Amazon Connect
カスタム CCP による従業員体験の向上

AWS Black Belt Online Seminar

清水 幸典
Connect Specialist SA

小柳 ちか子
Prototyping Engineer

2023/11

アジェンダ

1. コンタクトセンターの課題に対する技術的アプローチ
2. Demo：カスタム CCP を使用した顧客対応
3. サンプルカスタム CCP のアーキテクチャ解説
4. まとめ

本セミナーの対象者

- Amazon Connect カスタム CCP の具体例を知りたい方
- フロントエンド/バックエンド開発の基本的な知識のある方

自己紹介

名前：清水　幸典（しみず　ゆきのり）

所属：アマゾン ウェブ サービス ジャパン合同会社

CX ビジネス事業本部 Connect Specialist SA

経歴：

国内 SIer にてシステム開発、ビデオ会議ベンダー/セキュリティベンダーにてプリセールスエンジニアを経験

好きな AWS サービス：Amazon Connect, AWS Ground Station

コンタクトセンターの課題に対する技術的アプローチ

コンタクトセンターにおける顧客体験/従業員体験の課題

エージェント

コンタクトセンターにおける従業員体験の課題

エージェント

従業員体験を低下させる要因

- 独立した多数のツール
- 単純な問い合わせの繰り返し
- フォローアップ作業の負荷
- 顧客のクレームによる長時間対応
- タイムリーな支援の欠如
- 定量的でない評価

従業員体験が低下した結果

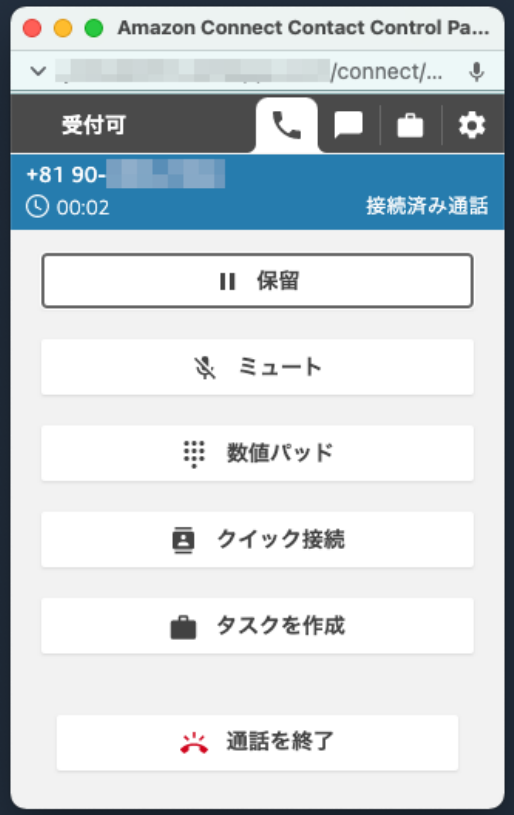
- 優秀な人材の流出
- 雇用の確保が困難
- 顧客体験の低下
- ビジネス状況の悪化

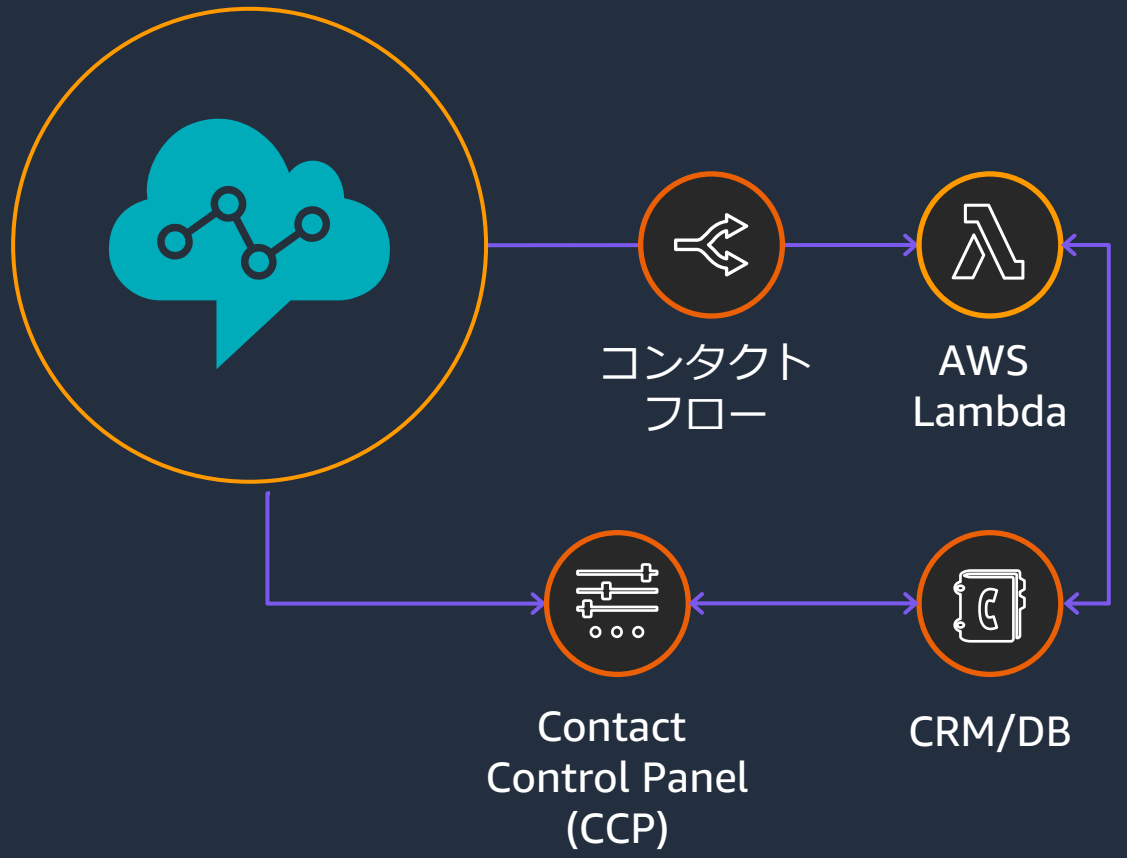
Amazon Connect の特徴

利用した価値に応じた従量課金制

API による柔軟な連携：カスタマイズ可能なソフトフォン

ブラウザベースのソフトフォン
(Contact Control Panel = CCP)





プログラマブル・コンタクトセンター

Amazon Connect

Amazon Connect APIs

ユーザ管理

ルーティングプロファイル

キュー

コンタクトフロー

インスタンス

クイック接続

ユーザ階層

オペレーション時間

セキュリティプロファイル

エージェントステータス

AWS CloudFormation

Amazon Connect と AWS サービスの統合による機能拡張

Development

AWS Lambda

Amazon API Gateway

AWS EventBridge

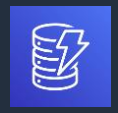
AWS Amplify

Storage

Amazon S3

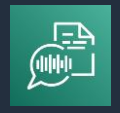
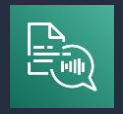
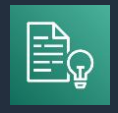
Amazon Glacier

Database

Amazon RDS

Amazon DynamoDB

Amazon Redshift

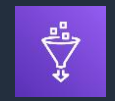
AI/ML

Amazon Lex

Amazon Transcribe

Amazon Polly

Amazon Comprehend


Amazon Connect

Analytics

Amazon Athena

Amazon Kinesis

Amazon Glue

Amazon QuickSight

Security

Amazon Cognito

AWS WAF

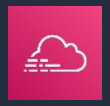
AWS Identity and Access Management

Network

Amazon CloudFront

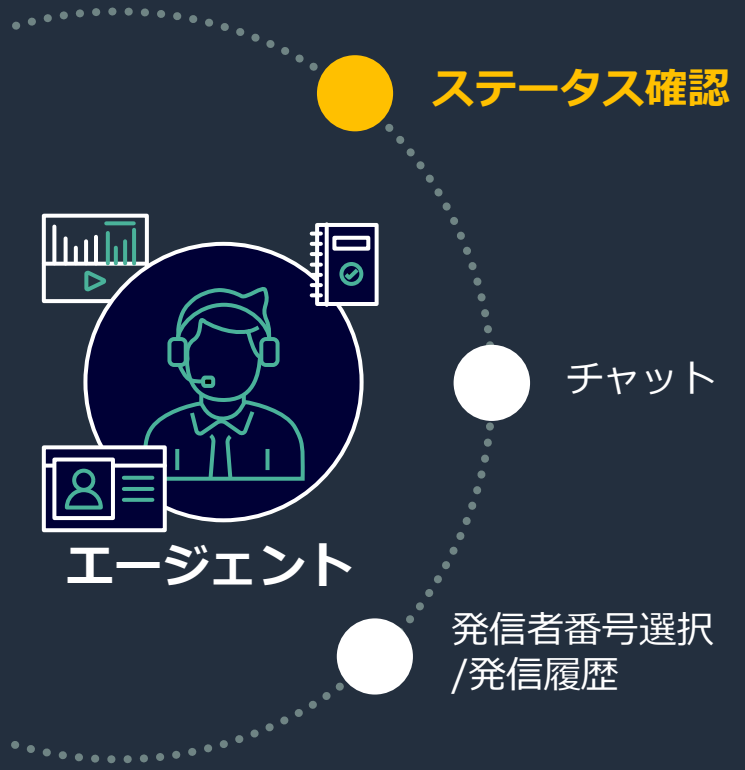
Management

Amazon CloudWatch

AWS CloudFormation

AWS CloudTrail

Demo :
カスタム CCP を
使用した顧客対応

ユースケース：ステータス確認



ご要望

- 他のエージェントの状況を把握してから通話を転送したい
- スーパーバイザーや管理者の状況を把握しつつ、モニタリングやバージインの依頼をしたい

開発で改善可能な体験

- **Amazon Connect のメトリクスからエージェントのステータスを取得してソフトフォンに表示**
- 通話中、休憩中、アフターコールワーク中、などの状況を把握し、転送失敗や待ち時間を削減する
- 作業効率の向上を実現

ユースケース：チャット

ご要望

- 顧客対応中や通話待機中に他のエージェントやスーパーバイザーと連絡をとりたい
- 顧客対応中に状況の共有や、在庫確認などの簡単な問い合わせを行いたい

開発で改善可能な体験

- **ソフトフォン内にチャットインタフェースを実装**
- CCP と併用可能なため、通話中も他のエージェントやスーパーバイザーと連絡可能
- コミュニケーションの円滑化と生産性向上を実現

ユースケース：発信者番号選択/発信履歴

ご要望

- アウトバウンドコール時、顧客に通知する番号を Amazon Connect の番号から任意に選択したい
- 発信履歴から再度コールしたい

開発で改善可能な体験

- **発信者番号選択/発信履歴のインタフェースを実装**
- Amazon Connect のアウトバウンドキューなどから発信者番号を選択
- 顧客に安心感を与えるアプローチが可能
- 従業員の架電作業を効率化

サンプルカスタム CCP のアーキテクチャ解説

自己紹介

名前：小柳 ちか子（おやなぎ ちか子）

所属：アマゾン ウェブ サービス ジャパン合同会社

プロトタイプエンジニアリング本部

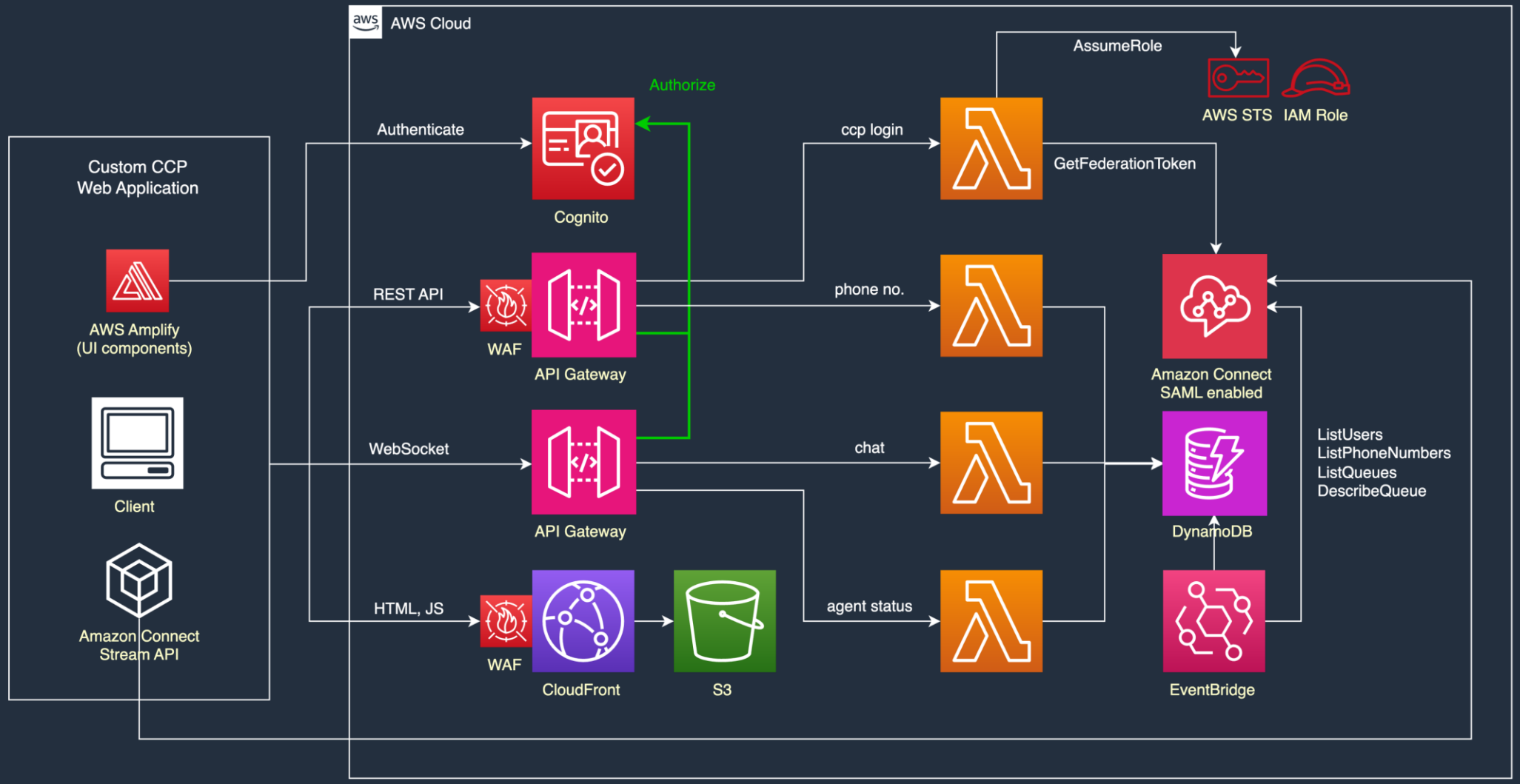
Prototyping Engineer

経歴：

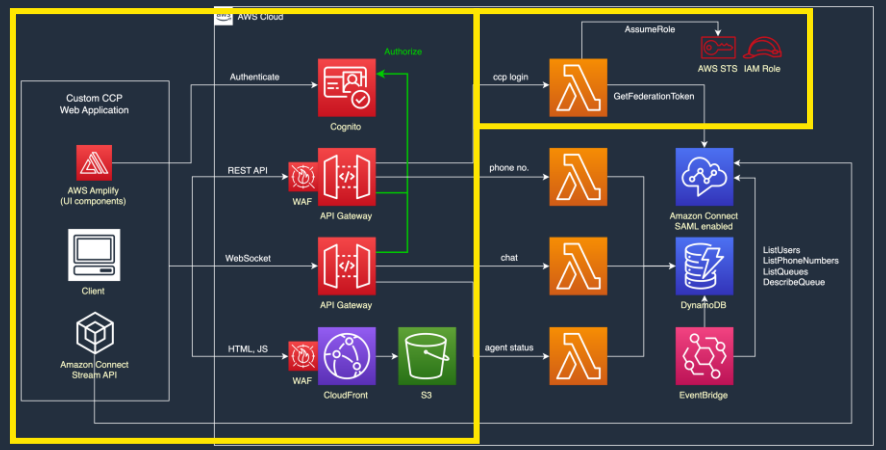
前職では自社エンタープライズ向けアプリケーションのフロントエンド開発と UI デザインに従事

好きな AWS サービス：Amazon Connect, AWS CDK

カスタム CCP アーキテクチャ全体像



カスタム CCP アーキテクチャ全体像

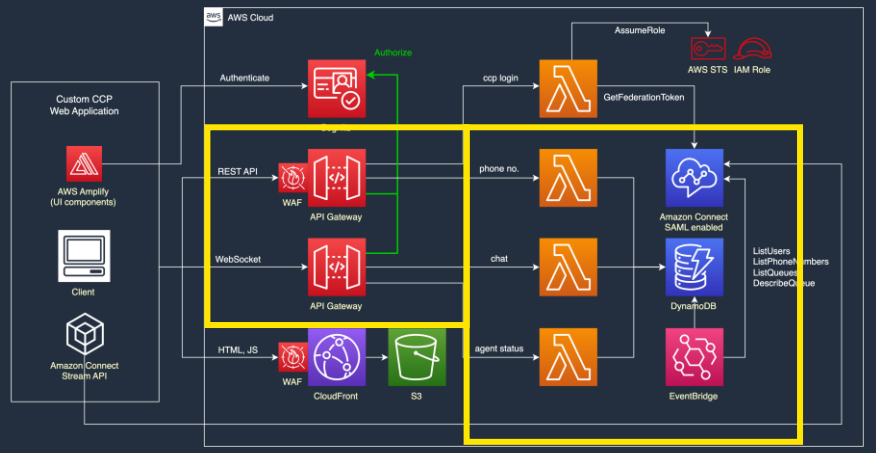


- フロントエンド
  - **CloudFront** + **S3** による静的ウェブサイト配信
  - Amazon Connect Streams API を利用しエージェントステータスを取得

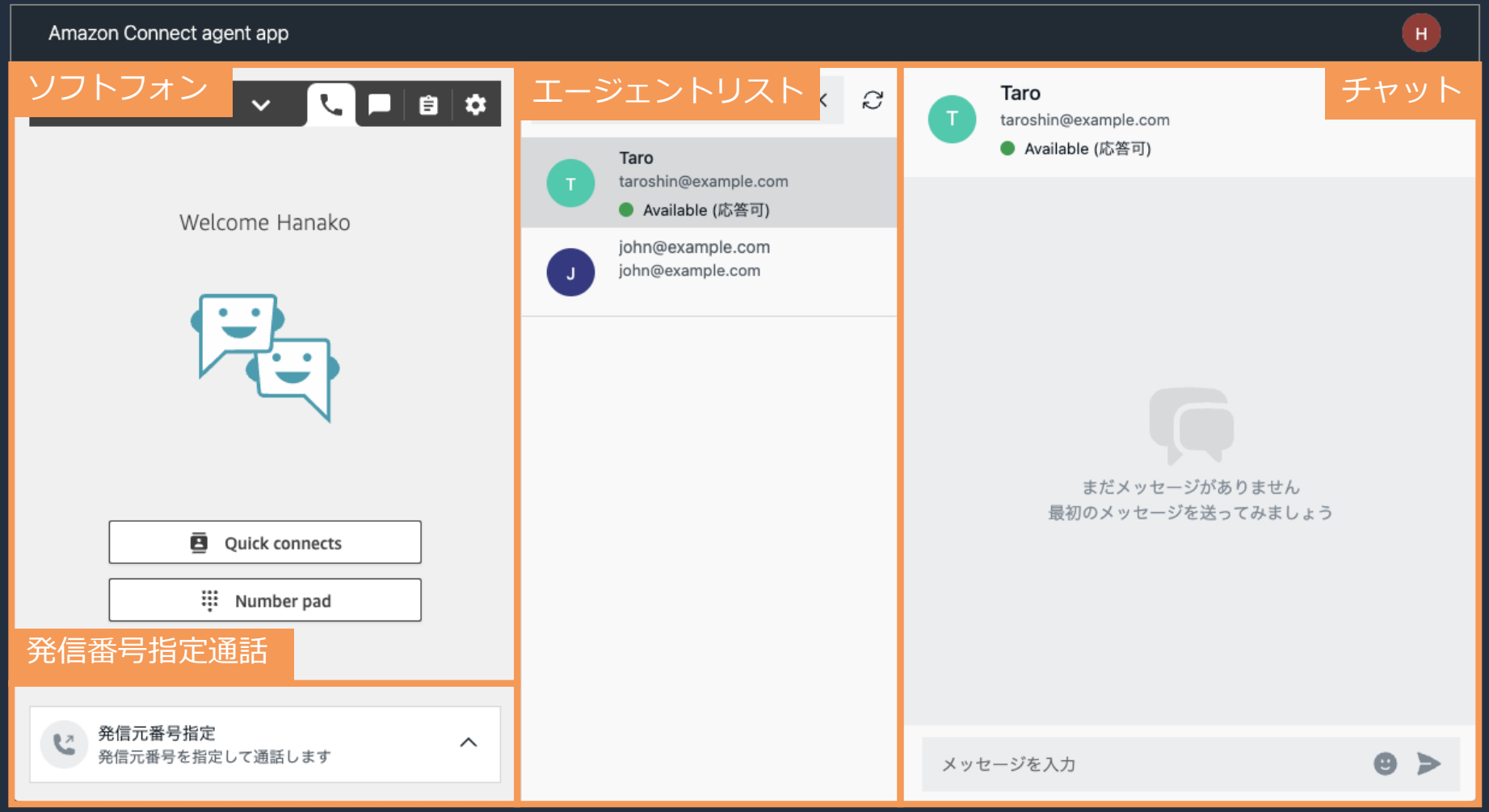
- 認証、セキュリティ
  - **Amazon Cognito** + **Amazon Connect** (SAML) によるシングルサインオンを実現
  - **AWS WAF** を利用し IPアドレス制限

カスタム CCP アーキテクチャ全体像

- バックエンド
  - サーバーレスアーキテクチャで 各 API を実装
    - **Amazon API Gateway (REST API, WebSocket API)**
    - **AWS Lambda**
    - **Amazon DynamoDB**
  - Amazon Connect のデータを **Amazon EventBridge** で DynamoDB に定期的にキャッシュ



画面構成



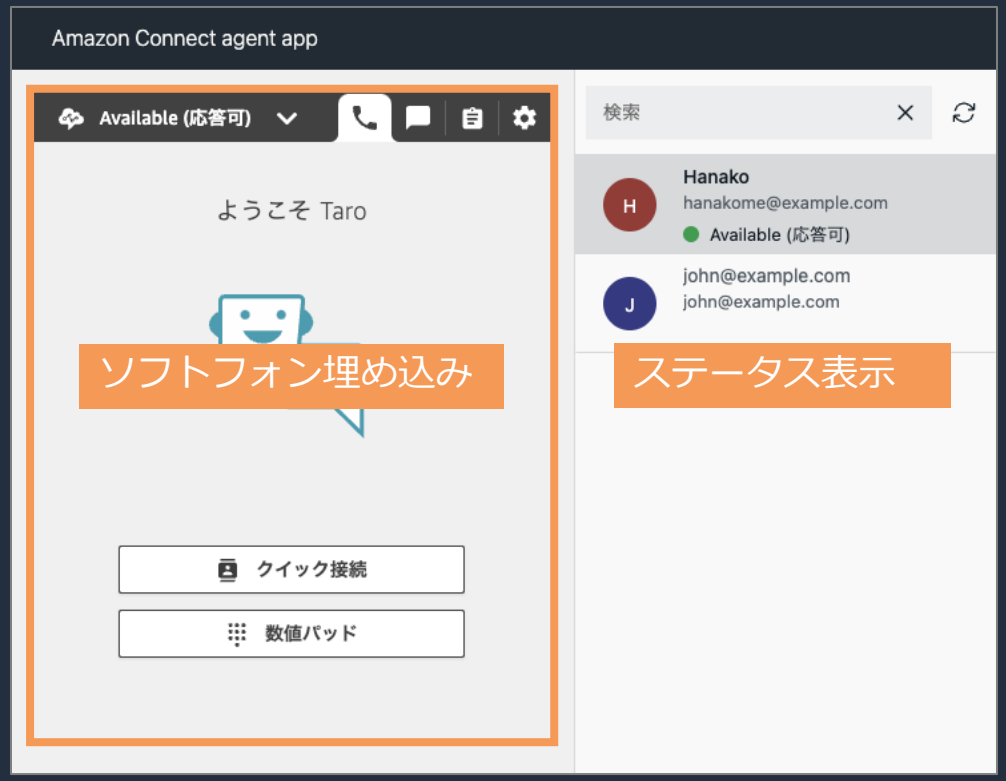
フロントエンドの構成

- 利用しているライブラリ
  - React 18
  - Amplify Library
    - React アプリケーションでユーザー認証を実装するために利用
    - @aws-amplify/ui-react の Authenticator コンポーネントを利用し、認証画面の実装が（ほぼ）不要
  - Amazon Connect Streams API
    - ソフトフォン (Contact Control Panel, CCP) をアプリケーションに埋め込んで、カスタム CCP を構築可能

参考：
https://docs.aws.amazon.com/ja_jp/prescriptive-guidance/latest/patterns/authenticate-react-application-users-by-using-amazon-cognito-and-aws-amplify.html

Amazon Connect Streams API

- JavaScript から利用可能なライブラリ
- initCCP でソフトフォン (CCP) をアプリケーション画面に埋め込む
- init 後はエージェントやコンタクト API が利用可能になり、API を利用して以下のような操作が可能となる
  - エージェントステータス取得
  - アウトバウンドコール開始



https://github.com/amazon-connect/amazon-connect-streams/blob/master/Documentation.md

Amazon Connect Streams API 読み込み

```
import 'amazon-connect-streams';
import { useEffect } from 'react’;
import OutboundCall from './OutboundCall';

function Ccp() {
  useEffect(() => {
    connect.core.initCCP(document.querySelector('#ccpContainer')!, {
      ccpUrl: 'https://customapp.my.connect.aws/ccp-v2’,
      loginPopup: false,
      region: 'ap-northeast-1’,
      softphone: {
        allowFramedSoftphone: true,
        disableRingtone: false,
      },
    });
    return () => {
      document.querySelector('#ccpContainer')!.innerHTML = ’’;
    };
  }, []);
  return (
    <div className="ccp">
      <div id="ccpContainer" className="softphone" />
      <div className="outbound-container"><OutboundCall /></div>
    </div>
  );
}
export default Ccp;
```

本カスタム CCP ではログインポップアップは不要なので false

以下のコールバックを登録しておくことで initialize が検知できる

以降は agent オブジェクトを介して CCP の操作やイベント購読などが利用可能

```
connect.agent(function(agent) { /* ... */ });
```

発信番号の指定機能

- バックエンド
  - API Gateway (REST API) + Lambda (Typescript) で実装
  - phonenumbers API が呼ばれるとキュー ARN に紐づいた電話番号をリストで返す
- フロントエンド
  - GET phonenumbers API を呼び電話番号リストを取得
  - 電話番号を選択し、通話開始ボタンを押す
  - Amazon Connect Streams API の agent.connect() を呼びアウトバウンドコールを実行

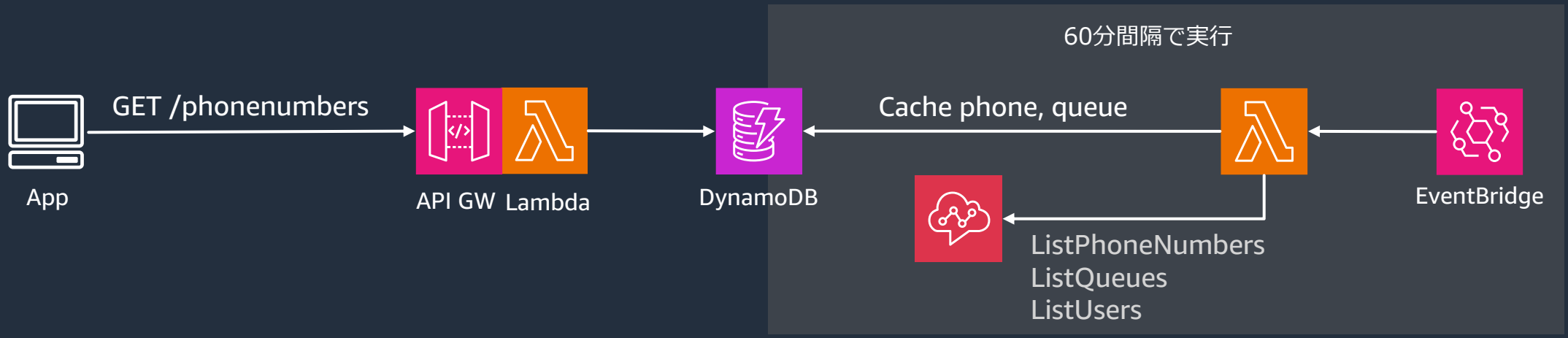
Amazon Connect Streams API の利用

- JavaScript から CCP を操作する：アウトバウンドコールの開始

```
const endpoint = connect.Endpoint.byPhoneNumber(相手先番号);
connect.agent(agent => {
  agent.connect(endpoint, { queueARN: 発信元番号のキューARN });
  success: function () {
    close();
  },
  failure: function () {
    setMessage({ error: t('CCP.MESSAGE_CALL_FAILED') });
  },
});
```

発信番号とユーザーリストのキャッシュ

- Amazon Connect の API 制限のデフォルト値
  - 1秒あたり2リクエスト (RateLimit) / 1秒あたり5リクエスト (BurstLimit)
  - 参考：API スロットリングのクォータ
    - https://docs.aws.amazon.com/ja_jp/connect/latest/adminguide/amazon-connect-service-limits.html#api-throttling-quotas
- 対応
  - Connect API のアクセス頻度を軽減する
- 本カスタム CCP では常にキャッシュからデータを取得することで頻度を調整
  - 電話番号、キュー、ユーザーリスト



エージェントのステータス確認機能

- エージェントのステータス一覧
  - Available、Offline、カスタムステータス
  - 発信中、通話中、アフターコールワーク
- フロントエンドでのステータス取得
  - Amazon Connect Streams API の agent のリスナーで取得
    - onStateChange, onRoutable, onOffline など
- バックエンドでのステータス処理
  - WebSocket API でステータス送受信を行う
  - Lambda (Typescript) で実装、new, update, list などのコマンドごとに処理を分岐

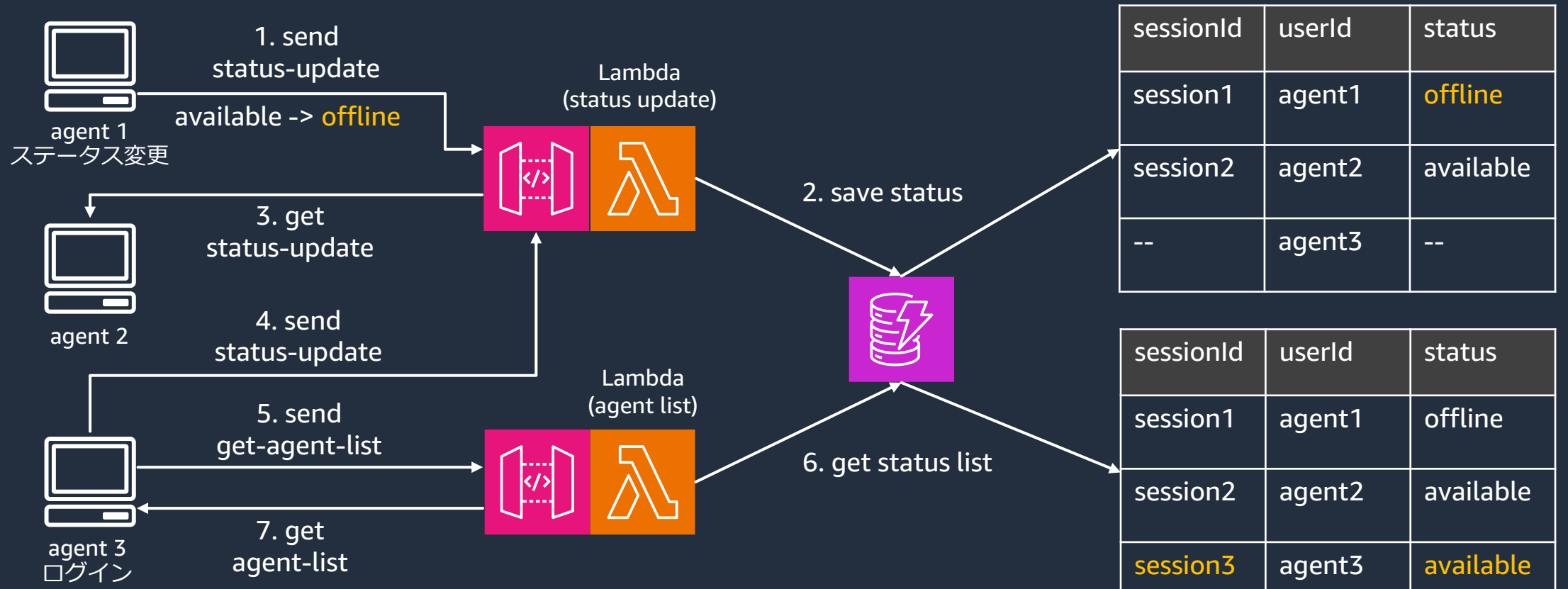
Amazon Connect Streams API の利用

- イベントの Subscribe：エージェントのステータスを取得

```
connect.agent((agent) => {
  // Monitor Available status
  agent.onRoutable(() => {
    dispatch({ type: 'send-update', payload: { agentStatus: agent.getState() } });
  });
  // Monitor Offline status
  agent.onOffline(() => {
    dispatch({ type: 'send-update', payload: { agentStatus: agent.getState() } });
  });
  // Monitor all status changes
  agent.onStateChange((agent) => {
    if (agent.newState == 'Busy' || agent.newState == 'CallingCustomer' || agent.newState == 'AfterCallWork') {
      dispatch({ type: 'send-update', payload: { systemStatus: agent.newState } });
    } else {
      const currentState = agent.agent.getState();
      if (isNotRoutableChange(currentState)) {
        dispatch({ type: 'send-update', payload: { agentStatus: currentState, systemStatus: '' } });
      } else {
        dispatch({ type: 'send-update', payload: { systemStatus: '' } });
      }
      ...
```

エージェントのステータス確認機能

- バックエンドのデータ送受信フロー (WebSocket API + Lambda)

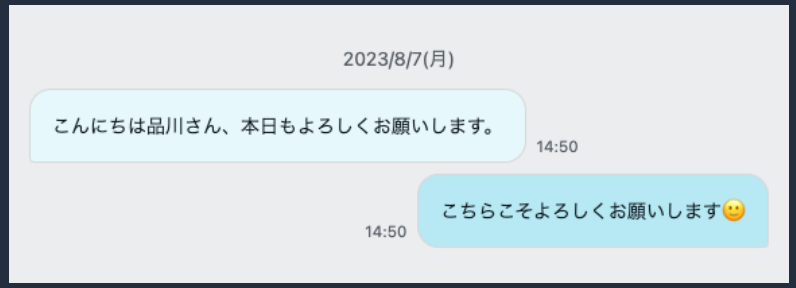


sessionId	userId	status
session1	agent1	offline
session2	agent2	available
--	agent3	--

sessionId	userId	status
session1	agent1	offline
session2	agent2	available
session3	agent3	available

エージェント間チャット機能

- WebSocket API でメッセージ送受信を行う



- バックエンドは Lambda (Typescript) で実装、add, list などのコマンドごとに処理を分岐

フロントエンド - send data to websocket API

```
socket.send(JSON.stringify({
  action: 'chat',
  command: 'add',
  params: {
    partnerName:
'hanakome@example.com',
    message: 'こちらこそよろしくお願
いします'
  }
}));
```

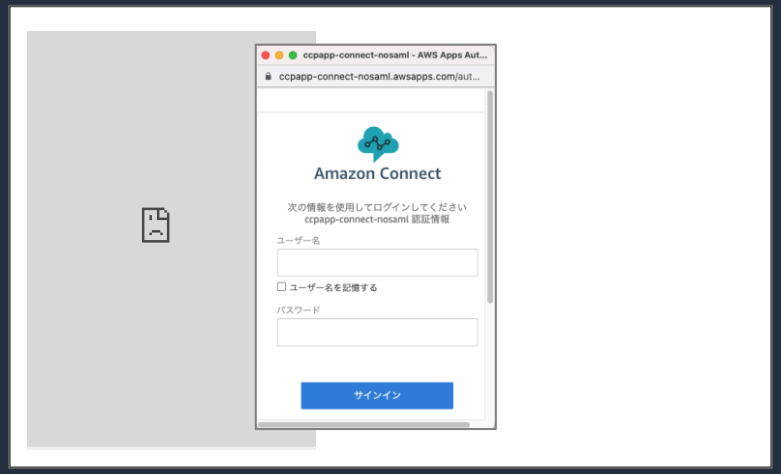
Lambda - ChatHandler

```
export const handler = async (event: any, context: Context) => {
  const body = JSON.parse(event.body) as RequestMessage;
  const command = body.command;
  switch (command) {
    case 'list':
      // chat table からチャット履歴を取得しメッセージを送信
      break;
    case 'add':
      // chat table にデータを push し partner にメッセージを送信
      break;
    default:
  }
  return { statusCode: 200 };
```

Amazon Connect CCP の認証

- CCP 埋め込みを行うと SAML でない場合はポップアップによる認証が行われる
- 認証はあくまで iframe で埋め込まれた CCP に対して行われるだけ、アプリケーション自体の認証ではない
- どのユーザーでも他人のチャットデータにアクセスできる可能性がある

認証前

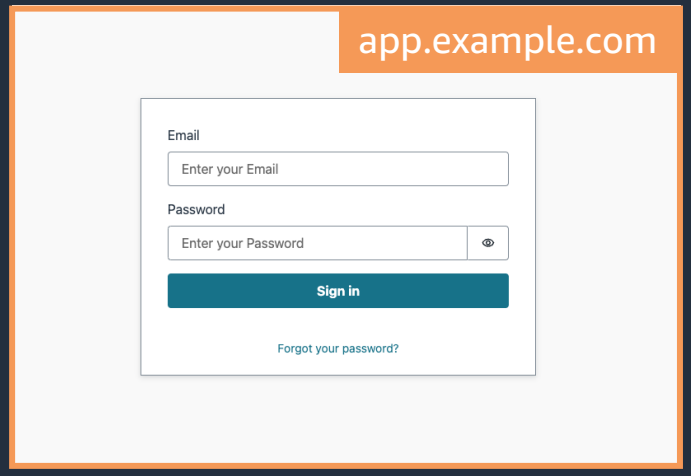


認証後

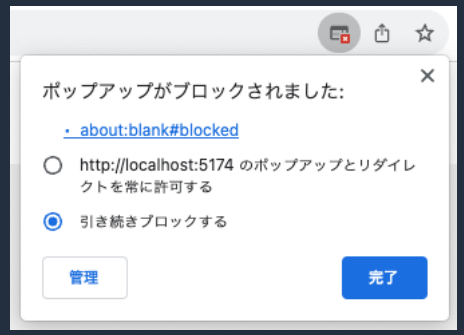
Amazon Connect CCP の認証（/w Cognito）

- Cognito による認証を行う
- Cognito を IdP とした SAML 認証を行うことで Cognito と Amazon Connect 間でシングルサインオンを実現
- Amazon Connect インスタンスを SAML enabled にする必要あり
- 副次的な効果 : **CCP のログインポップアップ不要 → ユーザー体験の改善**

認証前

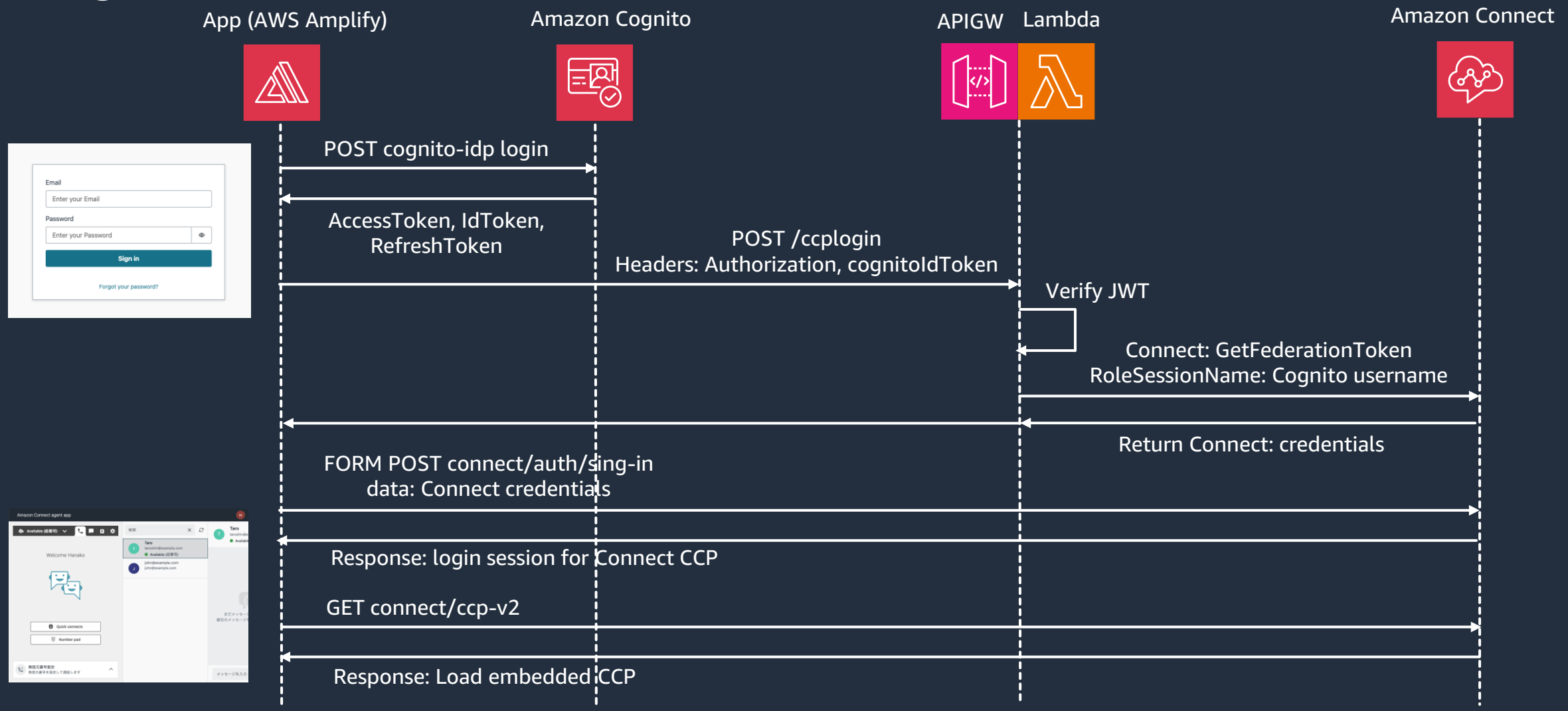


認証後



ポップアップブロック

Cognito + Connect (SAML) による認証の遷移図



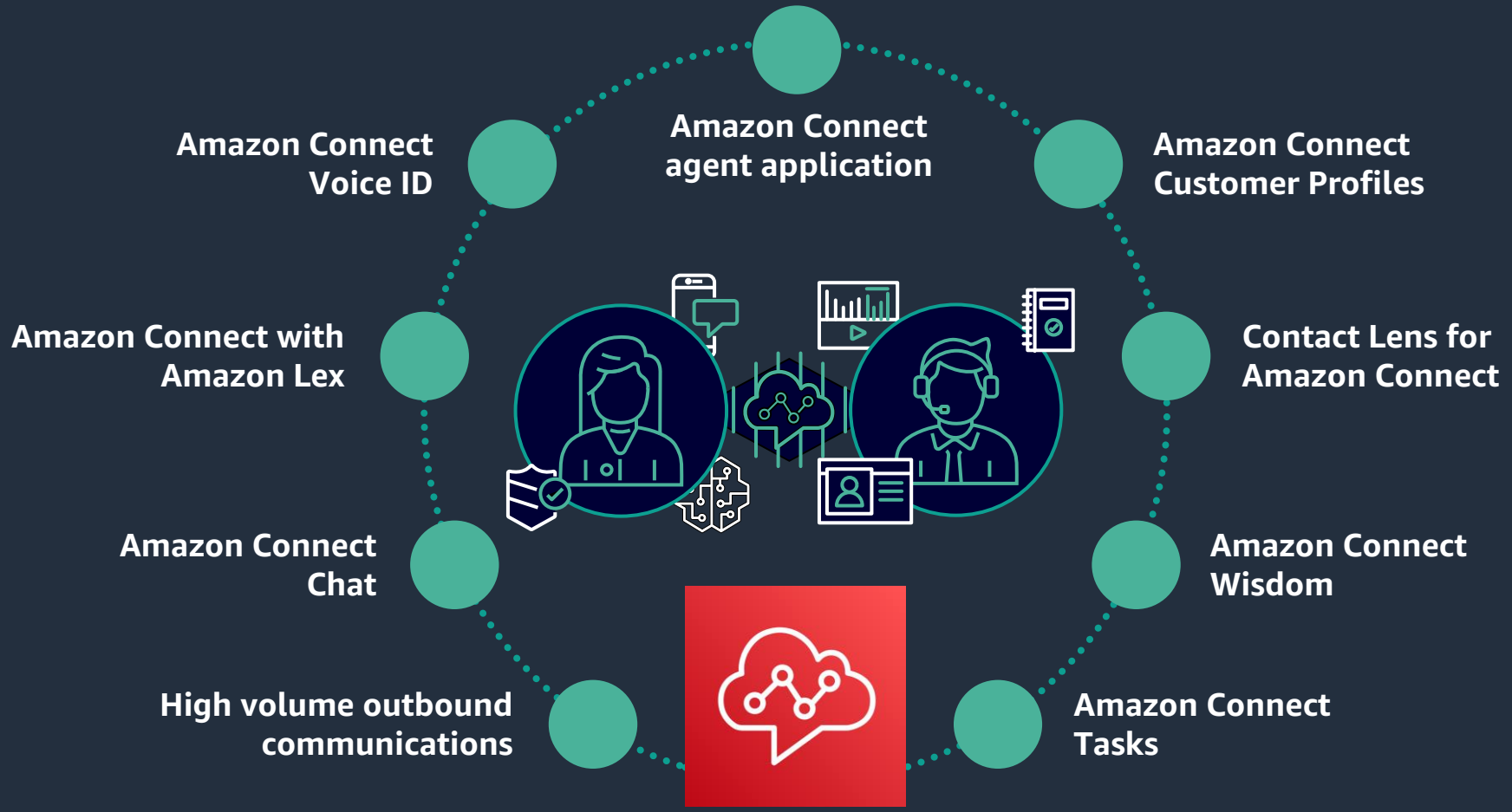
参考: https://github.com/amazon-connect/video-call-escalation

エージェント間チャット機能のセキュリティ

- API Gateway の 認証・認可
  - REST API では以下のオーソライザーが利用可能
    - **AWS IAM (via Authorization header)**
    - Cognito ユーザープール認証
    - Lambda 関数によるカスタム認証
  - WebSocket API では以下のオーソライザーが利用可能
    - AWS IAM (via query params in Signed wss URL)
    - **Lambda 関数によるカスタム認証 ... 後続で呼ばれる Lambda 関数に認証情報を渡すことができる**
- チャット履歴の取得
  - Lambda 関数内で Cognito id token (JWT) を validate し、そこで得た email でチャットの履歴をクエリして返す
  - 他のユーザーのチャット履歴は取得できない

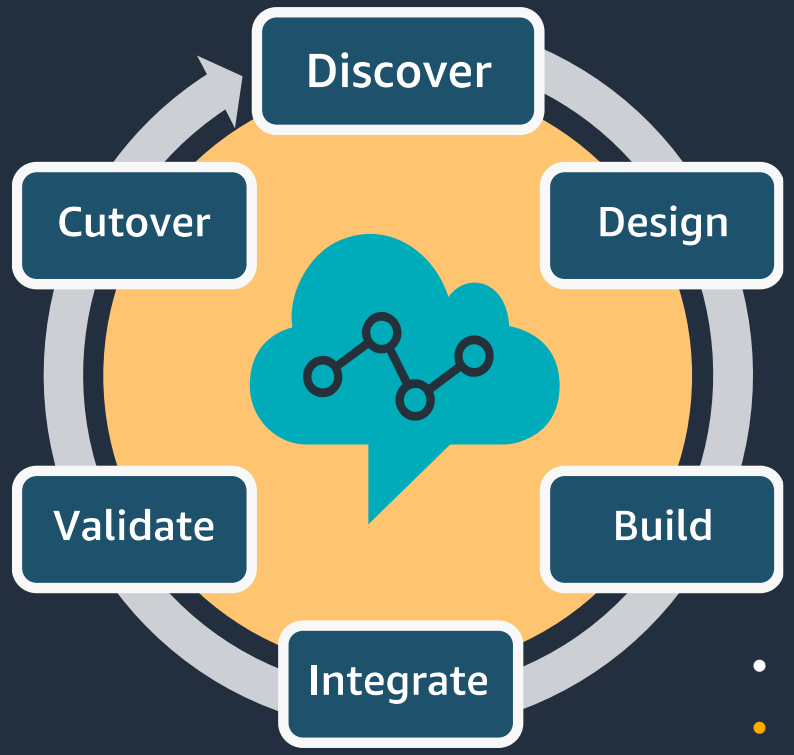
まとめ

顧客と従業員の利便性を向上



コンタクトフローとルールによるローコード/ノーコード構築
APIによるプログラマブルコンタクトセンター

クラウドコンタクトセンターによる継続的な改善



“think **BIG**, start **SMALL**, go **FAST**”

- 改善計画の策定
- Think big & start small

- 必要最低限の機能でサービスを開始

- コンタクトセンターの高度化を加速
- 最新機能による継続的な改善

まとめ

- Amazon Connect は API によるオープンな特徴を活かし、AWS サービスやサードパーティーサービスによる拡張が可能

- ソフトフォンのユーザーインタフェースを柔軟に拡張することで、オペレーターの生産性向上を支援する機能を実装し、コンタクトセンターの課題を技術的な面から解決するサポートを実現

技術情報の参考リンク

- Amazon Cognito と AWS Amplify を使用して React アプリケーションユーザーを認証する
- Video Call Escalation - powered by Amazon Connect and Amazon Chime SDK

次のステップ

- Amazon Connect Streams (GitHub)
- Amazon Connect API References
- Amazon Connect Streams API 解説 (AWS Black Belt)
- Amazon Connect Softphone Controls (Workshop)
- Amazon Connect Bootcamp
  - Integrate CCP into a custom web page (Workshop)

AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWS の技術担当者が、 AWS の各サービスやソリューションについてテーマごとに動画を公開します
- 以下の URL より、過去のセミナー含めた資料などをダウンロードすることができます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/
  - https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY

ご感想は X (Twitter) へ！ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では資料作成時点のサービス内容および価格についてご説明しています。AWS のサービスは常にアップデートを続けているため、最新の情報は AWS 公式ウェブサイト (https://aws.amazon.com/) にてご確認ください
- 資料作成には十分注意しておりますが、資料内の価格と AWS 公式ウェブサイト記載の価格に相違があった場合、AWS 公式ウェブサイトの価格を優先とさせていただきます
- 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます
- 技術的な内容に関しましては、有料の AWS サポート窓口へお問い合わせください
- 料金面でのお問い合わせに関しましては、カスタマーサポート窓口へお問い合わせください (マネジメントコンソールへのログインが必要です)

Thank you!