpoint	Hosts trained anomaly detection models.
Alerting	SNS	Pushes anomaly alerts to operators via email/SMS/Slack.
Analytics	QuickSight	Business intelligence dashboards for trends and alerts.
Deployment Control Machine	Rancher, ArgoCD, Semaphore (local)	Manages Kubernetes workloads, GitOps pipelines, and Ansible execution.
Backup	S3 Backup Buckets	Stores point-in-time backups of data and configs.
DR Replication	S3 Cross-Region Replication, DynamoDB Global Tables	Keeps DR environment in sync with primary.

Architecture Diagram



As seen in the diagram -

1. Telemetry from industrial assets is ingested by connectors in an AWS IoT Greengrass edge solution. Hot anomaly detection originates at the edge with stream analytics and machine learning inference.

2. Edge-to-cloud interfaces AWS IoT Core and AWS IoT SiteWise ingest telemetry.

3. Amazon Kinesis Data Analytics runs queries to determine anomalistic behavior in datasets on the warm path.

4. An Amazon Simple Storage Service (S3) data lake architecture stores raw and processed device telemetry, trained machine learning (ML) models, and ML inference results.

5. ML models are trained and used for batch inference on the cold anomaly detection path with Amazon SageMaker, or any of the application-level AI services such as Amazon Lookout for Equipment.

6. Code running in AWS Lambda functions analyzes the results of batch inference produced by ML models.

7. Operational technology teams consume alerts from SNS as emails, text messages, or integration into ticketing systems.

8. No-code dashboards assess real-time and historical machine performance.

9. Amazon QuickSight to evaluates history of anomalies, fleet performance, and other scaled analysis.

10. At high-Level this system has - Overview

- **Primary Region on AWS** — Hosts the production workload.
- **DR Region on AWS** — Mirrors the primary setup for failover.
- **Control Machine on local machine** — Runs Rancher, ArgoCD, and Ansible Semaphore as docker containers for controlled deployments.

Primary–DR Relationship

- **S3 Buckets**: Cross-region replication keeps DR data lake updated.
- **DynamoDB**: Global table replication ensures low-latency access in both regions.
- **SageMaker Models**: Synced to DR region via Ansible playbooks.
- **IoT Resources**: DR has matching Things, Certificates, and Policies for failover.
- **Kinesis Streams**: DR mirrors primary processing pipelines for zero data loss.

Local DevOps Control

- **Rancher**: UI for monitoring Kubernetes workloads (if used for data processing microservices).
- **ArgoCD**: GitOps tool for syncing infrastructure manifests from Git to Kubernetes clusters.
- **Ansible Semaphore**: Runs predefined playbooks for AWS resource setup, certificate rollout, Greengrass config updates, and failover procedures.

Deployment Flow

1. **Clone Git Repo** — Contains all IaC (Infrastructure as Code) playbooks and manifests.
2. Run Ansible Playbooks via CLI or Semaphore — Sets up Primary & DR environments.
3. **Edge Devices Connect** — IoT Core receives telemetry.

4. **Processing & Storage** — Kinesis, S3, DynamoDB store and process data.
5. **Model Inference** — SageMaker endpoints detect anomalies.
6. **Alerts & Dashboards** — SNS alerts operators, QuickSight visualizes trends.
7. **Replication to DR** — Continuous sync of data and configs.
8. **Failover** — DR becomes active in case of Primary outage.

Prerequisites

Control Machine Setup

Below setups are needed on the control machine.

```
# 1. cli script for installing  aws
        brew install awscli
# 2.  cli script for installing Ansible
        brew install ansible
# 3. cli script for installing Docker
        brew install --cask docker && open -a Docker
        docker --version
# 4. Rancher, ArgoCD, Semaphore in Docker
        # For rancher use the command -
        docker run -d --name rancher --restart=unless-stopped \
                -p 80:80 -p 443:443 \
                 -v /opt/rancher:/var/lib/rancher \
                --privileged \
                rancher/rancher:latest
        # For argo cd use the command -
                # a) Enable Kubernetes in Docker Desktop
                # b) Install Argo CD into the cluster
                        kubectl create namespace argocd
```

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

c) Port-forward the API/UI to localhost:8081

kubectl port-forward svc/argocd-server -n argocd 8081:443

d) Get the initial admin password

argocd admin initial-password -n argocd

For semaphore ui use the command -

docker run -d --name semaphore -p 3000:3000 \

-e SEMAPHORE_DB_DIALECT=bolt \

-e SEMAPHORE_ADMIN=admin \

-e SEMAPHORE_ADMIN_PASSWORD=changeme \

-e SEMAPHORE_ADMIN_NAME="Admin" \

-e SEMAPHORE_ADMIN_EMAIL=admin@localhost \

semaphoreui/semaphore:latest

-v semaphore-data:/var/lib/semaphore \

-v semaphore-config:/etc/semaphore

#5 For running native Ansible module playbook

ansible-galaxy collection install community.aws amazon.aws community.crypto

pip install boto3 botocore cryptography

AWS

1. Account with admin privileges

2. configure aws cli profile using script -

aws configure --profile default

User Inputs & Placeholders

Before deployment, set the cli environment using below script

```
export AWS_PROFILE=“default”

export AWS_PRIMARY_REGION="us-east-1"

export AWS_DR_REGION="us-west-2"

export AWS_ACCOUNT_ID="247590354562"

export TENANT_ID=“tenant001"
```

Primary Deployment

git clone https://github.com/inaw-tihor/iot-anomaly-deploy.git

cd iot-anomaly-deploy/infra/ansible

ansible-playbook -i inventories/hosts.ini playbooks/01-create-prim-resources.yml

ansible-playbook -i inventories/hosts.ini playbooks/02-create-s3-replication-role.yml

ansible-playbook -i inventories/hosts.ini playbooks/03-enable-s3-replication.yml

ansible-playbook -i inventories/hosts.ini playbooks/04-create-dynamodb-global.yml

Disaster Recovery Deployment

ansible-playbook -i inventories/hosts.ini playbooks/05-sync-models-to-dr.yml

ansible-playbook -i inventories/hosts.ini playbooks/06-create-sagemaker-dr.yml

ansible-playbook -i inventories/hosts.ini playbooks/07-create-kinesis-dr.yml

ansible-playbook -i inventories/hosts.ini playbooks/08-create-iot-dr.yml

Certificate Rollout & Greengrass Dual Publish

ansible-playbook -i inventories/hosts.ini playbooks/cert_rollout.yml

ansible-playbook -i inventories/hosts.ini playbooks/greengrass_dual_publish.yml

Post-Deployment Validation

Validate the replication using

cd validate

./validate_replication.sh

Checks

- S3 replication status
- DynamoDB in both regions
- SageMaker models in DR
- Kinesis streams in DR
- IoT resources in DR

Maintainers

Primary: DevOps Team Lead (devops@example.com)

Secondary: SRE Team (sre@example.com)

Business Impact

This system supports real-time monitoring of industrial assets. Downtime could delay anomaly detection and increase operational risk. Target RTO: 15 minutes. Target RPO: < 5 minutes.

Stakeholders

- Operations team

- Data Science team

- IT Security

- Management

Monitoring & Observability

AWS CloudWatch: IoT, Kinesis, DynamoDB, SageMaker metrics.

Prometheus/Grafana: Local container monitoring.

QuickSight: Dashboards for historical analysis.

Security, Patching, Upgrades & Notifications

- **Edge Devices**: Monthly `apt-get upgrade` via Ansible playbook
- **Containers**: Pull latest images monthly, rebuild configs
- **Alerts**: SNS topics integrated with Slack/PagerDuty
- **Policy**: IoT policy JSON stored in repo (infra/ansible/files/iot_policy.json)

Procedures

Primary Deployment

1. Clone Git repository and navigate to Ansible folder.

2. Run ansible-playbook 01-create-prim-resources.yml to create IoT, S3, DynamoDB, SageMaker, Kinesis.

3. Enable S3 replication using 02-create-s3-replication-role.yml and 03-enable-s3-replication.yml.

4. Create DynamoDB global tables with 04-create-dynamodb-global.yml.

Disaster Recovery Deployment

1. Sync SageMaker models (05-sync-models-to-dr.yml).

2. Create DR SageMaker endpoint (06-create-sagemaker-dr.yml).

3. Create DR Kinesis streams (07-create-kinesis-dr.yml).

4. Create DR IoT resources (08-create-iot-dr.yml).

Backup & Restore

Backup:

aws s3 sync for S3 buckets, dynamodb export-table-to-point-in-time for DynamoDB.

aws s3 sync s3://$S3_DATA_LAKE_BUCKET s3://$S3_BACKUP_BUCKET/data-lake

aws dynamodb export-table-to-point-in-time --table-arn arn:aws:dynamodb:${AWS_PRIMARY_REGION}:${AWS_ACCOUNT_ID}:table/${DYNAMODB_TABLE} --s3-bucket $S3_BACKUP_BUCKET

aws s3 sync s3://$S3_MODEL_BUCKET s3://$S3_BACKUP_BUCKET/models

Restore:

aws s3 sync from backup bucket to source bucket.

aws s3 sync s3://$S3_BACKUP_BUCKET/data-lake s3://$S3_DATA_LAKE_BUCKET

Environment Cleanup

aws s3 rb s3://$S3_DATA_LAKE_BUCKET --force

aws dynamodb delete-table --table-name $DYNAMODB_TABLE

aws kinesis delete-stream --stream-name $KINESIS_STREAM

DR Test & Failover Checklist

1. Stop IoT rule in primary.

2. Confirm DR stream & processing receive data.

3. Switch Greengrass to DR endpoint:

ansible-playbook -i inventories/hosts.ini playbooks/greengrass_dual_publish.yml --extra-vars "primary_endpoint=dr-endpoint"

4. Validate data in DR DynamoDB table.

5. Failback to Primary and validate.

Tenant Onboarding

Tenant onboarding with simple IoT Things (no Greengrass)

Create IoT Thing, keys & certs

```
aws iot create-thing \
 --thing-name "$THING_NAME" \
 --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"

aws iot create-keys-and-certificate \
 --set-as-active \
 --certificate-pem-outfile cert-${THING_NAME}.pem \
 --public-key-outfile public-${THING_NAME}.key \
 --private-key-outfile private-${THING_NAME}.key \
 --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE" \
 | tee cert-output.json

# Save the cert ARN for later:
export CERT_ARN=$(jq -r '.certificateArn' cert-output.json)
```

Download Amazon Root CA on the device (or here and SCP later):

```
curl -o AmazonRootCA1.pem https://www.amazontrust.com/repository/
AmazonRootCA1.pem
```

Attach a least-privilege IoT Thing policy

1. Create a minimal policy file (restrict to this tenant's topics):

```
cat > iot_policy_${TENANT_ID}.json <<'JSON'
{
  "Version": "2012-10-17",
```

```
  "Statement": [{
    "Effect": "Allow",
    "Action": ["iot:Connect", "iot:Publish", "iot:Subscribe", "iot:Receive"],
    "Resource": [
      "arn:aws:iot:${AWS_REGION}:${AWS_ACCOUNT_ID}:client/$
{THING_NAME}",
      "arn:aws:iot:${AWS_REGION}:${AWS_ACCOUNT_ID}:topic/tenants/$
{TENANT_ID}/*",
      "arn:aws:iot:${AWS_REGION}:${AWS_ACCOUNT_ID}:topicfilter/tenants/$
{TENANT_ID}/*"
    ]
  }]
}
JSON
```

Replace `${AWS_REGION}`, `${AWS_ACCOUNT_ID}`, `${THING_NAME}`, `${TENANT_ID}` in that file **or** use your repo template with envsubst.

2. Attach:

```
aws iot create-policy \
  --policy-name "$IOT_POLICY_NAME" \
  --policy-document file://iot_policy_${TENANT_ID}.json \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE" || true

aws iot attach-policy \
  --policy-name "$IOT_POLICY_NAME" \
  --target "$CERT_ARN" \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"

aws iot attach-thing-principal \
  --thing-name "$THING_NAME" \
  --principal "$CERT_ARN" \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"
```

Create tenant data plane (Kinesis, DynamoDB, S3 paths, SNS)

```
aws kinesis create-stream \

  --stream-name "$KINESIS_STREAM" \

  --shard-count 1 \

  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE" || true
```

```
# DDB table (per-tenant)

aws dynamodb create-table \

  --table-name "$DDB_TABLE" \

  --attribute-definitions AttributeName=pk,AttributeType=S
AttributeName=sk,AttributeType=S \

  --key-schema AttributeName=pk,KeyType=HASH AttributeName=sk,KeyType=RANGE \

  --billing-mode PAY_PER_REQUEST \

  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE" || true


# Ensure S3 bucket exists (or use shared bucket with per-tenant prefix)

aws s3api create-bucket \

  --bucket "$S3_DATALAKE_BUCKET" \

  --create-bucket-configuration LocationConstraint="$AWS_REGION_PRIMARY" \

  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE" || true


# SNS topic:

aws sns create-topic \

  --name "$SNS_TOPIC_NAME" \

  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"
```

Create an IoT Rule to route telemetry

This example fans out to **Kinesis** and **(optional) Lambda** for scoring.

```
export ROLE_ARN_FOR_RULE="<arn:aws:iam::${AWS_ACCOUNT_ID}:role/iot-
rule-actions-role>"

export LAMBDA_ARN="<arn:aws:lambda:${AWS_REGION_PRIMARY}:$
{AWS_ACCOUNT_ID}:function:score-telemetry-${TENANT_ID}>"


aws iot create-topic-rule \

  --rule-name "telemetry_to_kinesis_${TENANT_ID}" \
```

```
  --topic-rule-payload "{
    \"sql\": \"SELECT topic() as topic, timestamp() as ts, * FROM 'tenants/${TENANT_ID}/devices/+/telemetry'\",
    \"actions\": [
      {\"kinesis\": {\"roleArn\": \"${ROLE_ARN_FOR_RULE}\", \"streamName\": \"${KINESIS_STREAM}\", \"partitionKey\":\"${TENANT_ID}\"}}
      ${LAMBDA_ARN:+, {\"lambda\": {\"functionArn\": \"${LAMBDA_ARN}\"}}}
    ],
    \"ruleDisabled\": false
  }" \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"
```

Ensure iot-rule-actions-role has permissions for kinesis:PutRecord(s) and lambda:InvokeFunction.

IoT SiteWise asset model + asset

```
aws iotsitewise create-asset-model \
  --asset-model-name "${TENANT_ID}-PumpModel" \
  --asset-model-properties '[
    {"name":"Temperature","dataType":"DOUBLE","type":{"measurement":{}}},
    {"name":"Vibration","dataType":"DOUBLE","type":{"measurement":{}}}
  ]' \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"
```

Device side publisher (simple test)

1. Get the Data-ATS endpoint:

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"
# => a1b2c3d4e5-ats.iot.us-east-1.amazonaws.com
```

```
export IOT_ENDPOINT="<above>"

2. Publish test telemetry (MQTT with mTLS):

# Using mosquitto_pub installed on the device:

mosquitto_pub -h "$IOT_ENDPOINT" -p 8883 \
  --cafile AmazonRootCA1.pem \
  --cert cert-${THING_NAME}.pem \
  --key private-${THING_NAME}.key \
  -t "$TELEMETRY_TOPIC" \
  -m '{"ts":'"$(date +%s)"',"temperature":83.2,"vibration":0.19}' \
  -q 1
```

Validation of an onboarded tenant

Run these checks **end-to-end**. Do them once and save outputs in your onboarding ticket.

Connectivity & identity

```
# Thing & cert presence

aws iot describe-thing --thing-name "$THING_NAME" \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"


aws iot list-thing-principals --thing-name "$THING_NAME" \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"

# Expect: at least 1 certificate ARN listed
```

Publish a test payload and observe

1. **From device** (simple Thing or Greengrass edge component):

```
mosquitto_pub -h "$IOT_ENDPOINT" -p 8883 \
```

```
  --cafile AmazonRootCA1.pem \
  --cert cert-${THING_NAME}.pem \
  --key private-${THING_NAME}.key \
  -t "$TELEMETRY_TOPIC" \
  -m '{"ts":'"$(date +%s)"',"temperature":91.7,"vibration":0.25,"tenant":"'"$TENANT_ID"'"}'
-q 1
```

2. **subscribe** in another shell to confirm:

```
mosquitto_sub -h "$IOT_ENDPOINT" -p 8883 \
  --cafile AmazonRootCA1.pem \
  --cert cert-${THING_NAME}.pem \
  --key private-${THING_NAME}.key \
  -t "$IOT_TOPIC_BASE/+/#" -q 1
```

Data plane assertions

1. **Kinesis** (message arrival):

```
aws kinesis describe-stream-summary \
  --stream-name "$KINESIS_STREAM" \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"

# Quick sample read (find a shardId first)
SHARD=$(aws kinesis list-shards --stream-name "$KINESIS_STREAM" --query
'Shards[0].ShardId' --output text --region "$AWS_REGION_PRIMARY" --profile
"$AWS_PROFILE")
ITER=$(aws kinesis get-shard-iterator --stream-name "$KINESIS_STREAM" --shard-id
"$SHARD" --shard-iterator-type LATEST --query 'ShardIterator' --output text --region
"$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE")
aws kinesis get-records --shard-iterator "$ITER" --region "$AWS_REGION_PRIMARY" --
profile "$AWS_PROFILE"
```

Expect: Records[] length >= 1 after publish

2. **DynamoDB** (if your pipeline writes anomalies there):

```
aws dynamodb scan --table-name "$DDB_TABLE" \
  --select COUNT \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"
# Expect: Count >= 1 over time (depending on processing)
```

3. **S3** (if your pipeline stores raw/processed):

```
aws s3 ls s3://$S3_DATALAKE_BUCKET/${TENANT_ID}/ --recursive \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE" | tail
# Expect: new objects after publish
```

4. **Lambda** (if used in the rule):

```
aws cloudwatch logs describe-log-streams \
  --log-group-name "/aws/lambda/score-telemetry-${TENANT_ID}" \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE" | head
# Expect: recent log streams after publish
```

5. **SNS** (if alerts configured):

```
aws sns list-subscriptions-by-topic \
  --topic-arn "arn:aws:sns:${AWS_REGION_PRIMARY}:${AWS_ACCOUNT_ID}:${SNS_TOPIC_NAME}" \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"
# Expect: subscriptions Confirmed
```

Security & guardrails

- Verify the **IoT policy** only allows topics under `tenants/${TENANT_ID}/...` and clientId = `${THING_NAME}`.

- If Greengrass: ensure the **role alias** maps to an IAM role limited to required services (S3/Kinesis only, least-priv).
- Tag resources: tenant=${TENANT_ID}, env=prod|stage.

DR readiness (optional in onboarding)

- If using **dual-publish** firmware: publish to **DR endpoint** too and confirm DR Kinesis receives records.
- If using **S3 CRR/DDB Global Tables**: verify object replication and table replication show the DR region.

What to capture in the tenant onboarding ticket

- Tenant ID, regions, Thing name(s), certificate ARN(s).
- IoT Rule name(s) and actions.
- Data plane resources (stream/table/bucket prefix).
- Validation screenshots/snippets: Kinesis `get-records` sample, S3 listing, DDB count, Lambda logs, SNS test.
- Security notes: policy JSON link in repo, role alias/role ARNs.
- DR decision: dual-publish vs bridge, and outcome of DR smoke test.

Troubleshooting

S3 replication issues: Check IAM role trust policy.

IoT device not connecting: Verify certificate and policy.

Kinesis lag: Increase shard count.

References

Git Repository:

All playbooks, templates, and policies:

https://github.com/org/iot-anomaly-deploy/

Playbooks: infra/ansible/

Semaphore pipeline: semaphore/semaphore_pipeline.yaml