Industrial IoT Anomaly Detection — Operations Cookbook

Front Matter

Here is the front matter of this cookbook in yaml format.

```
---
doc_type: "runbook"
title: "Industrial IoT Anomaly Detection — Operations Cookbook"
slug: "iot-anomaly-ops"
version: "1.0"
service: "iot-anomaly-platform"
owners:
  - team: "DevOps"
    contact: "devops@example.com"
  - team: "SRE"
    contact: "sre@example.com"
audience: ["DevOps","SRE","Cloud Operations"]
envs: ["dev","stg","prod"]
regions: ["us-east-1","us-west-2"]
rto: "15m"
rpo: "5m"
tags:
     ["aws","iot","kinesis","sagemaker","dynamodb","s3","greengrass","a
     rgocd","ansible","rancher","dr"]
last_reviewed: "2024-06-15"
links:
  repo: "https://github.com/inaw-tihor/iot-anomaly-deploy"
  dashboards: "<quicksight/grafana urls>"
  alerts: "<pager/slack policy>"
  pipeline: "<semaphore/argocd url>"
---
```

Index

1) Purpose, Scope, Non-Goals

Purpose: Operate the Industrial IoT Anomaly Detection platform: deploy, replicate to DR, onboard tenants, validate end-to-end, monitor, secure, back up, restore, and fail over—**CLI-first**.
Scope: AWS IoT Core / SiteWise → Kinesis → S3 Data Lake → DynamoDB → SageMaker → SNS → QuickSight; Primary↔DR via S3 CRR & DynamoDB Global Tables; optional Greengrass edge. Local CI/CD (Rancher, ArgoCD, Ansible Semaphore) run as **Docker containers on the control machine** for manual gates only.
Non-Goals: ML model training details, business analytics design.

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.

2) Reference Architecture & Data Flow

Core components

- **Ingestion**: IoT Core (Things or Greengrass devices); SiteWise (asset modeling).
- **Warm path**: Kinesis Data Streams/Analytics for real-time anomaly scoring.
- **Cold path**: S3 Data Lake (raw + processed + models), SageMaker batch/inference, Lambda post-processing.
- **Serving**: DynamoDB (state/anomaly results), SNS (alerts), QuickSight (dashboards).
- **Control plane (local)**: Rancher, ArgoCD, Semaphore (all Docker on operator machine).
- **DR**: S3 Cross-Region Replication, DynamoDB Global Tables, mirrored IoT/Kinesis/SageMaker.

Primary–DR

- S3: Cross Region Replication (**CRR)** enabled between primary and Disaster Recovery (DR) buckets.
- DynamoDB: **Global Tables** spanning both regions.
- Kinesis/SageMaker/IoT: **mirrored resources** in DR for rapid activation.
- Greengrass (optional): **dual-publish** to Primary and DR endpoints.

Data flow (high-level)
Device → IoT Core/SiteWise → Kinesis (warm) → SageMaker (inference) → DynamoDB/S3 → SNS alerts → QuickSight dashboards; continuous S3 + DDB replication to DR.

Description

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.

Components of Architecture

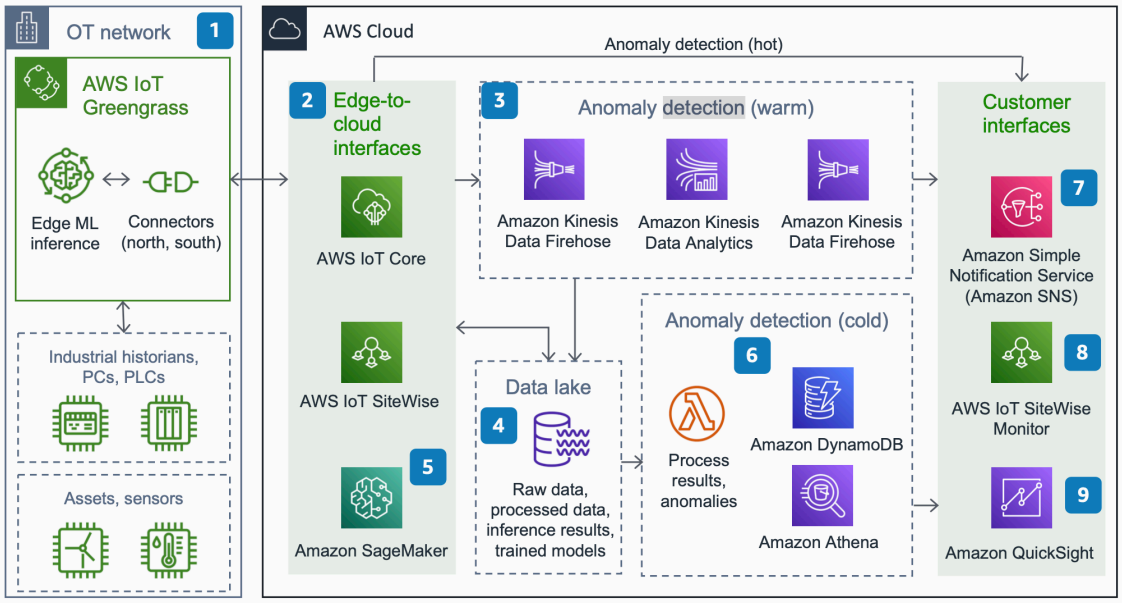
Layer	Service / Tool	Purpose
Smart Edge	IoT Greengrass	Runs local inference and forwards telemetry to AWS IoT Core.
Edge	IoT Core Thing Device	Sends raw telemetry data to AWS IoT Core
Ingestion	AWS IoT Core	Secure MQTT broker, receives device telemetry.
Data Routing	IoT Rules	Routes messages to Kinesis, 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

Anomaly detection for industrial workloads

Edge-to-cloud detection paths for hot, warm, and cold analysis

Use IoT, analytics, and machine learning services to inform operational technology teams of performance anomalies.



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.

Reviewed for technical accuracy October 12, 2021

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Reference Architecture

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.

3) Preconditions & Inputs (parameterized)

3.1 Control Machine Setup

Below setups are needed on the control machine.

```
# Required CLIs
brew install awscli ansible jq
brew install --cask docker && open -a Docker
docker --version

# Rancher (local, optional UI)
docker run -d --name rancher --restart=unless-stopped \
  -p 80:80 -p 443:443 --privileged \
  -v /opt/rancher:/var/lib/rancher rancher/rancher:latest

# Argo CD (requires Kubernetes enabled in Docker Desktop)
kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/
    argo-cd/stable/manifests/install.yaml
kubectl -n argocd get pods
# (optional) UI: kubectl port-forward svc/argocd-server -n argocd
    8081:443
# (optional) initial admin password:
#   argocd admin initial-password -n argocd

# Semaphore (Ansible UI)
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 \
  -v semaphore-data:/var/lib/semaphore -v semaphore-config:/etc/
      semaphore \
  semaphoreui/semaphore:latest

# Ansible AWS modules
ansible-galaxy collection install community.aws amazon.aws
    community.crypto
pip install boto3 botocore cryptography
```

3.2 AWS account & profile

```
aws configure ––profile default
```

3.3 Inputs (set once per session)

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="<12-digit>"
export TENANT_ID="tenant001"

# Common resource names (per-tenant patterns encouraged)
export KINESIS_STREAM="anomaly-stream-${TENANT_ID}"
export DDB_TABLE="anomaly-results-${TENANT_ID}"
export S3_DATALAKE_BUCKET="iot-data-lake-${TENANT_ID}-${AWS_PRIMARY_REGION}"
export SNS_TOPIC_NAME="alerts-${TENANT_ID}"
```

EXPECTED: aws --version, ansible --version, docker --version, jq --version all succeed.

4) Golden Path (Day-0/1/2)

1. **Day-0 (Provision)**: bootstrap Primary infra; enable S3 CRR & DDB Global Tables; baseline monitoring.
2. **Day-1 (Ship)**: mirror critical resources in DR; validate; enable alerts.
3. **Day-2 (Operate)**: tenant onboarding, cert rollouts, patching, cost/scale changes, backups & DR drills.

5) Primary Deployment (CLI-first)

IaC lives in Git. Clone onto control machine and run Ansible.

```
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
```

EXPECTED:

- S3 data lake bucket exists in **Primary**; CRR role created.
- DDB table exists in **both** regions (Global Table).
- Kinesis stream created; SiteWise model initialized; IoT rule scaffolding present.

Rollback:

re-apply last good tag (git checkout <tag> → re-run the same playbooks).

6) DR / Replica Deployment (CLI-first)

```
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
```

EXPECTED:

- SageMaker model artifacts available in DR; DR endpoint deployed.
- DR Kinesis stream present; mirrored IoT resources ready.

7) Tenant Onboarding

7A) Track A — IoT Things (no Greengrass)

```
# Thing + certs
THING_NAME="${TENANT_ID}-device1"
aws iot create-thing --thing-name "$THING_NAME" --region "$AWS_PRIMARY_REGION" --profile "$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_PRIMARY_REGION" --profile "$AWS_PROFILE" | tee cert-output.json
export CERT_ARN=$(jq -r '.certificateArn' cert-output.json)

# Amazon Root CA
curl -o AmazonRootCA1.pem https://www.amazontrust.com/repository/AmazonRootCA1.pem

# Least-privilege IoT policy (scoped to this tenant/topic space)
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

IOT_POLICY_NAME="tenant_policy_${TENANT_ID}"
aws iot create-policy --policy-name "$IOT_POLICY_NAME" --policy-document file://iot_policy_${TENANT_ID}.json \
  --region "$AWS_PRIMARY_REGION" --profile "$AWS_PROFILE" || true
aws iot attach-policy --policy-name "$IOT_POLICY_NAME" --target "$CERT_ARN" \
  --region "$AWS_PRIMARY_REGION" --profile "$AWS_PROFILE"
aws iot attach-thing-principal --thing-name "$THING_NAME" --principal "$CERT_ARN" \
  --region "$AWS_PRIMARY_REGION" --profile "$AWS_PROFILE"

# Tenant data plane
aws kinesis create-stream --stream-name "$KINESIS_STREAM" --shard-count 1 \
  --region "$AWS_PRIMARY_REGION" --profile "$AWS_PROFILE" || true

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_PRIMARY_REGION" --profile "$AWS_PROFILE" || true

aws s3api create-bucket --bucket "$S3_DATALAKE_BUCKET" \
  --create-bucket-configuration LocationConstraint="$AWS_PRIMARY_REGION" \
  --region "$AWS_PRIMARY_REGION" --profile "$AWS_PROFILE" || true

aws sns create-topic --name "$SNS_TOPIC_NAME" --region "$AWS_PRIMARY_REGION" --profile "$AWS_PROFILE"

# IoT Rule → Kinesis (+optional Lambda for scoring)
ROLE_ARN_FOR_RULE="<arn:aws:iam::${AWS_ACCOUNT_ID}:role/iot-rule-actions-role>"
LAMBDA_ARN="<arn:aws:lambda:${AWS_PRIMARY_REGION}:${AWS_ACCOUNT_ID}:function:score-telemetry-${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 "$AWS_PRIMARY_REGION" --profile "$AWS_PROFILE"
```

EXPECTED: Thing + cert active; policy attached; Kinesis/DDB/S3/SNS exist; IoT Rule enabled.

7B) Track B — Greengrass (edge, dual-publish optional)

```
# Certificate rollout to devices
ansible-playbook -i inventories/hosts.ini playbooks/cert_rollout.yml

# Dual-publish config (Primary + DR)
ansible-playbook -i inventories/hosts.ini playbooks/
     greengrass_dual_publish.yml \
  --extra-vars "primary_endpoint=<primary-ats-endpoint>
       dr_endpoint=<dr-ats-endpoint>"
```

EXPECTED: New certs installed; device publishes to Primary and DR (or switchable).

8) Post-Deployment Validation

```
# Endpoint + test telemetry
IOT_ENDPOINT=$(aws iot describe-endpoint --endpoint-type iot:Data-ATS \
  --region "$AWS_PRIMARY_REGION" --profile "$AWS_PROFILE" --query endpointAddress -o
        text)
TELEMETRY_TOPIC="tenants/${TENANT_ID}/devices/${THING_NAME}/telemetry"

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

# Kinesis sample read
SHARD=$(aws kinesis list-shards --stream-name "$KINESIS_STREAM" --query
      'Shards[0].ShardId' --output text --region "$AWS_PRIMARY_REGION" --profile
      "$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_PRIMARY_REGION" --profile "$AWS_PROFILE")
aws kinesis get-records --shard-iterator "$ITER" --region "$AWS_PRIMARY_REGION" --profile
      "$AWS_PROFILE"

# DynamoDB count
aws dynamodb scan --table-name "$DDB_TABLE" --select COUNT \
  --region "$AWS_PRIMARY_REGION" --profile "$AWS_PROFILE"

# S3 object listing
aws s3 ls s3://$S3_DATALAKE_BUCKET/${TENANT_ID}/ --recursive \
  --region "$AWS_PRIMARY_REGION" --profile "$AWS_PROFILE" | tail
```

EXPECTED:

- get-records returns ≥1 record shortly after publish.
- DDB COUNT grows as pipeline processes.
- S3 shows new objects.
- (If Lambda used) CloudWatch logs show recent invocations.
- (If SNS used) Topic has Confirmed subscriptions.

9) Monitoring & Observability

CloudWatch: IoT auth failures / Rule errors, Kinesis IteratorAge, DynamoDB Throttles & ConsumedCapacity, SageMaker Invocations & Latency.

Prometheus/Grafana: local container health (Rancher/ArgoCD/Semaphore).

QuickSight: anomaly trends, fleet performance.

Alert catalog (examples)

Alert	Signal	Threshold	Action
Kinesis lag	IteratorAge	> 10 min	Add shards / scale consumers
IoT Rule errors	Failed actions	> 10 in 5 min	Inspect rule role/payload
SageMaker latency	p95	> 1s 10 min	Scale endpoint / check model
DDB throttles	ThrottledRequests	> 0 sustained	Raise WCU/RCU or backoff

10) Security, Patching, Upgrades & Notifications

Edge devices: monthly OS updates via Ansible (apt/yum).

Containers: pull latest base images monthly; rebuild.

IoT policies: least privilege—restrict to `tenants/${TENANT_ID}/**` and `client/${THING_NAME}`.

Notifications: SNS topics to Slack/PagerDuty; acknowledgement required.

Secrets: store in SSM/Secrets Manager; rotate on schedule.

11) Backup, Restore, DR & Failover

11.1 Backups

```
# S3 data and models
aws s3 sync s3://$S3_DATALAKE_BUCKET s3://$S3_BACKUP_BUCKET/data-lake
aws s3 sync s3://$S3_MODEL_BUCKET s3://$S3_BACKUP_BUCKET/models

# DynamoDB PITR export
aws dynamodb export-table-to-point-in-time \
  --table-arn arn:aws:dynamodb:${AWS_PRIMARY_REGION}:${AWS_ACCOUNT_ID}:table/${DDB_TABLE}
        \
  --s3-bucket $S3_BACKUP_BUCKET
```

11.2 Restore

```
aws s3 sync s3://$S3_BACKUP_BUCKET/data-lake s3://$S3_DATALAKE_BUCKET
# (Use DDB import-from-S3 or point stack to restored table if needed.)
```

11.3 DR Failover (play-by-play)

Quiesce Primary: disable IoT Rules in Primary.

Flip publishing: switch device endpoint to DR or enable Greengrass dual-publish.

Validate DR path: Kinesis records flowing, DDB writes, SageMaker healthy.

Comms: incident updates every 15m.

Failback: re-enable Primary, verify parity, switch endpoints back.

Greengrass helper (dual-publish):

```
ansible-playbook -i inventories/hosts.ini playbooks/greengrass_dual_publish.yml \
  --extra-vars "primary_endpoint=<primary> dr_endpoint=<dr>"
```

12) Environment Cleanup

```
aws s3 rb s3://$S3_DATALAKE_BUCKET --force
aws dynamodb delete-table --table-name $DDB_TABLE
aws kinesis delete-stream --stream-name $KINESIS_STREAM
# (Mind replication & shared resources before delete.)
```

13) Troubleshooting (quick table)

Symptom	Likely cause	Fast fix
Device can't connect (mTLS)	Wrong endpoint/cert/ policy	Verify ATS endpoint; re-issue cert; attach policy
No S3 CRR objects in DR	CRR role or rule misconfigured	Fix bucket replication config & IAM trust
Kinesis lag grows	Under-sharded / slow consumer	Add shards; scale consumers; optimize batch size
DDB throttles	Insufficient capacity	Increase RCU/WCU; enable adaptive capacity
SageMaker slow	Model/instance underprovisioned	Scale up/out; check model container logs

14) Business Impact & Stakeholders

Impact: delays in anomaly detection raise operational risk across industrial assets.

Targets: RTO 15m, RPO < 5m.

Stakeholders: Operations, Data Science, IT Security, Management.

15) Change History & Review Cadence

v1.0 — Initial runbook consolidation; DR & onboarding finalized.

Review quarterly or after major architecture change.

16) References

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

- Playbooks: infra/ansible/
- Semaphore pipeline: semaphore/semaphore_pipeline.yaml
- IoT policy template: infra/ansible/files/iot_policy.json

==========

17) Miscellaneous Procedures

17.1) Tenant onboarding with simple IoT Things (no Greengrass)

17.1.1) 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)
```

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

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

17.1.3) Attach a least-privilege IoT Thing policy

17.1.3.1) 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.

17.1.3.2) 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"
```

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

17.1.4.1) Kinesis

```
aws kinesis create-stream \
  --stream-name "$KINESIS_STREAM" \
  --shard-count 1 \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE" || true
```

17.1.4.2) 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
```

17.1.4.3) 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
```

17.1.4.4) SNS topic:

```
aws sns create-topic \
  --name "$SNS_TOPIC_NAME" \
  --region "$AWS_REGION_PRIMARY" --profile "$AWS_PROFILE"
```

17.1.5) 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 (Optional)

```
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"
```

17.1.6) 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
```

17.2) Validation of an onboarded tenant

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

17.2.1) 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"
```

Expected: at least 1 certificate ARN listed

17.2.2) Publish a test payload and observe

17.2.2.1) **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
```

17.2.2.2) **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
```

17.2.3) Data plane assertions

17.2.3.1) **Kinesis** (message arrival):

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

17.2.3.2) 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"
```

Expected: Records[] length >= 1 after publish

17.2.3.3) **DynamoDB** (if your pipeline writes anomalies there):

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

17.2.3.4) **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
```
Expected: new objects after publish

17.2.3.5) **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
```
Expected: recent log streams after publish

17.2.3.6) **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"
```
Expected: 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.

17.3) 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
```

17.4) 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.