

# Edge Deep LearningにおけるMLOps

# Edge Deep Learningを商用導入する際の課題

## 機械学習モデルの商用導入における課題

- データ管理
- 実験管理
- モデル管理
- アプリケーション管理

## Edge環境特有の課題

- デバイス管理
- デバイスへのデプロイの管理
- デバイスからのデータ収集



# Edge Deep Learningを商用導入する際の課題を AWSのコンポーネントを使って解決する

## 機械学習モデルの商用導入における課題

- データ管理 → Amazon SageMaker Ground Truth / Amazon S3
- 実験管理 → Amazon SageMaker
- モデル管理 → Amazon SageMaker / AWS IoT Greengrass
- アプリケーション管理 → AWS Lambda / AWS IoT Greengrass

## Edge環境特有の課題

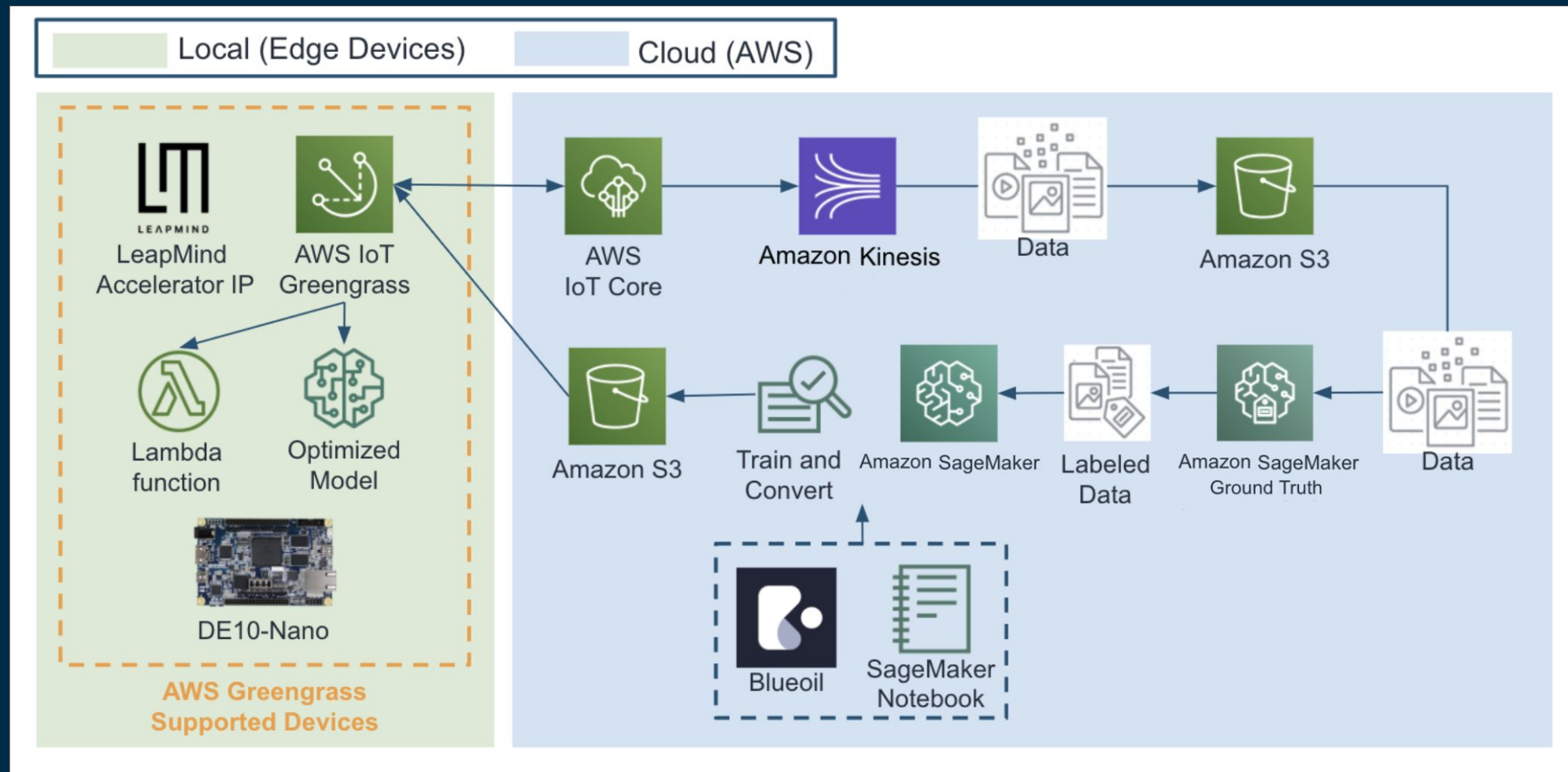
- デバイス管理 → AWS IoT Things / AWS IoT Greengrass
- デバイスへのデプロイの管理 → AWS IoT Greengrass
- デバイスからのデータ収集 → AWS IoT Greengrass / Amazon Kinesis



# Bluegrass

A sample MLOps system of LeapMind Blueoil x AWS Components (AWS IoT Greengrass, Amazon SageMaker)

Blueoil x AWS Componentsで構成されたシステムをGithubにて[OSSとして公開](#)しました。



# Bluegrass

## 技術ブログのご紹介

下記のように、AWSより詳細な技術ブログが公開されています。ぜひこちらもご確認ください。

### Training / Converting Models (Amazon SageMaker x Intel FPGA Edge Devices)

- [\[English\] Using Fewer Resources to Run Deep Learning Inference on Intel FPGA Edge Devices](#)
- [\[和訳\] 効率的にインテル® FPGA エッジデバイス上の深層学習推論を実行する](#)

追加で下記について、LeapMindより技術ブログが公開される予定となっています！

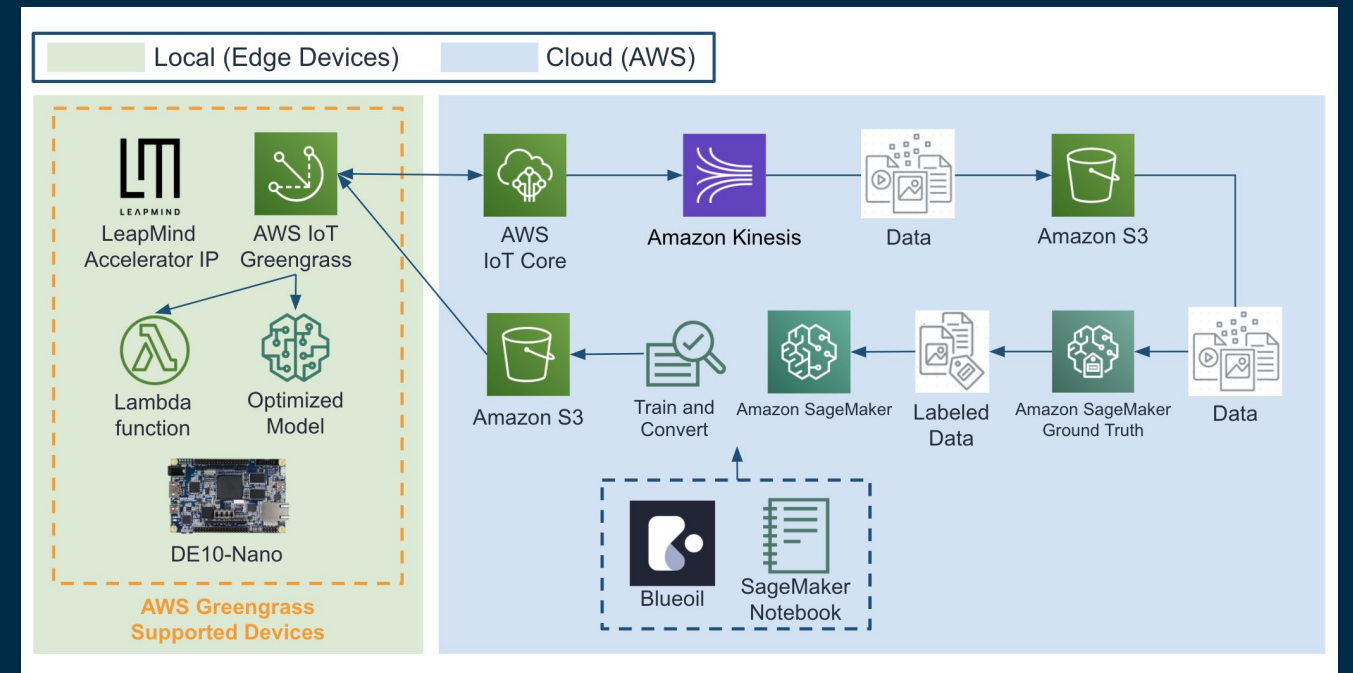
### Deployment / Device Management / Data Aggregation (AWS IoT Greengrass x Intel FPGA Edge Devices)



# Setup AWS component and devices by bluegrass

下記のStepで必要な全てのコンポーネントが準備され、Device上での推論デモを実行できます。

- Step 0 事前準備
- Step 1 Amazon S3 bucketとIAM Roleの作成
- Step 2 Amazon SageMaker上での学習の実行
- Step 3 AWS IoT Greengrass Groupの作成
- Step 4 Deviceのセットアップ
- Step 5 Application Deployと推論実施



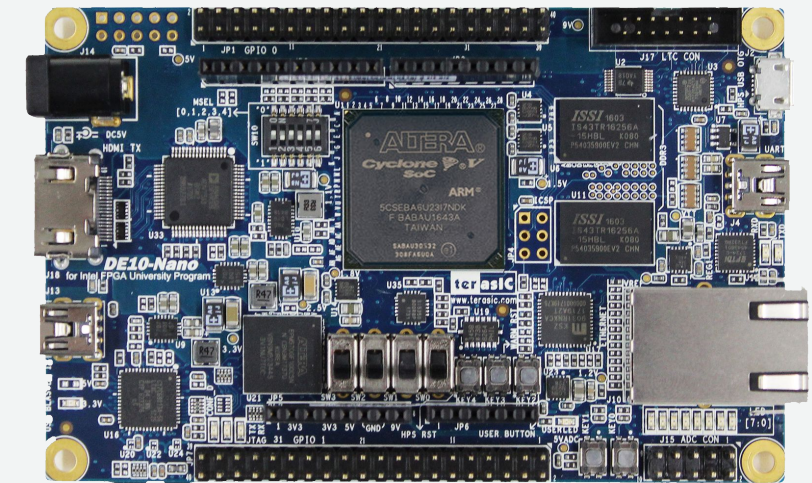
Step 0

# 事前準備

## AWS CLI及びAnsibleのセットアップとDeviceの準備

### 前提条件

- AWS CLIのセットアップ
- DeviceセットアップのためのAnsible
- Terasic社製DE10-Nanoボード及びUSBカメラ
- イーサネットケーブルとインターネット接続
- ノート PC またはローカル開発環境

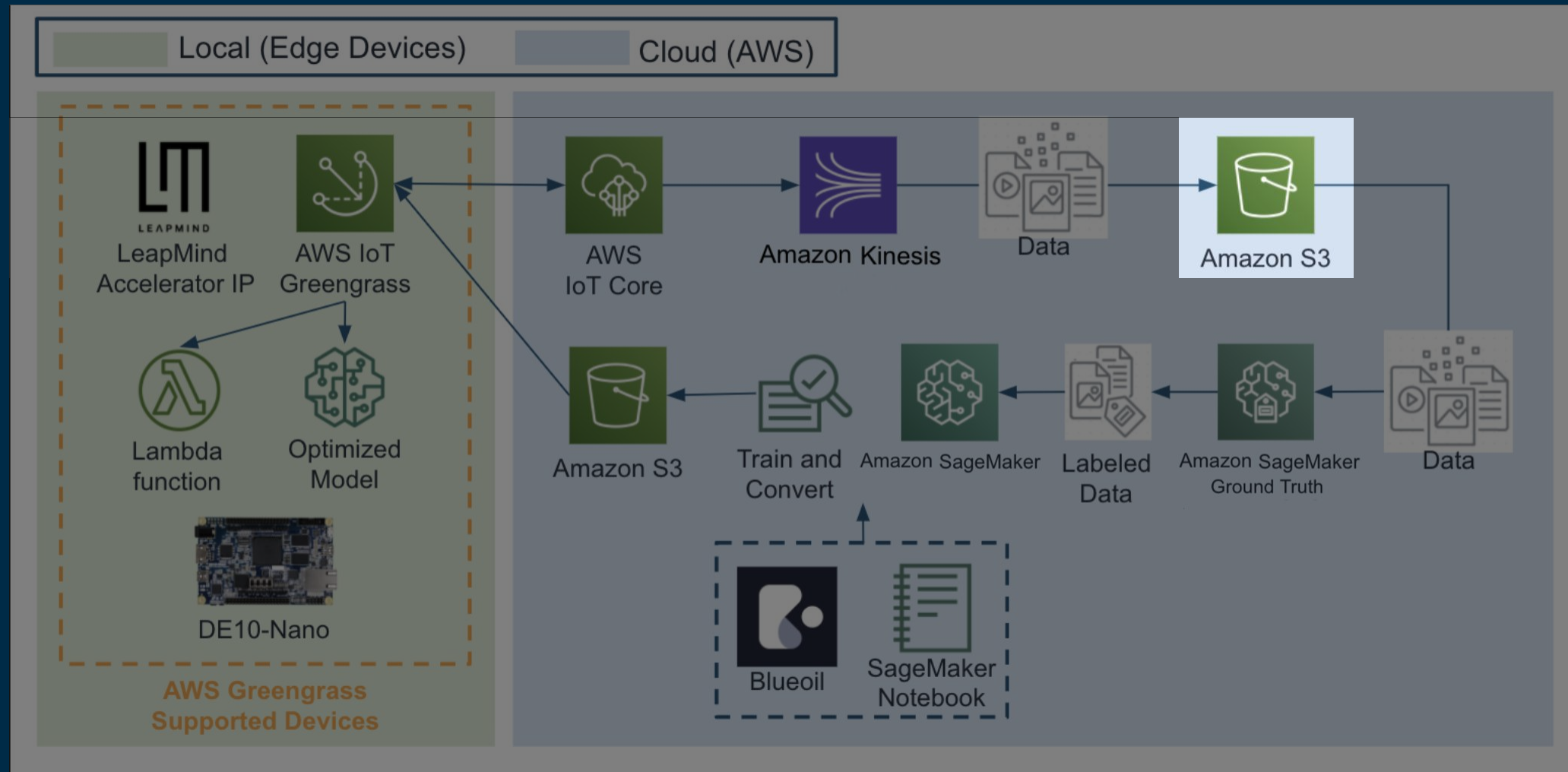


Cyclone® V SoC FPGAを搭載した  
Terasic社製DE10-Nanoボード

Step 1

# Amazon S3 bucketとIAM Roleの作成

下記のコンポーネントを作成します。





## Step 1

# Amazon S3 bucketとIAM Roleの作成

## Greengrass用S3 bucket及びRoleの作成

下記のコマンドでS3 bucket及びGreengrass用のIAM Roleを作成します。

```
$ export BLUEGRASS_S3_BUCKET_NAME=[your S3 bucket name]
$ aws cloudformation create-stack --stack-name BluegrassS3 --capabilities
CAPABILITY_NAMED_IAM --template-body file://$(pwd)/deploy/s3.yaml --parameters
ParameterKey="S3BucketName",ParameterValue="${BLUEGRASS_S3_BUCKET_NAME}"
```

- BluegrassRole
  - Greengrass用のRole
- BluegrassS3Bucket
  - Greengrassで利用するためのS3 bucket, 推論データなどを保存する
- BluegrassS3Policy
  - BluegrassS3BucketやSageMakerのbucketへの読み書きを可能にするPolicy
- BlueoilSageMakerRole
  - SageMaker用のRole

Step 1

# Amazon S3 bucketとIAM Roleの作成

## Greengrass用S3 bucket及びRoleの作成

下記のように作成されます。

Resources (4) <span>🔄</span>					
<input type="text" value="Search resources"/> <span>⚙️</span>					
Logical ID ▲	Physical ID ▼	Type ▼	Status ▼	Status reason ▼	
BluegrassRole	<a href="#">BluegrassRole</a> <span>🔗</span>	AWS::IAM::Role	✔️ CREATE_COMPLETE	-	
BluegrassS3Bucket	<a href="#">leapmind-bluegrass-demo</a> <span>🔗</span>	AWS::S3::Bucket	✔️ CREATE_COMPLETE	-	
BluegrassS3Policy	Blueg-Blue-	AWS::IAM::Policy	✔️ CREATE_COMPLETE	-	
BlueoilSagemakerRole	<a href="#">BlueoilSagemakerRole</a> <span>🔗</span>	AWS::IAM::Role	✔️ CREATE_COMPLETE	-	

## Step 1

# Amazon S3 bucketとIAM Roleの作成

## AWS CloudFormation Templateの抜粋

- BluegrassS3Policy

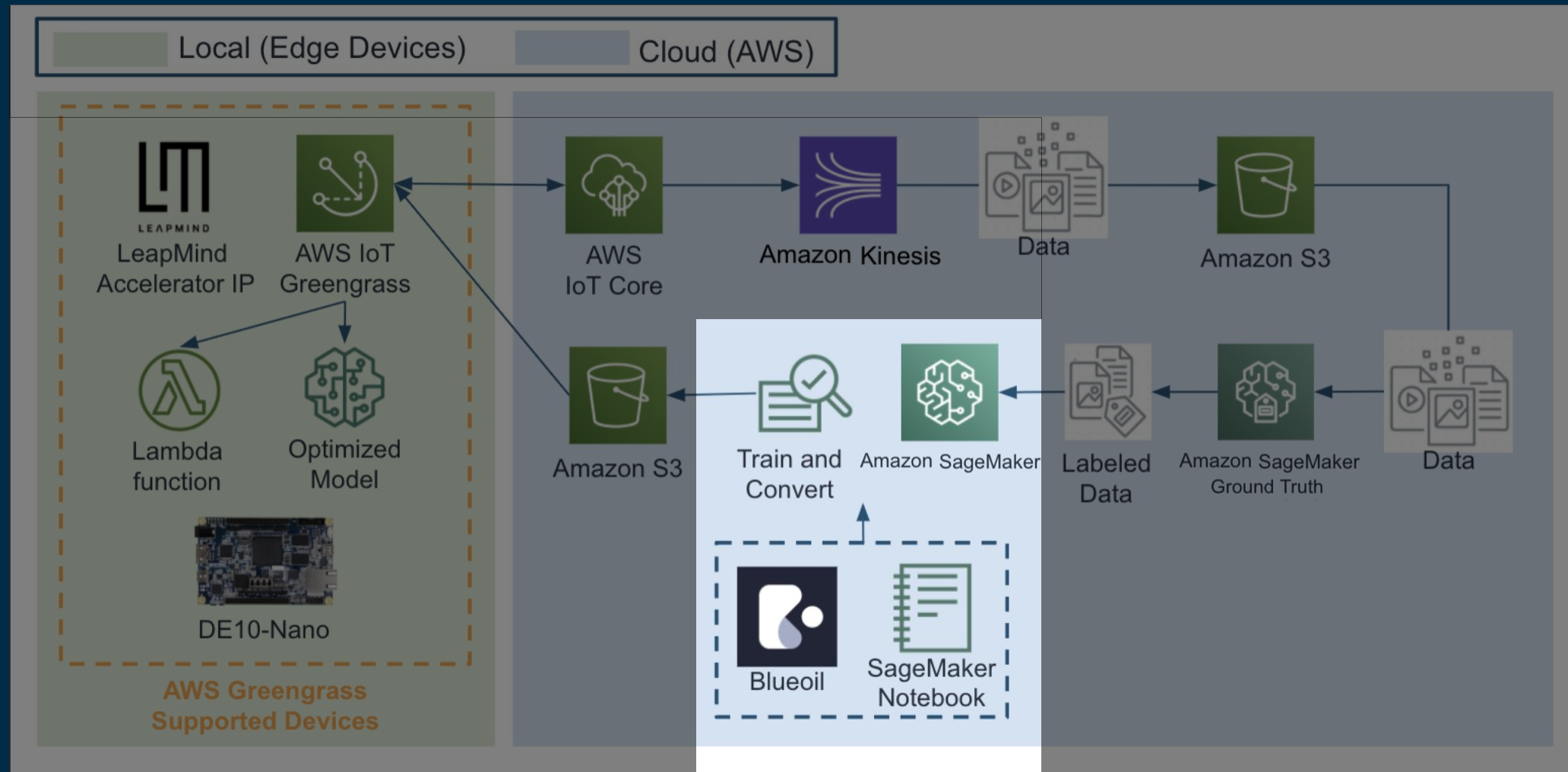
```
BluegrassS3Policy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyName: "BluegrassS3Policy"
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
        -
          Action:
            - "s3:ListBucket"
          Effect: "Allow"
          Resource:
            - "arn:aws:s3:::SageMaker"
            - Fn::GetAtt:
                - BluegrassS3Bucket
                - Arn
        -
```

```
          Action:
            - "s3:GetObject"
            - "s3:PutObject"
            - "s3:DeleteObject"
          Effect: "Allow"
          Resource:
            - "arn:aws:s3:::SageMaker/*"
            - Fn::Join:
                - "/"
                - - Fn::GetAtt:
                    - BluegrassS3Bucket
                    - Arn
                  - "*"
        -
      Roles:
        - Ref: "BluegrassRole"
        - Ref: "BlueoilSagemakerRole"
```

Step 2

# Amazon SageMaker上での学習の実行

下記のコンポーネントを作成します。



Step 2

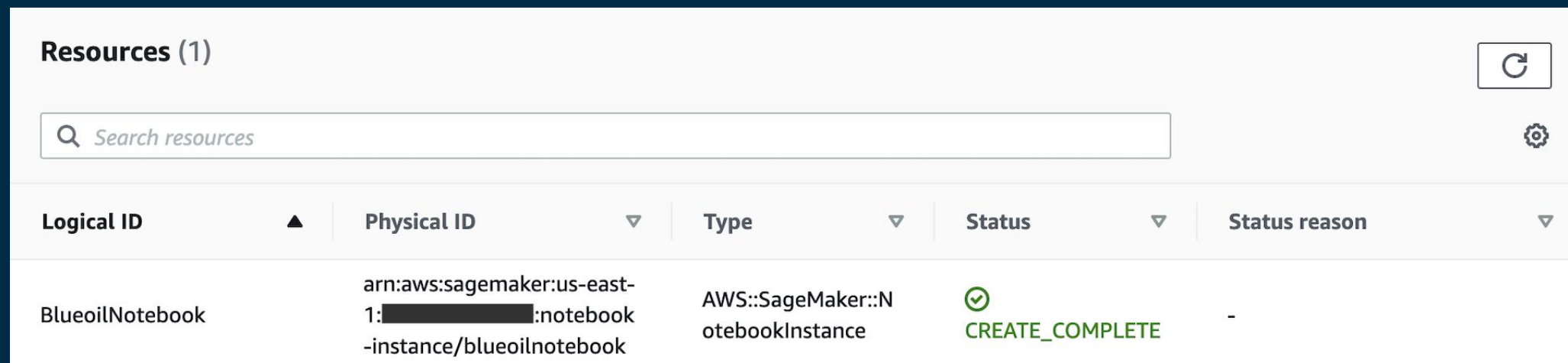
# Amazon SageMaker上での学習の実行

## Amazon SageMaker Notebookの作成

下記のコマンドでAmazon SageMaker Notebookを作成します。

```
$ aws cloudformation create-stack --stack-name BlueoilSagemaker --template-body file://$(pwd)/deploy/sagemaker.yaml
```

- BlueoilNotebook
  - BluegrassがDefaultで配置されたSageMaker Notebook



The screenshot shows the AWS CloudFormation console interface. At the top, it says "Resources (1)" with a refresh icon. Below that is a search bar labeled "Search resources" and a settings gear icon. The main content is a table with the following columns: Logical ID, Physical ID, Type, Status, and Status reason. The table contains one row for the resource "BlueoilNotebook".

Logical ID	Physical ID	Type	Status	Status reason
BlueoilNotebook	arn:aws:sagemaker:us-east-1:██████████:notebook-instance/blueoilnotebook	AWS::SageMaker::NotebookInstance	✔ CREATE_COMPLETE	-

## Step 2

# Amazon SageMaker上での学習の実行

## AWS CloudFormation Templateの抜粋

- BlueoilNotebook

```
BlueoilNotebook:
  Type: 'AWS::SageMaker::NotebookInstance'
  Properties:
    InstanceType: "m1.t3.medium"
    NotebookInstanceName: "BlueoilNotebook"
    RoleArn:
      Fn::Join:
        - ':'
        - - 'arn:aws:iam:'
          - Ref: 'AWS::AccountId'
          - role/BlueoilSagemakerRole
    DefaultCodeRepository: 'https://github.com/LeapMind/bluegrass'
```

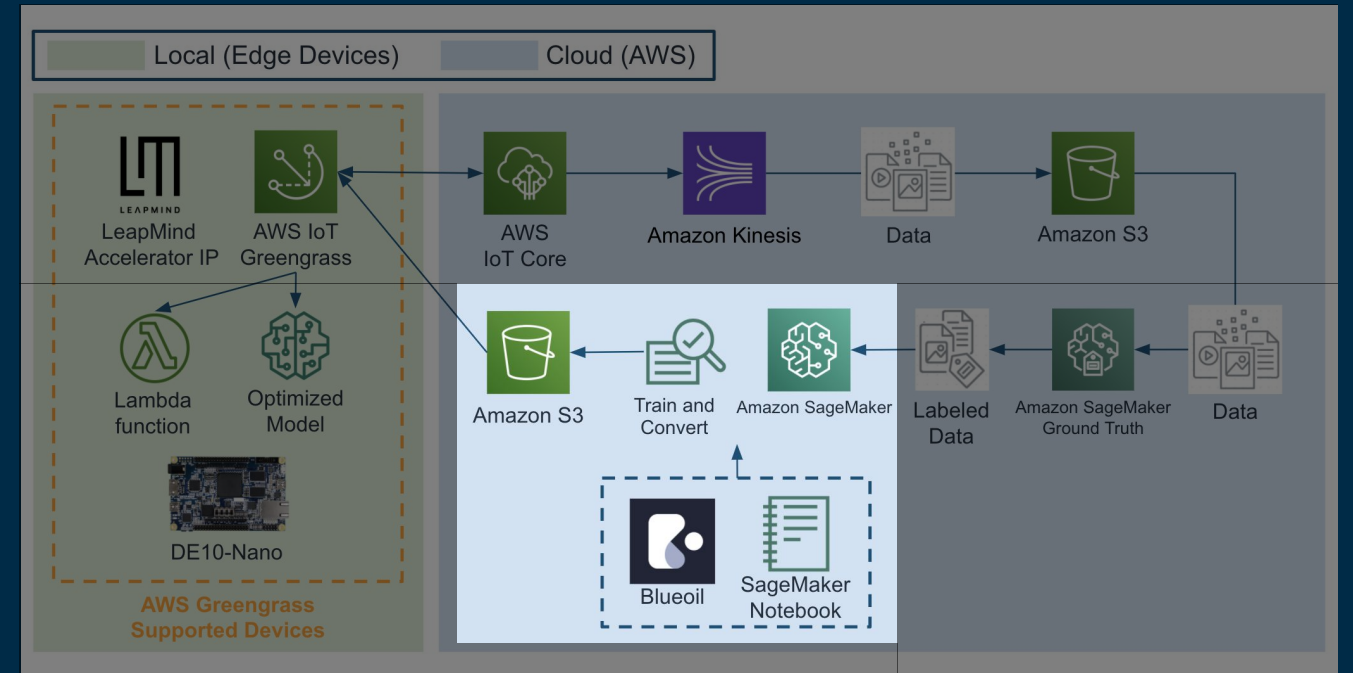
## Step 2

# Amazon SageMaker上での学習の実行

## Bluegrassにおける役割

SageMaker NotebookからオンデマンドでGPU Instanceを立ち上げて学習を実行することが可能となります。

- モデルの学習はSageMaker SDKを利用してTraining Jobとして実行
- Binary fileへの変換はSageMaker Processingを利用



## Step 2

# Amazon SageMaker上での学習の実行

## An example of training with Blueoil

学習を実行するためのexampleとしてnotebookを提供しているので、AWSコンソールからSageMaker Notebookを起動し、blueoil\_\*\_example.ipynbを開くことで学習を実行することができます。

```
!bash ./docker_push_ecr.sh blueoil-sagemaker blueoil/blueoil:v0.22.0

Preparing data (upload CIFAR-10 to Amazon S3)

Create sagemaker session

import sagemaker
sess = sagemaker.Session()

import os
import shutil

def upload_data(sess, path, key_prefix='data', compress=False):
    if compress:
        path = shutil.make_archive(path, 'gztar', '.', path)
    s3_data = sess.upload_data(path=path, key_prefix=key_prefix)
    return s3_data

Download CIFAR-10 dataset

!curl -O https://s3-ap-northeast-1.amazonaws.com/leapmind-public-storage/datasets/cifar.tgz
!tar xzf cifar.tgz

Upload dataset
```



## Step 2

# Amazon SageMaker上での学習の実行

## SageMaker Estimatorを利用したon-demand training

esrimator.fit()を実行することで、GPU instanceが立ち上がり、そのinstance上で指定したDocker imageでの学習が開始されます。

```
Train  
  
On Amazon Sagemaker on-demand instance  
  
Create session  
  
In []: import boto3  
  
algorithm_name = 'blueoil-sagemaker'  
  
client = boto3.client('sts')  
account = client.get_caller_identity()['Account']  
  
my_session = boto3.session.Session()  
region = my_session.region_name  
  
ecr_image = '{}.dkr.ecr.{}.amazonaws.com/{}:latest'.format(account, region, algorithm_name)  
  
Run training  
  
In []: import sagemaker  
from sagemaker.estimator import Estimator  
  
train_instance_type = 'ml.p2.xlarge'  
  
estimator = Estimator(  
    image_name=ecr_image,  
    role=sagemaker.get_execution_role(),  
    train_instance_count=1,  
    train_instance_type=train_instance_type,  
    hyperparameters={  
        'config': '/opt/ml/input/data/config/cifar10_sample.py',  
        'experiment_id': 'cifar10_sample'  
    })  
  
In []: estimator.fit({'dataset': train_data, 'config': config_data})
```

## Step 2

# Amazon SageMaker上での学習の実行

## SageMaker Processingを利用したon-demand処理

Processor.runで任意のコードを実行させることが可能です。今回はconvert用のpython scriptを実行しています。

### Convert

#### On Amazon Sagemaker on-demand instance

```
In [ ]: from sagemaker.processing import ScriptProcessor, ProcessingInput, ProcessingOutput

convert_instance_type = 'ml.m5.xlarge'

processor = ScriptProcessor(
    image_uri=ecr_image,
    role=sagemaker.get_execution_role(),
    command=['python3'],
    base_job_name="blueoil-convert",
    instance_count=1,
    instance_type=convert_instance_type)

In [ ]: trained_model = estimator.model_data
converted_model = os.path.join(os.path.dirname(trained_model), 'converted')

In [ ]: processor.run(code='script/main.py',
    inputs=[
        ProcessingInput(source=train_data, destination='/opt/ml/processing/input/data/dataset'),
        ProcessingInput(source=estimator.model_data, destination='/opt/ml/processing/input/data/model'),
    ],
    outputs=[
        ProcessingOutput(source='/opt/ml/processing/output/converted', destination=converted_model),
    ],
    arguments=['convert', '--experiment_id', 'cifar10_sample'],
)

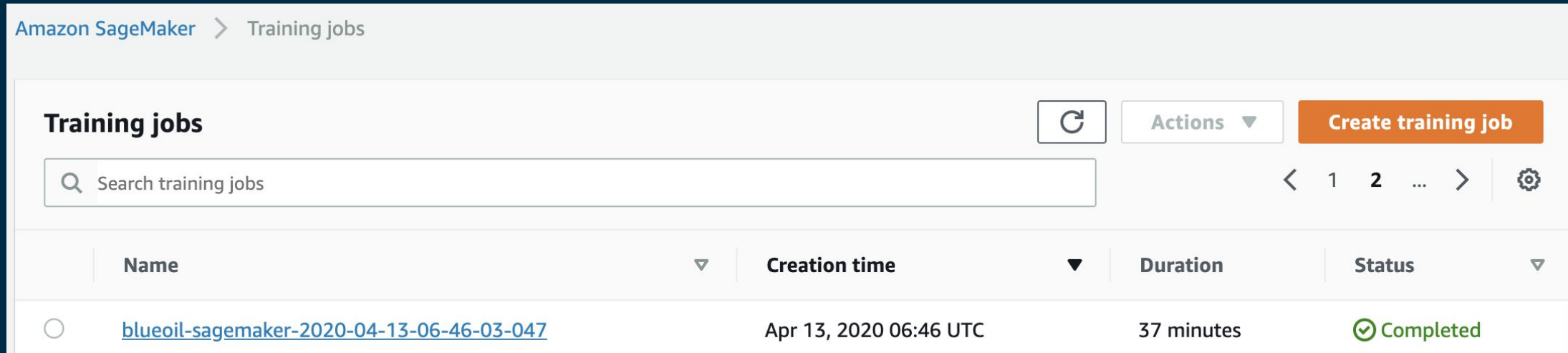
print(f"Converted models are saved to {converted_model}")
```

## Step 2

# Amazon SageMaker上での学習の実行

## 学習結果の確認

データセット・実験環境・実験コンフィグ・実験結果等の必要な情報が、実験を行う毎に Training Jobsという単位で管理されます。



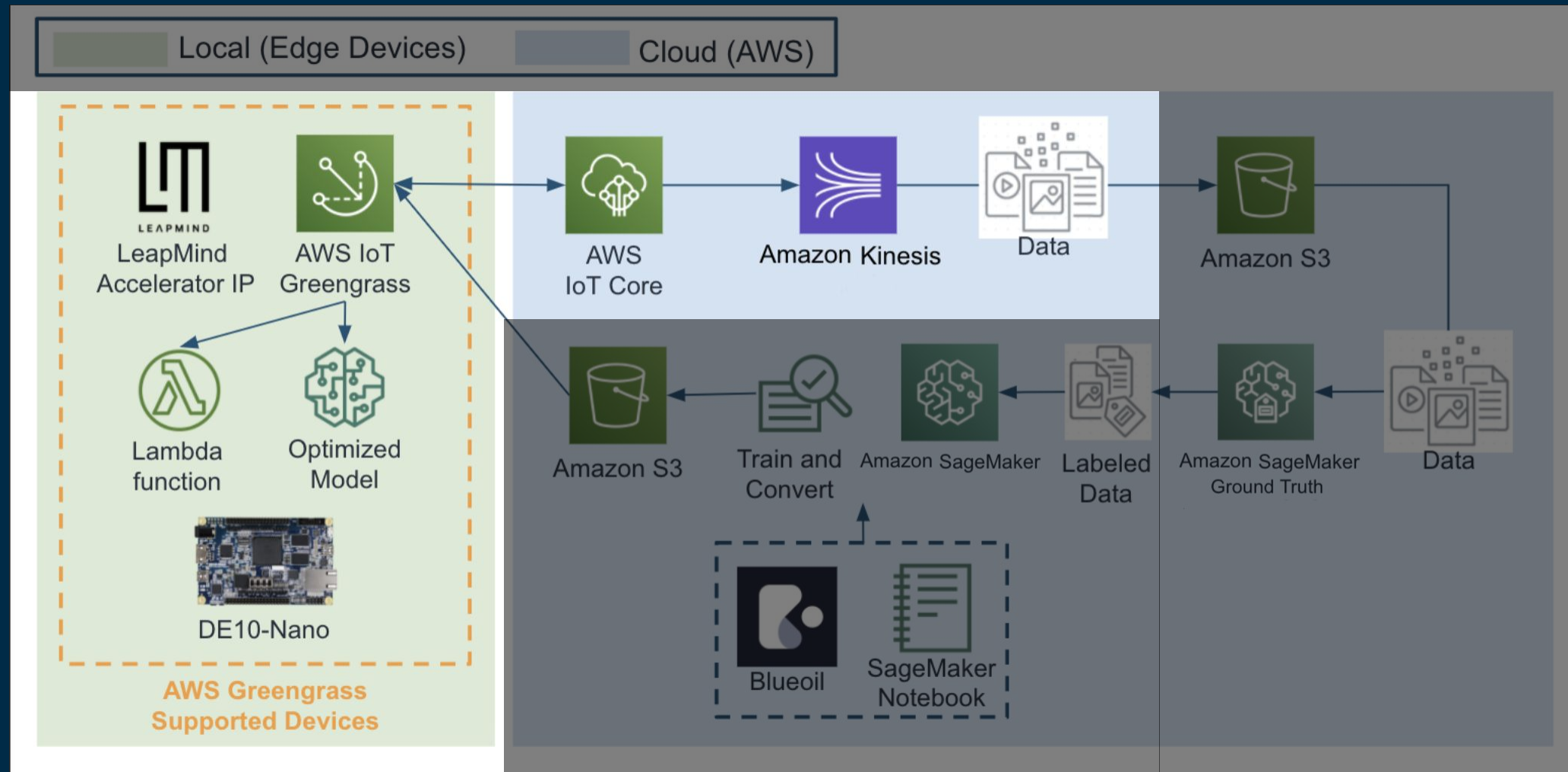
The screenshot shows the Amazon SageMaker console interface for Training Jobs. At the top, there is a breadcrumb navigation: "Amazon SageMaker > Training jobs". Below this, the "Training jobs" section is displayed. It includes a search bar with the placeholder text "Search training jobs", a refresh button, an "Actions" dropdown menu, and a prominent orange "Create training job" button. Below the search bar, there is a table with the following columns: "Name", "Creation time", "Duration", and "Status". A single training job is listed with the name "blueoil-sagemaker-2020-04-13-06-46-03-047", a creation time of "Apr 13, 2020 06:46 UTC", a duration of "37 minutes", and a status of "Completed" (indicated by a green checkmark icon).

- 実験結果はAmazon S3に保存
- 実験環境はコンテナ化することで再現性をもたせることが可能
- 実験時のログの保存や、実験実施中のリソース使用率(GPU/CPU/Mem)なども記録される

Step 3

# AWS IoT Greengrass Groupの作成

下記のコンポーネントを作成します。



### Step 3

# AWS IoT Greengrass Groupの作成

## Greengrass用の証明書の作成

下記のコマンドでそれぞれのDeviceに配置する証明書を作成しAWS上に登録します。

```
$ deploy/create_cert.sh
...
Certificate files are created in certs/xxxxxx...xxxxxx
certificateArn is "arn:aws:iot:xxxxxx:xxxxxx:cert/xxxxxx...xxxxxx"
```

今回の構成ではDE10-Nanoを2台紐付ける構成となっているため、それぞれのデバイス用に上記コマンドを2回実施し、得られたArnを環境変数として下記のように設定します。

```
$ export CERT_ARN1=[your certificateArn1]
$ export CERT_ARN2=[your certificateArn2]
```

Step 3

# AWS IoT Greengrass Groupの作成

## AWS IoT Greengrass Groupの作成

下記のコマンドでAWS IoT Greengrass Groupを作成します。

```
$ export MODEL_S3_URI=[your S3 uri]
$ aws cloudformation create-stack --stack-name Bluegrass --capabilities
CAPABILITY_NAMED_IAM --template-body "$(aws cloudformation package
--template-file deploy/greengrass.yaml --s3-bucket ${BLUEGRASS_S3_BUCKET_NAME})"
\
--parameters \
ParameterKey="ModelS3Uri",ParameterValue="${MODEL_S3_URI}" \
ParameterKey="S3BucketName",ParameterValue="${BLUEGRASS_S3_BUCKET_NAME}" \
ParameterKey="Core01CertificateArn",ParameterValue="${CERT_ARN1}" \
ParameterKey="Core02CertificateArn",ParameterValue="${CERT_ARN2}"
```

ここでMODEL\_S3\_URIはSageMakerで学習したモデルの場合、下記のようになります。

```
s3://sagemaker-xxxxxx/blueoil-sagemaker-2020-XX-XX-XX-XX-XX-XXZ/output/converted/output.tar.gz
```



## Step 3

# AWS IoT Greengrass Groupの作成

## AWS CloudFormation Templateの抜粋

- Core01Group

下記の定義と紐付けています。

- StreamConnectorDefinitionVersion
  - Kinesis Firehoseなど
- Core01DefinitionVersion
  - Deviceの証明書など
- InferenceFunctionDefinitionVersion
  - Lambda Functionなど
- ModelResourceDefinitionVersion
  - 機械学習モデルのS3パスなど
- StreamSubscriptionDefinitionVersion
  - MQTT Topic, Source, Targetなど

```
Core01Group:
  Type: 'AWS::Greengrass::Group'
  Properties:
    Name: Core01Group
    RoleArn:
      Fn::Join:
        - ':'
        - - 'arn:aws:iam:'
          - Ref: 'AWS::AccountId'
          - role/BluegrassRole
    InitialVersion:
      ConnectorDefinitionVersionArn:
        Ref: StreamConnectorDefinitionVersion
      CoreDefinitionVersionArn:
        Ref: Core01DefinitionVersion
      FunctionDefinitionVersionArn:
        Ref: InferenceFunctionDefinitionVersion
      ResourceDefinitionVersionArn:
        Ref: ModelResourceDefinitionVersion
      SubscriptionDefinitionVersionArn:
        Ref: StreamSubscriptionDefinitionVersion
```

## Step 3

# AWS IoT Greengrass Groupの作成

## AWS CloudFormation Templateの抜粋

- StreamSubscriptionDefinitionVersion

```
StreamSubscriptionDefinitionVersion:
  Type: 'AWS::Greengrass::SubscriptionDefinitionVersion'
  Properties:
    SubscriptionDefinitionId:
      Ref: StreamSubscriptionDefinition
    Subscriptions:
      - Id: InferenceResultToStream
        Source:
          Ref: BluegrassInferenceServerFunctionGGAlias
        Subject: kinesisfirehose/message
        Target:
          Fn::Join:
            - ':'
            - - 'arn:aws:greengrass'
              - Ref: 'AWS::Region'
              - ':/connectors/KinesisFirehose/versions/3'
```

- BluegrassInferenceServerFunction

```
BluegrassInferenceServerFunction:
  Type: 'AWS::Lambda::Function'
  Properties:
    FunctionName: "bluegrass_inference_server"
    Handler: "run_server.run_server_handler"
    Role:
      Fn::GetAtt:
        - "BluegrassInferenceServerExecutionRole"
        - "Arn"
    Code: ./lambda_function
    Runtime: "python2.7"
```



### Step 3

# AWS IoT Greengrass Groupの作成

## CloudFormation Templateに定義された全リソース

29個のリソースが作成されます。

Core01Group:  
Core02Group:

InferenceFunctionDefinition:  
InferenceFunctionDefinitionVersion:

ModelResourceDefinition:  
ModelResourceDefinitionVersion:

StreamConnectorDefinition:  
StreamConnectorDefinitionVersion:

StreamSubscriptionDefinition:  
StreamSubscriptionDefinitionVersion:

Core01:  
Core01Definition:  
Core01DefinitionVersion:  
Core01ThingPrincipalAttachment:  
Core01PolicyPrincipalAttachment:

Core02:  
Core02Definition:  
Core02DefinitionVersion:  
Core02ThingPrincipalAttachment:  
Core02PolicyPrincipalAttachment:

BluegrassPolicy:

BluegrassInferenceServerExecutionRole:  
BluegrassInferenceServerFunction:  
BluegrassInferenceServerFunctionFirstVersion:  
BluegrassInferenceServerFunctionGGAlias:

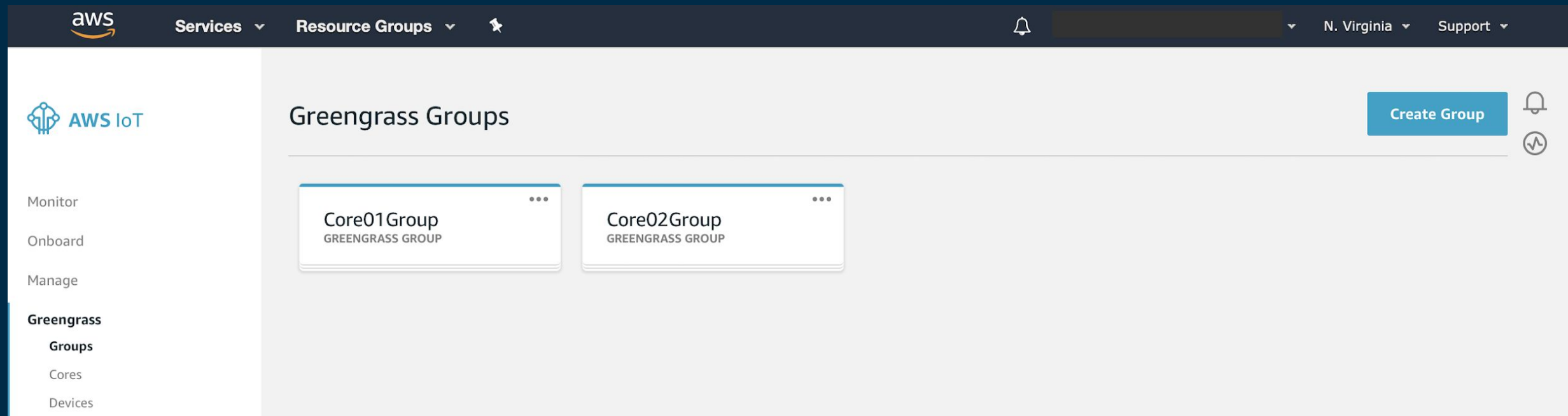
BluegrassStreamingRole:  
BluegrassStreamingS3Policy:  
BluegrassStreamingPolicy:  
BluegrassInferenceStream:

Step 3

# AWS IoT Greengrass Groupの作成

## AWS IoT Greengrass Groupの確認

下記のようにAWS IoTコンソールにて作成されたGreengrass Groupが確認できます。

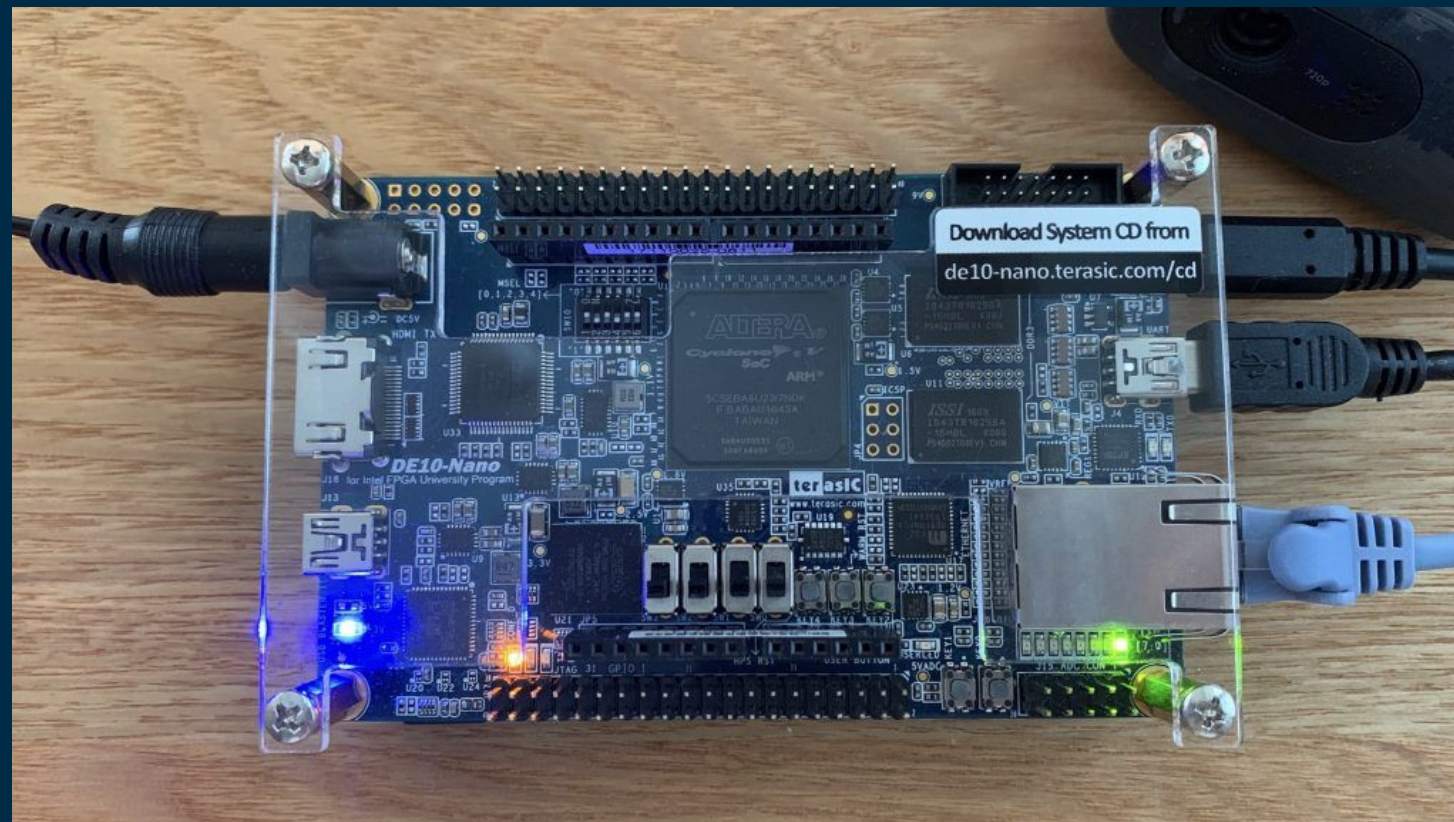


## Step 4

# Deviceのセットアップ

## DE10-Nanoのセットアップ

公開されているmicroSDカードイメージを書き込んだSDカードを挿入し、下記のようにEthernet Cable及び電源を接続してデバイスを起動します。起動後にNW設定等を実施し、固定アドレスでssh loginができるように設定します。詳細は[AWS Blog](#)を参照。



## Step 4

# Deviceのセットアップ

## DE10-Nanoのセットアップ

下記のコマンドでGreengrassのインストール及びBlueoil用の推論環境のセットアップを実施します。

```
$ ansible-playbook -i [IP address of Core1],[IP address of Core2]  
ansible/playbook.yml
```

次に証明書の配置を行います。これにてセットアップは完了です。

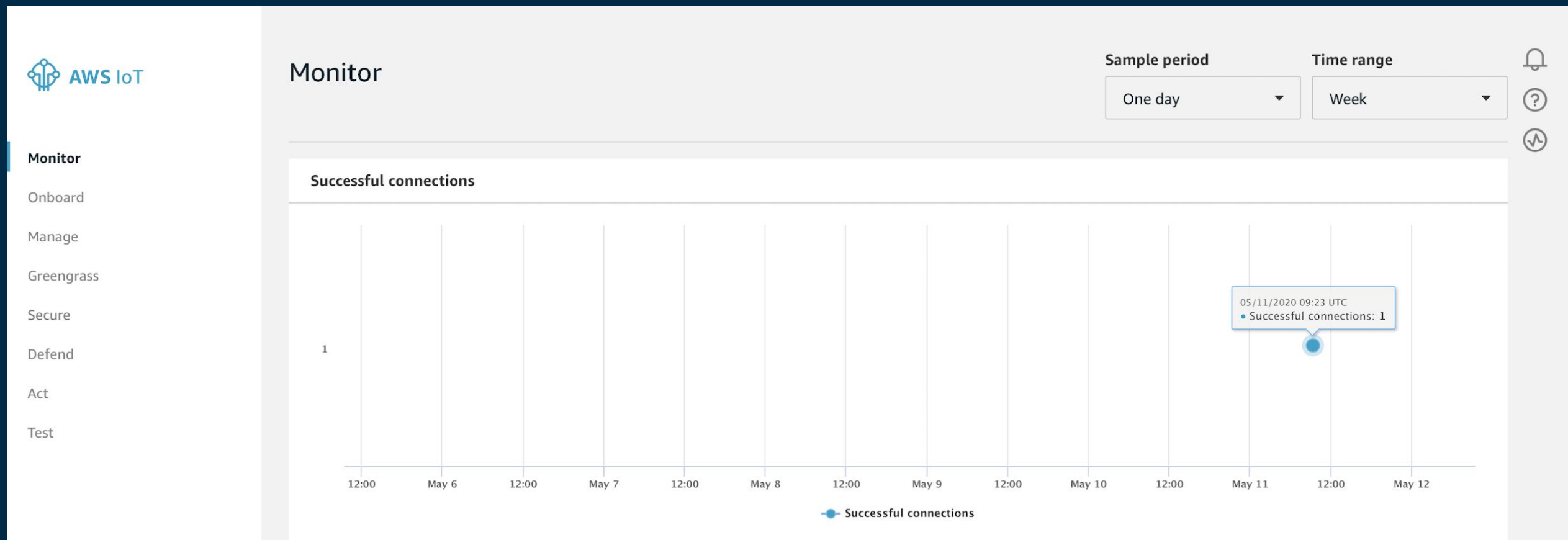
```
$ export CERT_ID1=$(echo ${CERT_ARN1} | sed -e 's/.*\///g')  
$ export THING_ARN1=[thing arn of your Core1]  
$ export IOT_HOST=$(aws iot describe-endpoint --endpoint-type iot:Data-ATS  
--query 'endpointAddress' --output=text)  
$ export GG_HOST=$(echo ${IOT_HOST} | sed -e 's/.*-ats/greengrass-ats/g')  
  
$ ansible-playbook -i [IP address of Core1], ansible/playbook_certs_deploy.yml  
--extra-vars "cert_id=${CERT_ID1} thing_arn=${THING_ARN1} iot_host=${IOT_HOST}  
gg_host=${GG_HOST}"
```

Step 4

# Deviceのセットアップ

## AWS IoT コンソールから接続確認

下記のようにAWS IoTのコンソール上から接続が確認できます。



Step 5

# Application Deployと推論実施

## AWS IoT コンソールからDeploy

AWS IoTのコンソール上から対象のGroupを選択し、下記のように、ActionsからDeployの実行を行います。

The screenshot shows the AWS IoT console interface for a Greengrass group named 'Core01 Group'. The group status is 'Not deployed'. On the left, there is a navigation menu with 'Deployments' selected. The main content area shows 'Group history overview' with a 'By deployment' filter. A message states 'There are no deployments for this Greengrass Group yet'. An 'Actions' dropdown menu is open, showing three options: 'Deploy' (highlighted with an orange underline), 'Delete Group', and 'Reset Deployments'. The entire 'Actions' menu is enclosed in an orange rectangular box.

Step 5

# Application Deployと推論実施

AWS IoT コンソールからDeploy

In progress → Successfully completed となればDeploy完了です。

GREENGRASS GROUP

## Core01Group

● In progress Actions ▾

Deployments Group history overview By deployment ▾

	Deployed	Version	Status
Devices	May 12, 2020 6:33:07 PM +0900	[REDACTED]	● In progress
Devices	May 12, 2020 6:33:07 PM +0900	[REDACTED]	● Successfully completed

## Step 5

# Application Deployと推論実施

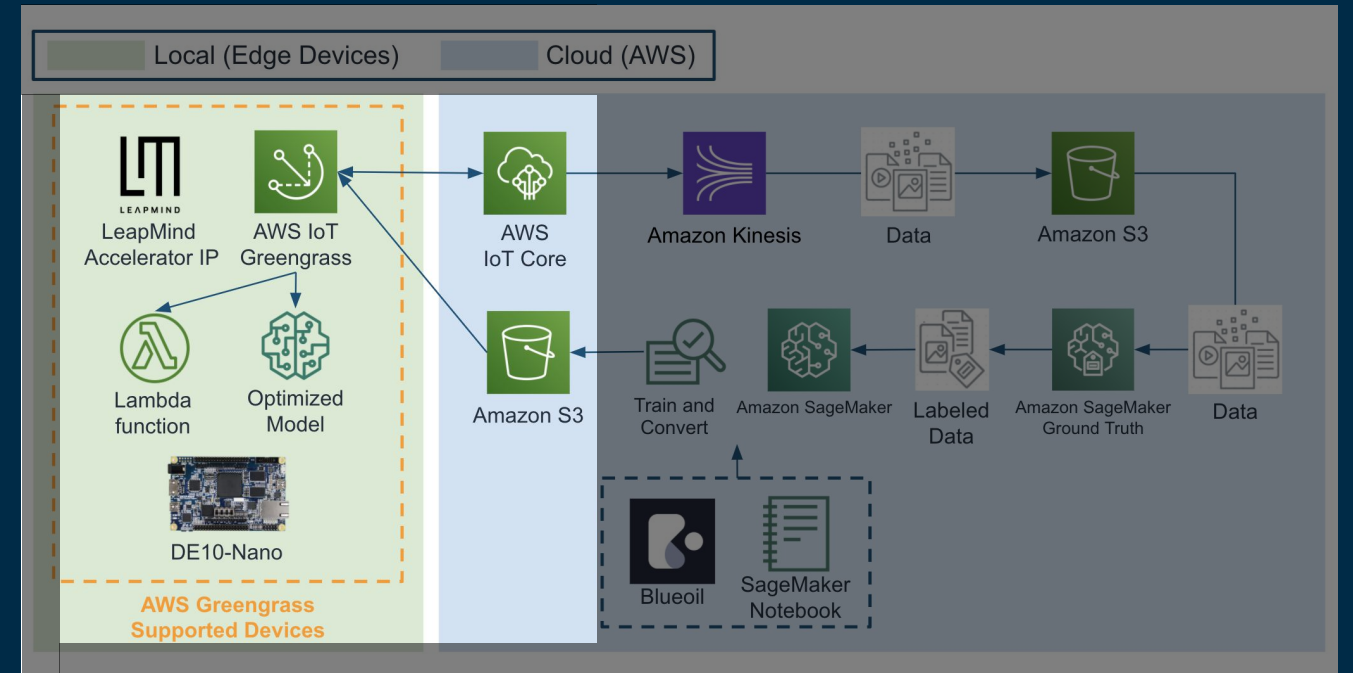
## Deploy application to FPGA devices

デプロイ対象のモデルやアプリケーションは下記のように管理されます。

- 学習済みモデル ← S3 from SageMaker
- 推論アプリ ← AWS Lambda Function

上記を含む”Deployment”をデバイスごとに適用することで、Cloudからデバイス上アプリケーション全てを管理可能

- Deploymentは各デプロイ作業ごとにVersioning  
= 過去の特定のバージョンへの切り戻しが容易
- デプロイ後のアプリはスタンドアロンで動作。また、新たなデプロイ情報は次にデバイスがIoT Coreに接続された際に伝播する。  
= Networkに常に接続されていない状態でも動作



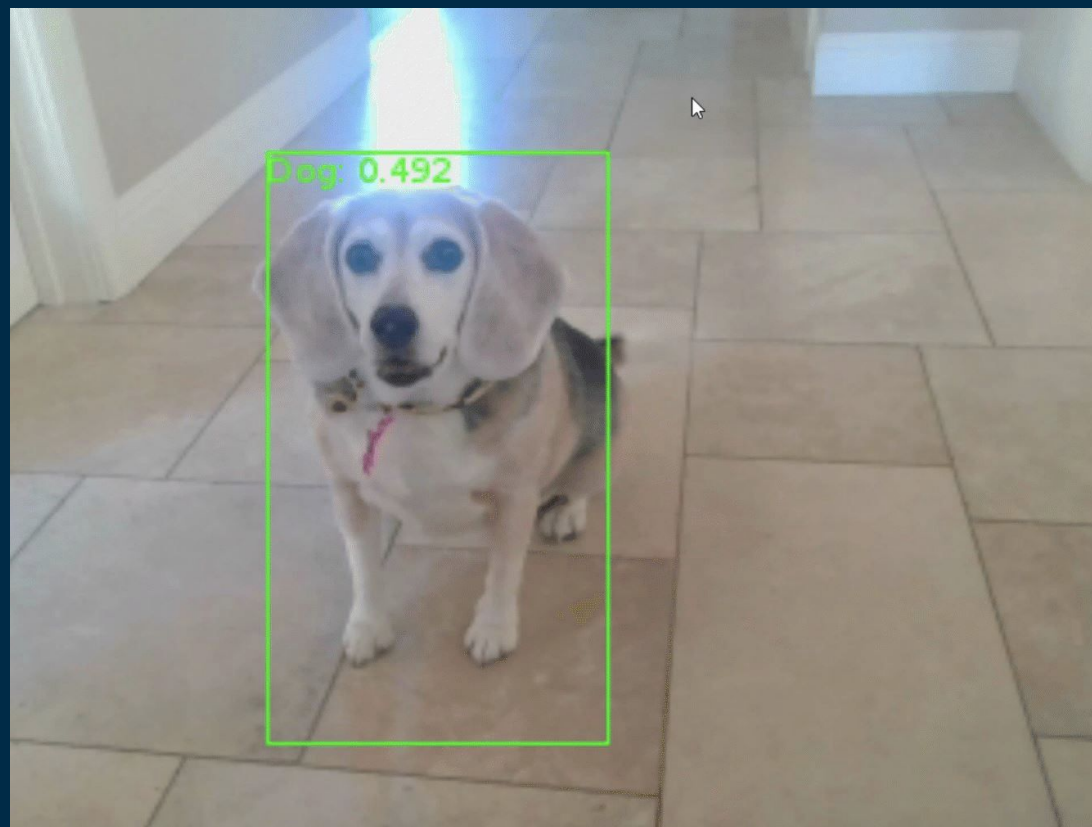


Step 5

# Application Deployと推論実施

## Device上の推論アプリケーションの確認

http://<deviceのIP address>:8080にブラウザ経由でアクセスすることで、カメラから取得した画像のリアルタイムな推論結果を確認することができます。



Step 5

# Application Deployと推論実施

## 推論結果の確認

AWS IoT コンソールから推論結果が確認できます。

The screenshot shows the AWS IoT console interface for an MQTT client. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', a notification bell, the region 'N. Virginia', and 'Support'. The left sidebar lists navigation options: Monitor, Onboard, Manage, Greengrass, Secure, Defend, Act, and Test. The main content area is titled 'MQTT client' and shows a 'Subscriptions' section. Under 'Subscriptions', there are links for 'Subscribe to a topic' and 'Publish to a topic'. Below these links, there is a 'Subscribe' section with a text input field containing 'inference/result' and a 'Subscribe to topic' button. There is also a 'Max message capture' section with a dropdown menu set to '100'.

Step 5

# Application Deployと推論実施

## 推論結果の確認

AWS IoT コンソールから推論結果が確認できます。

```
inference/result          May 12, 2020 6:40:01 PM +0900          Export Hide

{
  "benchmark": {},
  "classes": [
    {
      "id": 0,
      "name": "face"
    }
  ],
  "date": "2020-05-12T09:40:01.486344",
  "results": [
    {
      "file_path": null,
      "prediction": [
        {
          "box": [
            158.37029266357422,
            38.650482177734375,
            131.25941467285156,
```

## Step 5

# Application Deployと推論実施

## Aggregate data from devices

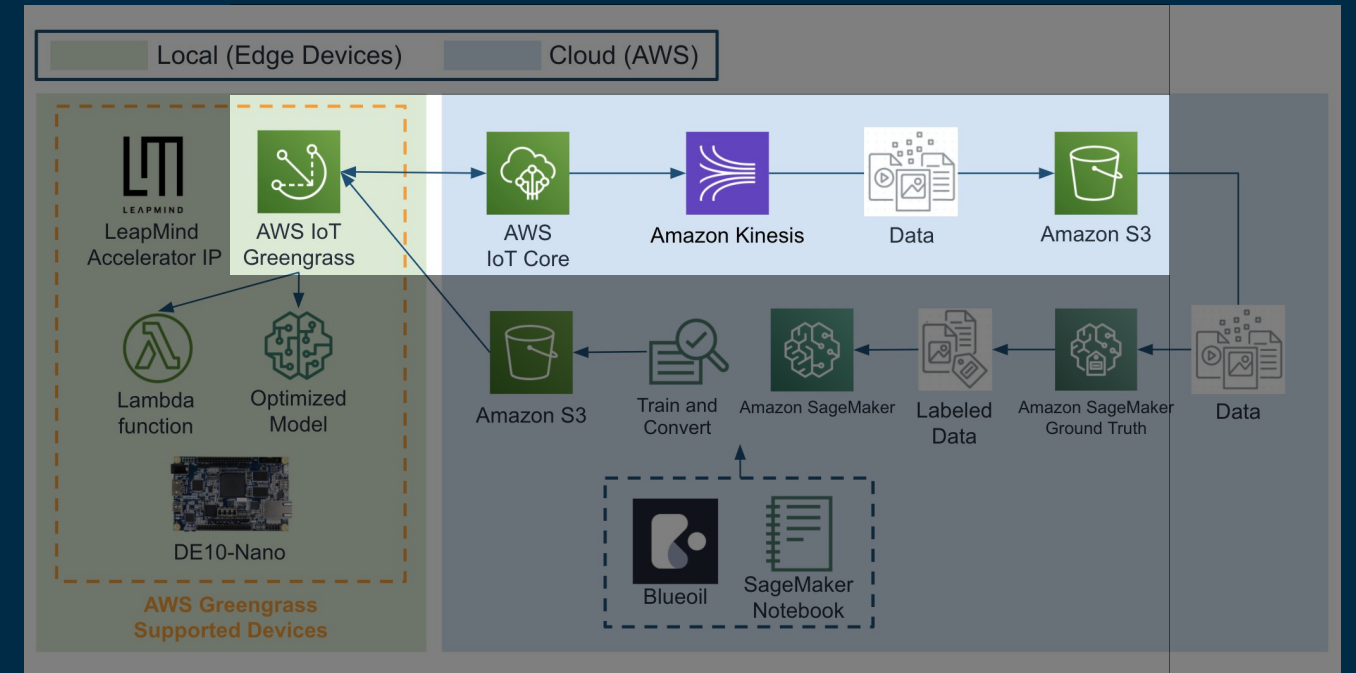
データの収集方法は下記のような方法があります

### Greengrass Connector

- Connectorを通じて、様々なAWSサービスやサードパーティコンポーネントとデータの連携が可能
- Kinesis Firehoseコネクタを使うと、Amazon Kinesis Data Firehose 配信ストリームを通じて、Amazon S3などにデータを送信できる

### Stream Manager for AWS IoT Greengrass

- Networkに接続されていない・断続的な接続が発生するようなシナリオにおいてもデータストリームを管理可能
- データはまずローカルのストリームマネージャーに処理/保存され、ユーザーの定義した優先度・タイムアウトにしたがってNetworkアクセス時にCloud側に送信される



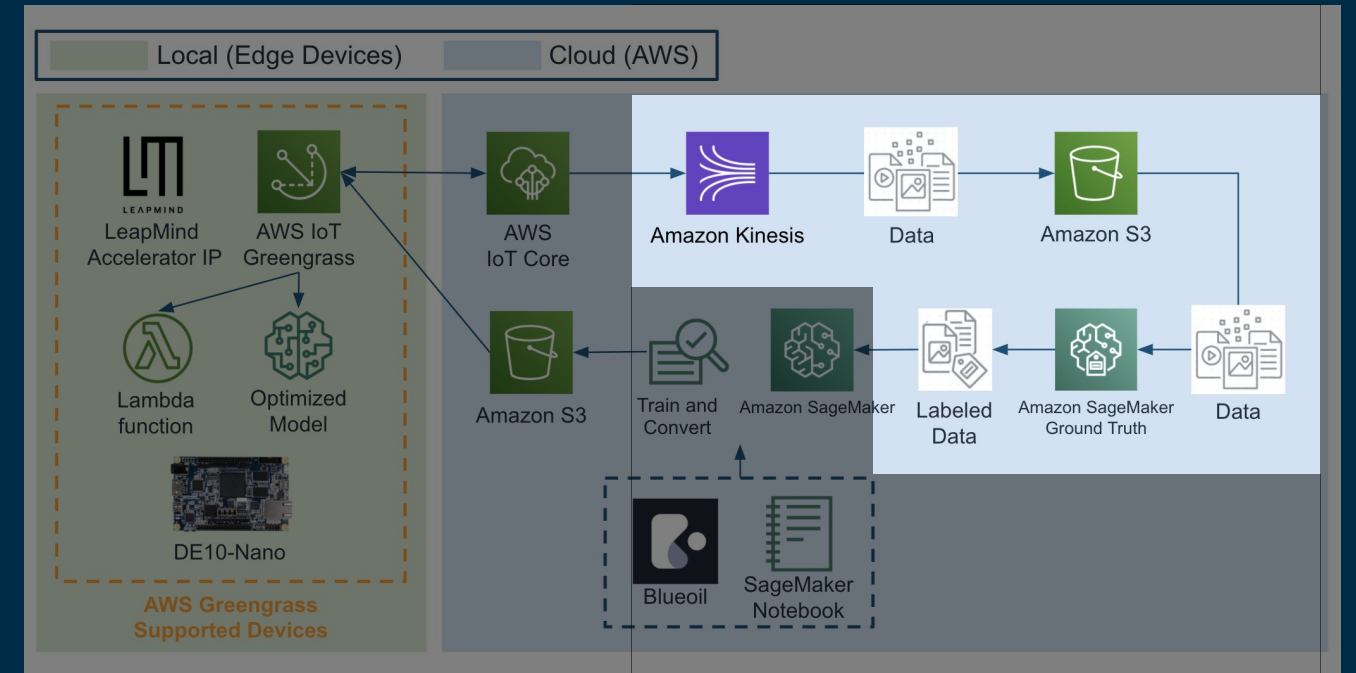
Step 5

# Application Deployと推論実施

## Aggregate data from devices

### データの保存・再利用

- 前述のような機能を利用し、有用なデータを適切な形式で Amazon S3に保存
- 取得データを再学習用のデータセットとして利用可能
  - SageMaker Ground Truthを利用してデータのアノテーションを実施



# まとめ

Amazon SageMakerとAWS IoT Greengrassを使えば  
**Edge Deep Learningの運用環境を  
Managedな形で簡単に構築することができる**

自前構築と比較して

導入コスト低

運用コスト低

スモールスタート可能

**ぜひ下記を参考に手元で試してみてください！**

<https://github.com/LeapMind/bluegrass>

[\[和訳\] 効率的にインテル® FPGA エッジデバイス上の深層学習推論を実行する](#)



Thank you.