



Water Sensor Tracker

A Cloud-Native IoT Solution

Project Overview

The Water Sensor Tracker is a cloud-native IoT solution built with:

-  AWS AWS CDK (Infrastructure as Code)
-  Python Lambda Functions
- API Gateway REST API
- DynamoDB for real-time data
- S3 for data archival

Architecture

```
graph LR
    Sensor[IoT Sensor] --> API[API Gateway]
    API --> Lambda[Lambda Function]
    Lambda --> DynamoDB[DynamoDB Table]
    Lambda --> S3[S3 Bucket]

    style Sensor fill:#81ecec,stroke:#00cec9
    style API fill:#74b9ff,stroke:#0984e3
    style Lambda fill:#a8e6cf,stroke:#3b7d4f
    style DynamoDB fill:#ffeaa7,stroke:#fdcb6e
    style S3 fill:#fab1a0,stroke:#e17055
```

Infrastructure (CDK)

Stack Components

DynamoDB Table

- Partition Key: `sensorId`
- Sort Key: `timestamp`
- TTL: 90 days
- Environment-specific settings:
 - Dev: Delete on destroy
 - Prod: Retain on destroy

S3 Bucket

- Versioning enabled
- Lifecycle rules:
 - Dev: Move to IA after 30 days

Lambda Function Implementation

Data Processing (POST /sensors)

```
def process_sensor_data(event):
    # Validate required fields
    required_fields = ['sensorId', 'temperature',
                       'humidity', 'waterLevel']

    # Store in DynamoDB
    item = {
        'sensorId': body['sensorId'],
        'timestamp': timestamp,
        'temperature': body['temperature'],
        'humidity': body['humidity'],
        'waterLevel': body['waterLevel'],
        'ttl': int(time.time()) + (90 * 24 * 60 * 60)
    }

    # Archive to S3
    s3.put_object(
        Bucket=bucket_name,
        Key=f"sensors/{body['sensorId']}/{timestamp}.json",
        Body=json.dumps(item)
```

Testing Strategy

Infrastructure Tests (Jest)

```
it('creates DynamoDB table with dev settings', () => {
  template.hasResourceProperties('AWS::DynamoDB::Table', {
    BillingMode: 'PAY_PER_REQUEST',
    TimeToLiveSpecification: {
      AttributeName: 'ttl',
      Enabled: true
    }
  });

  template.hasResource('AWS::DynamoDB::Table', {
    DeletionPolicy: 'Delete',
    UpdateReplacePolicy: 'Delete'
  });
});
```

Lambda Function Tests (pytest)

```
def test_post_sensor_data(dynamodb_table, s3_bucket):
```

Development Workflow

Local Development

```
# Install dependencies
npm install
pip install -r requirements-dev.txt

# Run tests
npm test
```

Deployment

```
# Deploy to development
npm run deploy:dev

# Deploy to production
npm run deploy:prod
```

Best Practices Implemented

Infrastructure as Code

- Versioned infrastructure
- Environment-specific configurations
- Consistent deployments

Security

- Least privilege IAM roles
- Environment separation
- Resource policies

Testing

- Comprehensive test coverage
- Mock AWS services

Future Enhancements

Monitoring & Alerting

- CloudWatch alarms
- SNS notifications
- Anomaly detection

Analytics

- Data visualization
- Trend analysis
- Reporting system

Security

- API authentication
- Data encryption

Thank You!

For more information:

- Source code: [GitHub Repository]
- Documentation: [Project Wiki]
- Contact: [Team Contact]