Amazon Connect　機能開発

（要件ヒアリングから開発・テスト・本番展開を実施）

目次

01

インターンシップを通じて

インターンシップを通じて

できたこと
- Amazon Connectの基本的機能の実装
- Lambdaを用いた独自の機能の開発
- DynamoDBの効果的なテーブル設計
- AWSサービスを利用した一連の機能開発

得られたこと
- CloudFormationを使った容易な環境展開

できなかったこと
- システムの利用者目線に立った機能開発

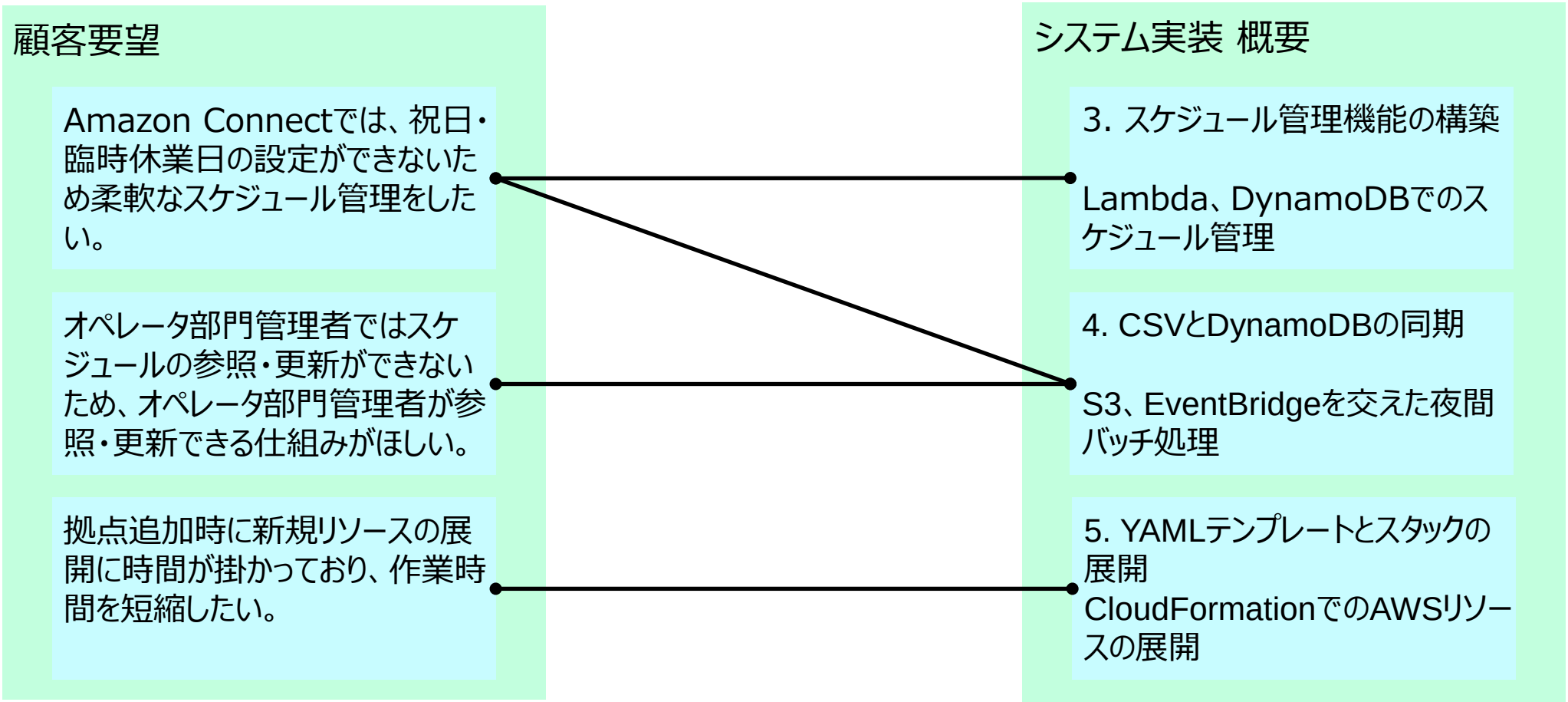
気付いたこと
- 単体テストの重要性
- Lambdaの利便性
- Infrastructure as Codeの有効性

02

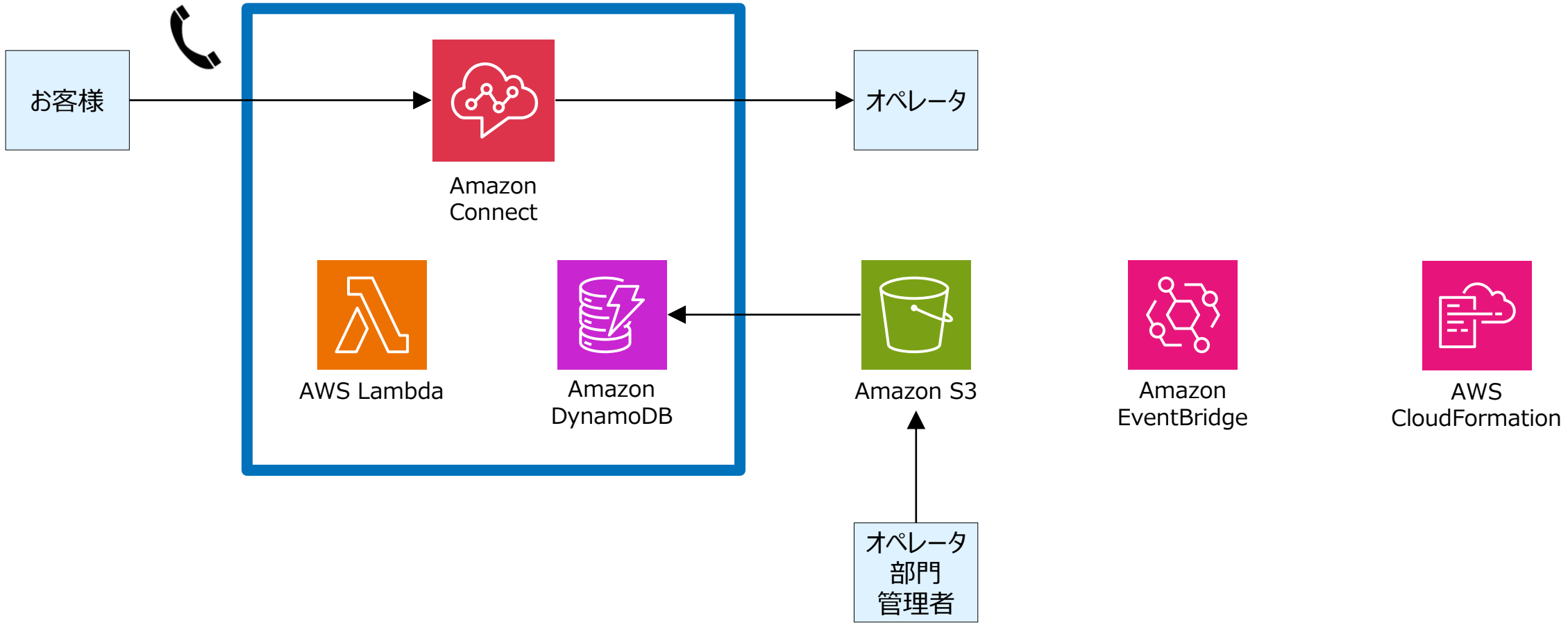
要求とシステム

インターン期間の課題

テーマ：
　顧客からの要望に基づき、要件ヒアリングから開発・テスト・本番展開までの一連の流れを経験する



全体構成


お客様
Amazon
Connect
オペレータ
AWS Lambda
Amazon
DynamoDB
Amazon S3
Amazon
EventBridge
AWS
CloudFormation
オペレータ
部門
管理者

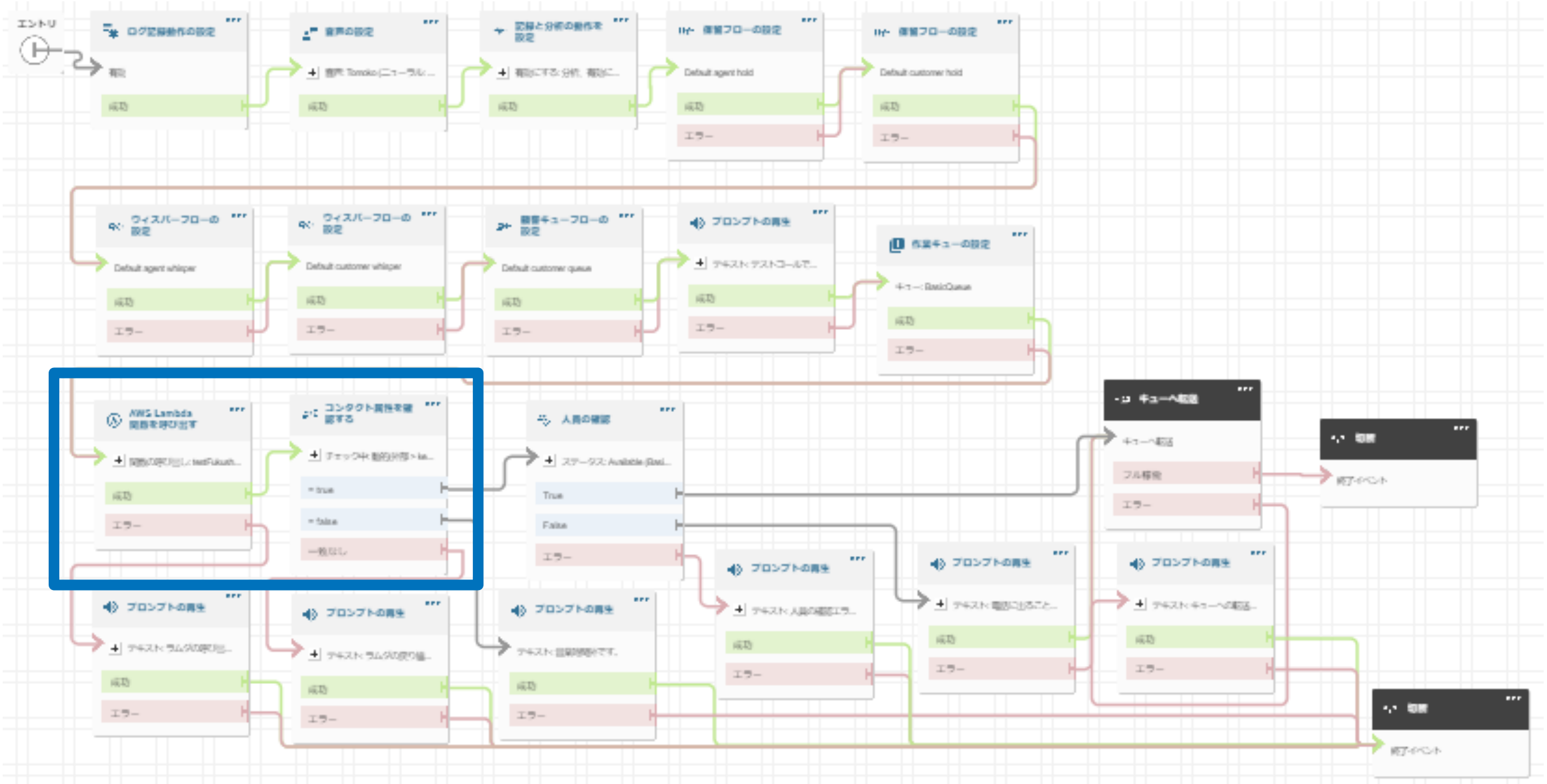
03

スケジュール管理機能の構築

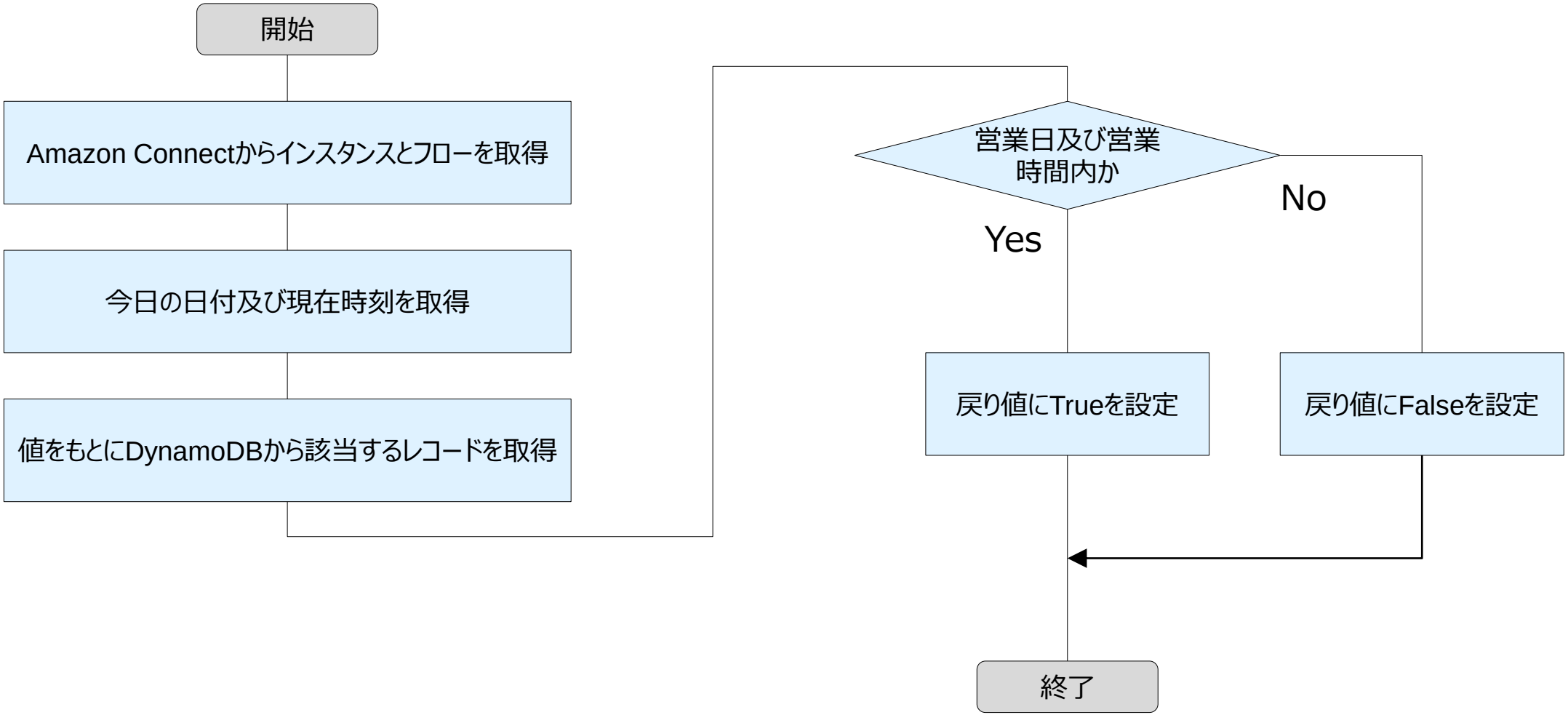
Amazon Connectの設定

- 新規ユーザ登録、エージェントステータスの追加
- オペレーション時間の設定
- ルーティングプロファイルの作成
- キューに発信電話番号及び発信フローの割り当て
- 着信用電話番号に着信フローの割り当て

Amazon Connect 着信フロー



Lambdaのフローチャート



DynamDBのテーブル設計

NO	項目名(日本語)	項目名(英語)	属性	キー項目	項目説明
1	キュー	queue	文字列	パーティションキー	AmazonConnectのキューを識別。
2	日付	day	文字列	ソートキー	一意の日付を設定。
3	休日判定	isHoliday	BOOL		休日かどうか判定する。 休日の場合はtrue。
4	臨時休業日判定	isTempHoliday	BOOL		臨時休業日かどうか判定する。 臨時休業日の場合はtrue。
5	臨時休業時間開始	startTempClosingTime	文字列		臨時休業時間の開始時刻。
6	臨時休業時間終了	endTempClosingTime	文字列		臨時休業時間の終了時刻。

単体テスト

テストケース番号	テスト内容	設定内容	期待される結果	テスト結果
1	平日 営業時間内		戻り値がTrue	OK
2	平日 営業時間外	環境変数でテスト実施時間を営業時間外に設定	戻り値がFalse	OK
3	祝日	DynamoDBでテスト実施日を祝日に設定	戻り値がFalse	OK
4	臨時休業日	DynamoDBでテスト実施日を臨時休業日に設定	戻り値がFalse	OK
5	臨時休業時間内	DynamoDBでテスト実施時間を臨時休業時間内に設定	戻り値がFalse	OK
6	臨時休業時間外	DynamoDBでテスト実施時間を臨時休業時間外に設定	戻り値がTrue	OK

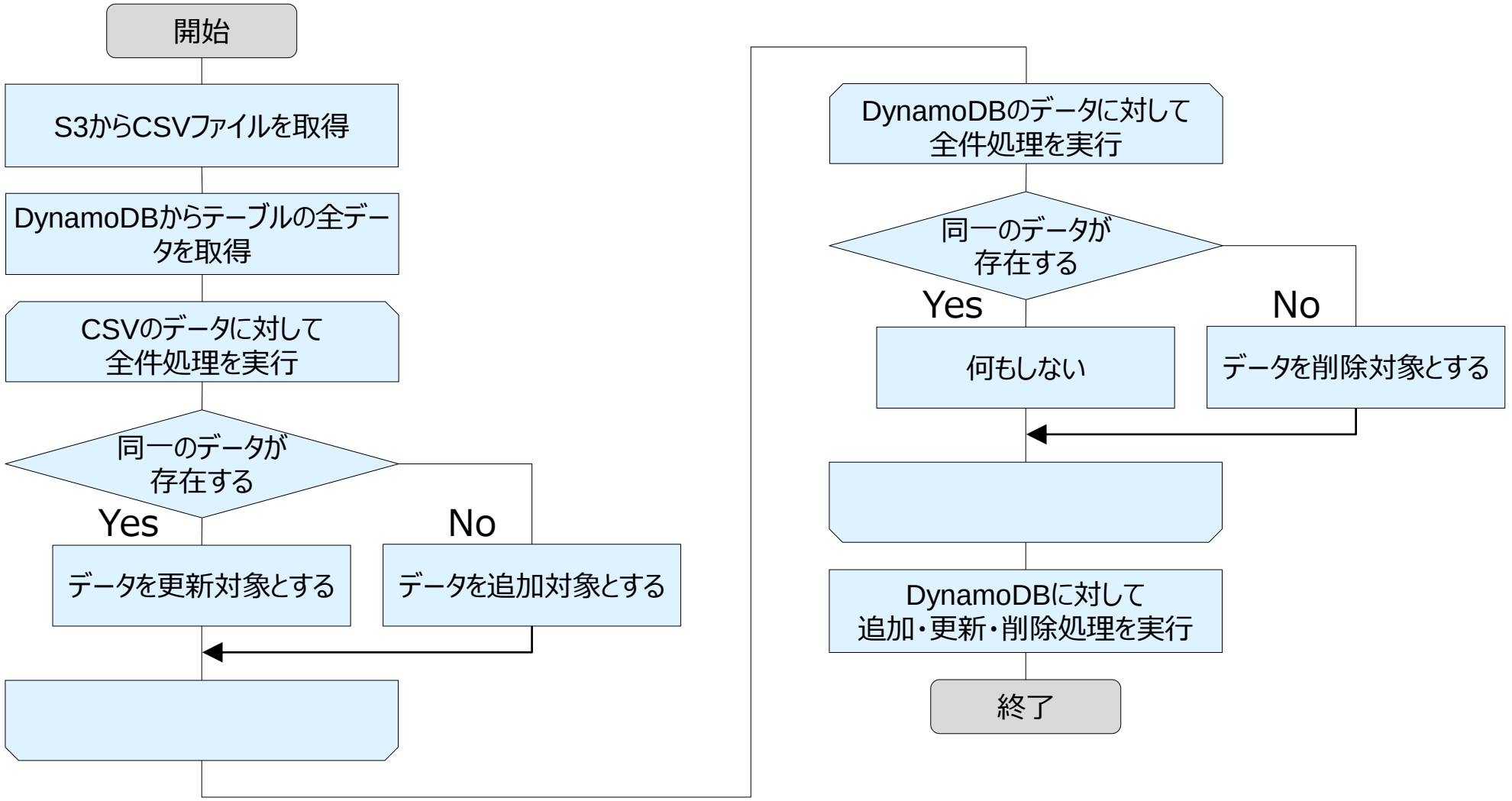
その他

- 言語：Python
- 環境変数の設定
- IAMロールの適用と適切なIAMポリシーの作成

04

CSVとDynamoDBの同期

Lambdaのフローチャート



CSVファイルのデータ構造設計

NO	項目名(日本語)	項目名(英語)	属性	キー項目	項目説明
1	キュー	queue	文字列		AmazonConnectのキューを識別。
2	日付	day	文字列		一意の日付を設定。
3	休日判定	isHoliday	文字列		休日かどうか判定する。 休日の場合はtrue。
4	臨時休業日判定	isTempHoliday	文字列		臨時休業日かどうか判定する。 臨時休業日の場合はtrue。
5	臨時休業時間開始	startTempClosingTime	文字列		臨時休業時間の開始時刻。
6	臨時休業時間終了	endTempClosingTime	文字列		臨時休業時間の終了時刻。

Amazon EventBridge

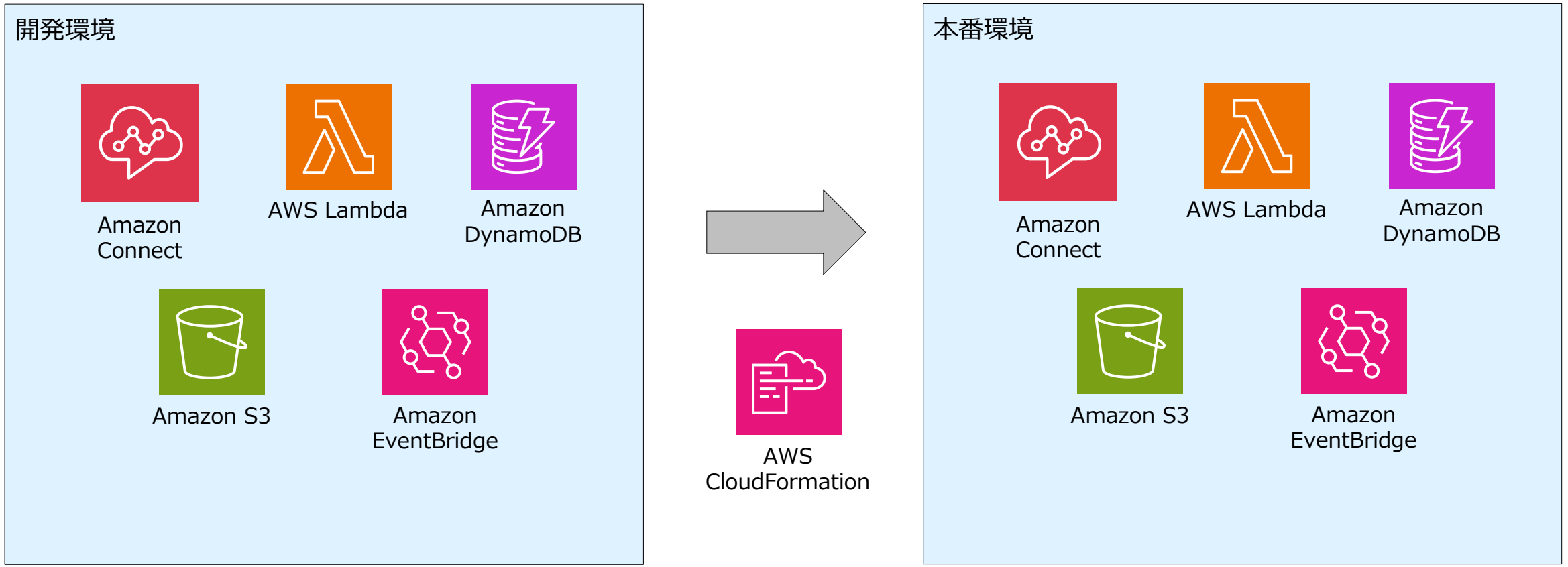
- スケジュールの作成
- cron式の適用
- Lambda関数の登録

05

YAMLテンプレートとスタックの展開

本番展開を想定した移行を実施

- DynamoDBテーブル：スケジュール情報を保持
- Lambda関数：スケジュールを管理
- S3バケット： CSVファイルをアップロード
- Lambda関数： CSVファイルとDynamoDBの同期
- EventBridge：夜間バッチ処理を実行
- Amazon Connectの着信フロー



DynamoDBテーブルのYAMLテンプレート

```yaml
AWSTemplateFormatVersion: "2010-09-09"

Resources:
  myDynamoDBTable:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        - AttributeName: queue
          AttributeType: S
        - AttributeName: day
          AttributeType: S
      KeySchema:
        - AttributeName: queue
          KeyType: HASH
        - AttributeName: day
          KeyType: RANGE
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 5
      TableName: "intellilink-internship-2023-summer-test_rest_biz"
      Tags:
        - Key: "intern"
          Value: "intern"
```

06

振り返り

振り返り

できたこと
- Amazon Connectの基本的機能の実装
- Lambdaを用いた独自の機能の開発
- DynamoDBの効果的なテーブル設計
- AWSサービスを利用した一連の機能開発

得られたこと
- CloudFormationを使った容易な環境展開

今後も継続していきたいこと

- **未知の領域への挑戦**
- **システムの全体像を見渡すこと**

できなかったこと
- システムの利用者目線に立った機能開発

気付いたこと
- 単体テストの重要性
- Lambdaの利便性
- Infrastructure as Codeの有効性

今後取り組んでいきたいこと

- **素早くPDCAサイクルを回すこと**
- **プログラミング及びシステム開発への深い理解**