コマース用スマフォアプリにおける AWS構成＆Cognito活用事例

クルーズ株式会社
SHOPLIST.com事業本部
稲垣 剛之 / 加川 申祐
技術統括本部 TeamZeus
田沢 知志

© CROOZ,Inc.

CROOZって何やってる会社？

CROOZは、ソーシャルゲームやネット通販を中心に、世界中にインターネットサービスを提供するエンターテインメント企業です

© CROOZ,Inc.

アジェンダ

- SHOPLIST.comのAWS構成/負荷 概要
- Congito活用事例
- 今後のAWS活用予定・方向性
- 今後AWSに期待したいこと

© CROOZ,Inc.

SHOPLIST.comの
AWS構成/負荷 概要

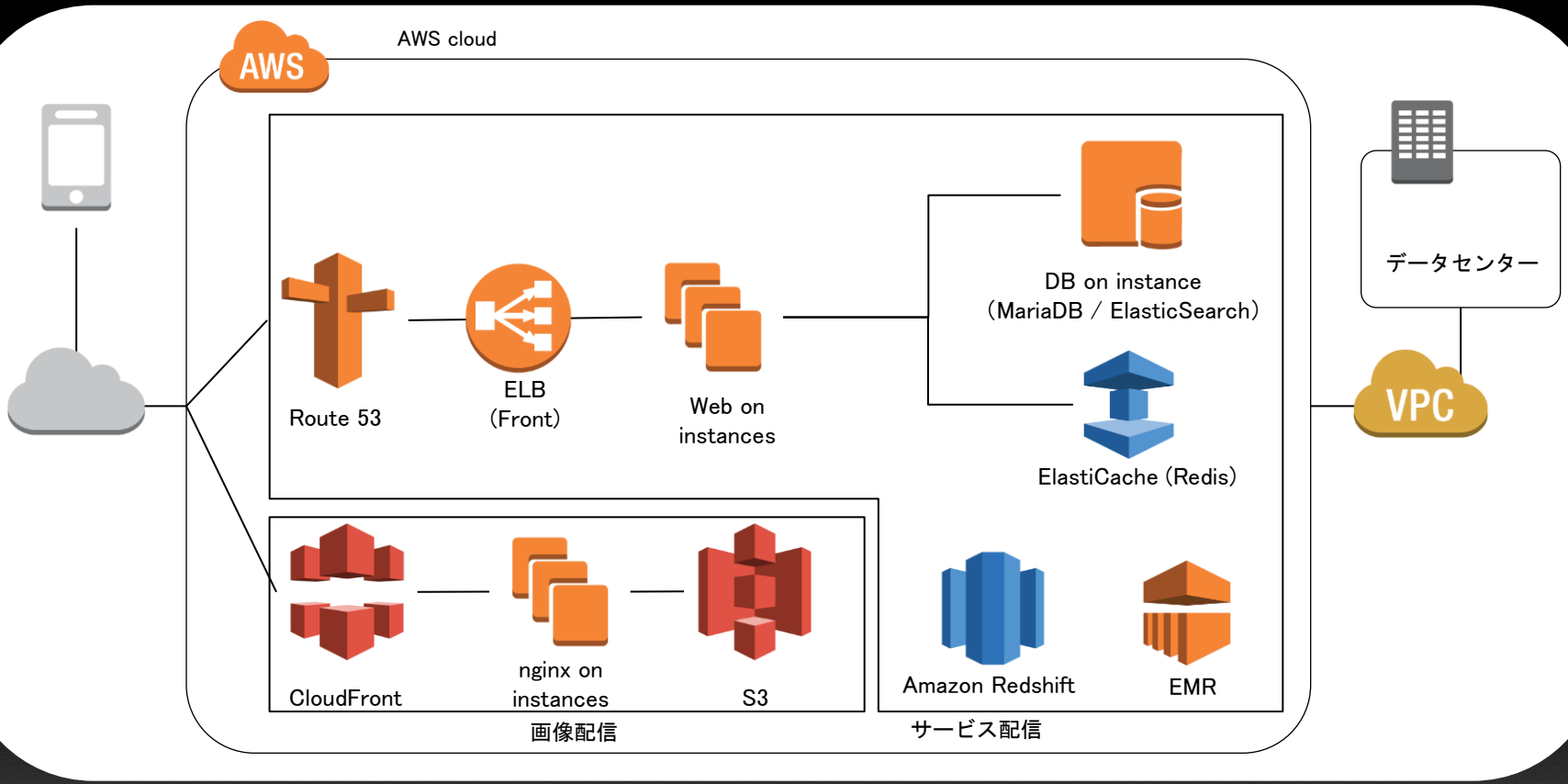
© CROOZ,Inc.

■SHOPLIST.com ご紹介

- 年間取扱高約100億円(2014年度)
- 単月取扱高 10億円突破(2015/4)
- 取扱高の90%以上はスマートフォン経由

© CROOZ,Inc.

■インフラ構成概要



© CROOZ,Inc.

■OS/Middleware概要

- OS:CentOS6.4
- Web:apache2.2系/PHP5.4系
  - 社内独自フレームワークVENUS使用
- Cache:Amazon ElastiCache(Redis)
- DB(ﾄﾗﾝｻﾞｸｼｮﾝ系):MariaDB 10.0.13系
- DB(検索系):Elasticsearch 1.5.2系
- Amazon Elastic MapReduce(Spark 1.3.1)

© CROOZ,Inc.

■インスタンスタイプ(2015/5時点)

- インスタンスタイプは3-6か月単位で再検討
  - 旧/新インスタンスを比較すると、コストパフォーマンスは3割以上良い(印象)
- 現在メインで使用してるタイプは・・
  - Web系：c3.2xlarge、c3.xlarge
  - DB系：r3.4xlarge、r3.2xlarge
    - EBSはgp2をメインで使用
  - Cache：cache.r3.large
  - その他(バッチ系)：m3.xlarge

© CROOZ,Inc.

■スケーラビリティに関して

- 年３回実施の「メガセール」事例
- 通常時の約5倍のアクセス数(ピークは10倍)
- ピーク時の
  - 過去最大リクエスト数は1,000r/s
- ただしLatencyはサイト全体900msを超えることはない

© CROOZ,Inc.

■コストの考慮点

- 1インスタンスあたりのrequests/sを想定
  - 弊社参考例
  - Web(c3.2xlarge)　100r/s
  - DB(r3.2xlarge gp2)　200r/s
  - Cache(r3.large)　500r/s
- 想定requests/sから必要インスタンスを算出
  売上の?%以内をクラウドコスト目標に

© CROOZ,Inc.

■画像配信に関して

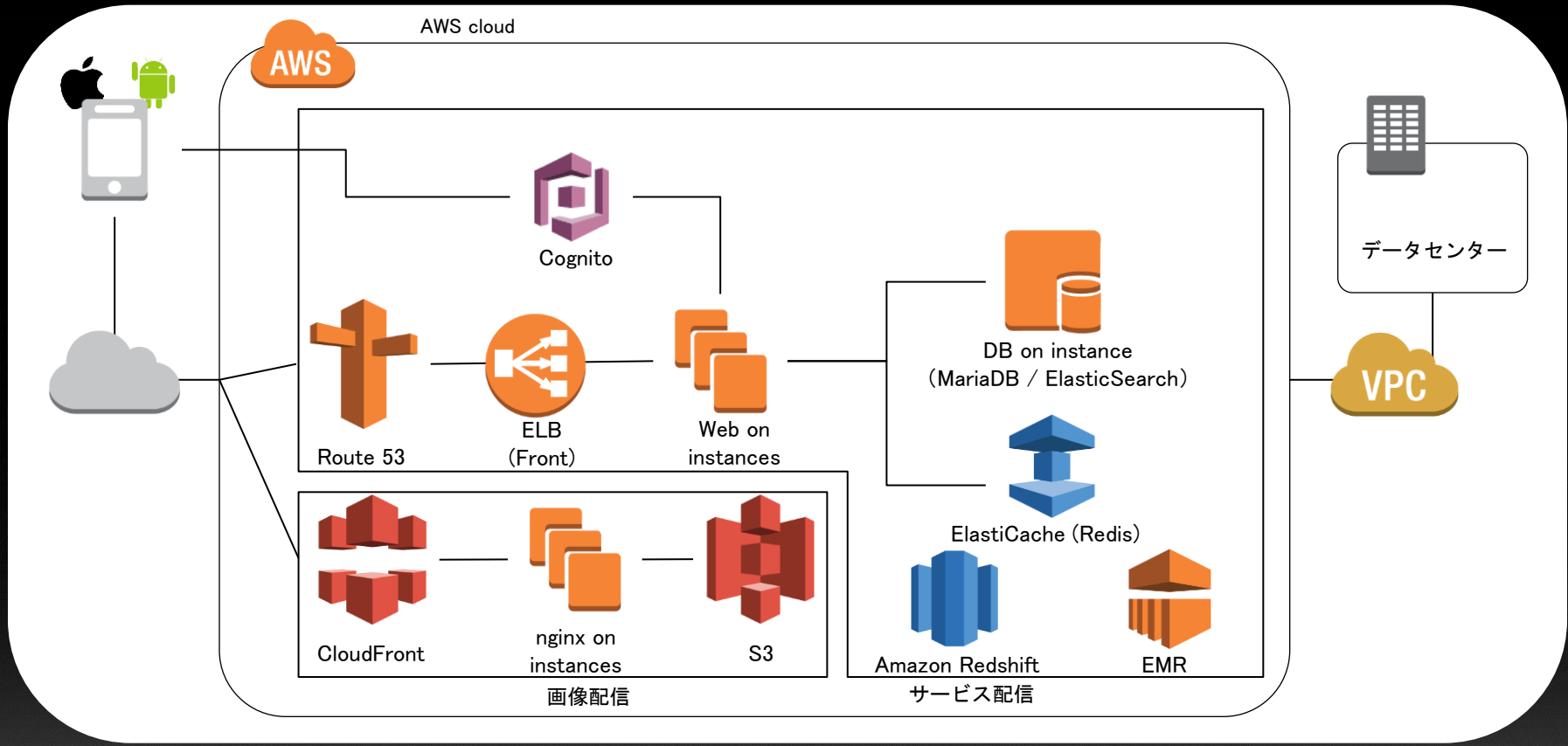
- 画像配信はCloudFrontを使用
  - オリジンはS3に配置
  - サムネ作成はnginx/SMALL LIGHT使用
  - Reports & Analytics 機能もあり
  - 数TB/日の配信で利用
  - リザーブドプラン契約により3-4割安に

© CROOZ,Inc.

Congito活用事例

© CROOZ,Inc.

■インフラ構成概要



© CROOZ,Inc.

■Cognitoを選定した理由

- 極力EC2を経由せず、アプリ・新サービスからのアクセスを捌きたい
- SHOPLIST.comの認証基盤を今後新サービスにも展開したい
- アプリのパーソナライズデータやユーザの行動情報を活用したい

© CROOZ,Inc.

■導入時に苦労した点 / 導入して良かった点

- 既存の認証基盤を活用する場合、id管理にのみ使用するのであれば一手間増えるだけになってしまう
- デバイス間のデータ同期の仕組みを新規で作らなくて良い
- Latencyは気になるレベル

© CROOZ,Inc.

■具体的なプログラムの概要

- SHOPLIST.comアプリではdeveloper authenticated identitiesを使用
- ゲストモード(unauthenticated identities)にも対応

実装自体は容易です

© CROOZ,Inc.

iOS実装

```
// ゲストログイン
AWSCognitoCredentialsProvider *credentialsProvider = [AWSCognitoCredentialsProvider
                                                      credentialsWithRegionType:AWSRegionUSEast1
                                                      accountId:_accountId
                                                      identityPoolId:_identityPoolId
                                                      unauthRoleArn:nil // Arnはnil指定
                                                      authRoleArn:nil]; // Arnはnil指定

AWSServiceConfiguration *configuration = [AWSServiceConfiguration
                                          configurationWithRegion:AWSRegionUSEast1
                                          credentialsProvider:credentialsProvider];

[AWSServiceManager defaultServiceManager].defaultServiceConfiguration = configuration;


// Developer authenticated identities
CustomIdentityProvider *customIdentityProvider = [[CustomIdentityProvider alloc] initWithIdProvider:idProvider
                                                                                  accountId:_accountId
                                                                                  identityPoolId:_identityPoolId
                                                                                  token:token];
customIdentityProvider.logins = @{name:token};

AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider alloc] initWithRegionType:AWSRegionUSEast1
                                                                                           identityProvider:customIdentityProvider
                                                                                           unauthRoleArn:nil    // Arnはnil指定
                                                                                           authRoleArn:nil]; // Arnはnil指定
AWSServiceConfiguration *configuration = [AWSServiceConfiguration configurationWithRegion:AWSRegionUSEast1
                                                                                  credentialsProvider:credentialsProvider];

[AWSServiceManager defaultServiceManager].defaultServiceConfiguration = configuration;
```

© CROOZ,Inc.

iOS実装(CustomIdentityProvider)

```
- (BFTask *)getIdentityId {
    // ゲストログインに対応していると切り替え時にself.identityIdがあるためサーバサイドにリクエストが通らないため、サーバサイドにリクエストを飛ばしたことを判定する必要があります
    if (self.identityId && model.isMemberLoginRequest) {
        return [BFTask taskWithResult:@{@"result" : @"1"}];
    } else {
        return [[BFTask taskWithResult:nil] continueWithBlock:^id(BFTask *task) {
            if ( self.identityId && model.isMemberLoginRequest) {
                return [BFTask taskWithResult:@{@"result" : @"1"}];
            } else {
                return [self refresh];
            }
        }];
    }
}

- (BFTask *)refresh {
    BFTaskCompletionSource *source = [BFTaskCompletionSource taskCompletionSource];
    ApiRequest *authApi = [_idProvider.apiManager generateAuthApiByToken:_token]; // SHOPLISTのAPIリクエストの実装
    [authApi requestAsyncCompletionHandler:^(ApiRequest *request) {
        NSDictionary *response = request.response;
        if (![request hasSucceeded]) {
            [source setResult:response];
        } else if ([[response valueForKey:@"result"] intValue] == 1) {
            model.isMemberLoginRequest = YES;
            self.identityId = [[response valueForKey:@"data"] valueForKey:@"identityId"];
            self.token = [[response valueForKey:@"data"] valueForKey:@"token"];
            [source setResult:response];
        } else {
            [source setResult:response];
        }
    }];
    return [source task];
}
```

© CROOZ,Inc.

■サーバサイド実装(PHP)

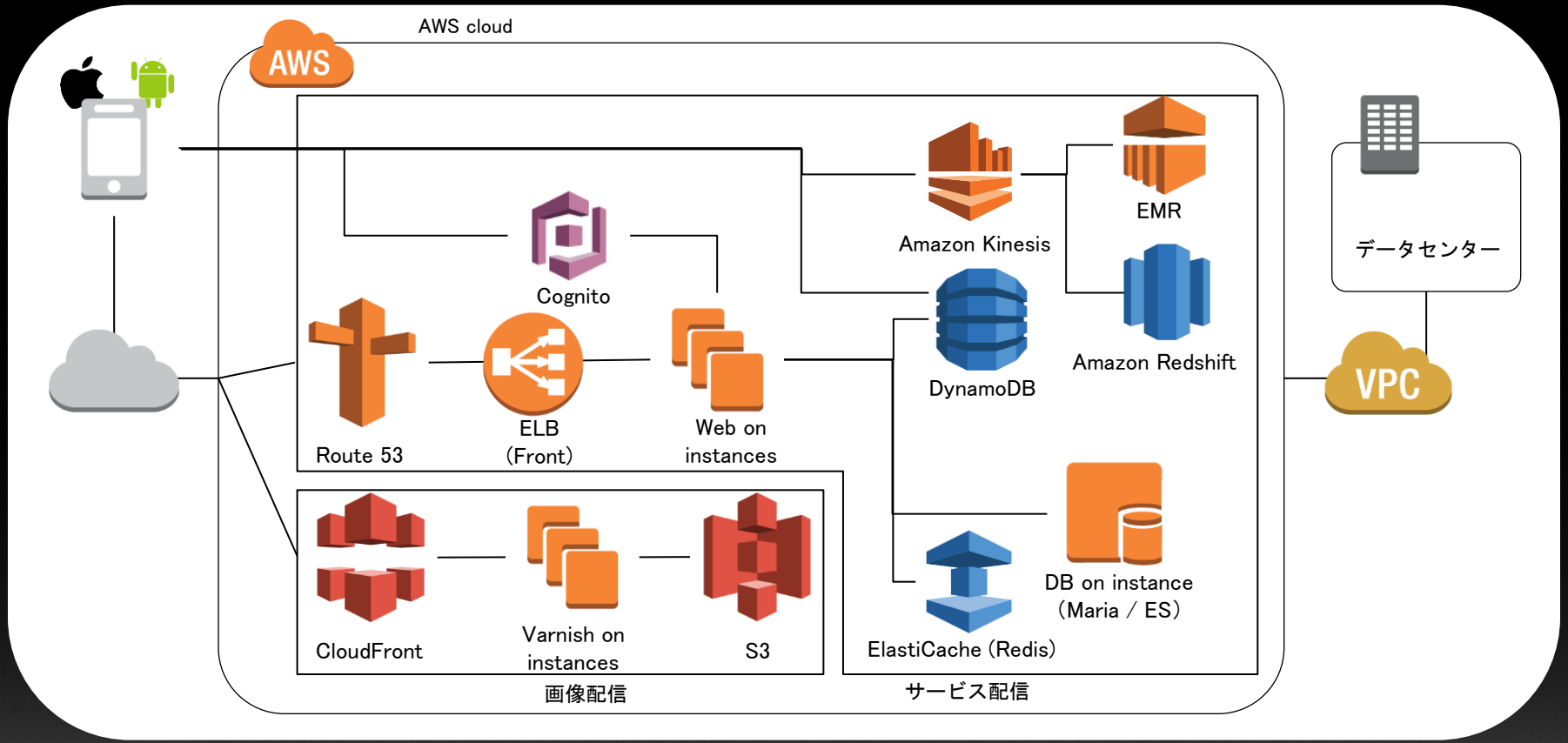
```
class CognitoAuthenticator {
    public function register($token, $oldToken = false) {
        $result = array();
        $config = array('key' => 'XXXXXXX', 'secret' => 'XXXXXXX', 'region'  => 'us-east-1');
        try {
            $idClient = CognitoIdentityClient::factory($config);
            $response = $idClient->GetOpenIdTokenForDeveloperIdentity(array(
                                                                    'IdentityPoolId' => 'xxxxxxxxxxxxx',
                                                                    'Logins' => array(
                                                                                'name' => $token,
                                                                     )
                                                                ));
            $identityId = $response->get('IdentityId');
            $openIdToken = $response->get('Token');
            // 古いtokenのマージ(無期限tokenというわけにもいかないので更新された場合の対応が必要)
            if ($oldToken) {
                $mergeResponse = $idClient->MergeDeveloperIdentities(array(
                                                                    'DestinationUserIdentifier' => $token,
                                                                     'DeveloperProviderName' => 'xxxxxxxxxxxxxxx',
                                                                     'IdentityPoolId' => 'xxxxxxxxxxxxxxxxxx',
                                                                     'SourceUserIdentifier' => $oldToken)
                                                                );
                $identityId = $mergeResponse->get('IdentityId');
            }
            $result['identityId'] = $identityId;
            $result['token'] = $openIdToken;
        } catch (Exception $e) {
            // エラー処理
        }
        return $result;
    }
}
```

© CROOZ,Inc.

今後のAWS活用予定・方向性

© CROOZ,Inc.

■ 今後のAWS活用方針



© CROOZ,Inc.

■ECサイトを更に便利にするために

- リアルタイムに精度の高いアイテムをお勧めデバイス間でのシームレス、パーソナライズされたショッピング環境の提供
- ユーザメリットにフォーカスした検索並び順
- 未来販売予測による売れ筋アイテムの効率的な在庫確保

© CROOZ,Inc.

今後AWSに期待したいこと

© CROOZ,Inc.

■今後リリースされる機能が盛りだくさん！

- Machine Learning
  - 未来販売予測、不適切レビュー・商品検知
- Aurora
  - 無停止&柔軟なRDB拡張
- Lambda
  - 最低限インスタンスでのサービス実現
- etc…

© CROOZ,Inc.

■今後AWSに期待したいことは・・・

- 無停止でのインスタンスタイプ変更
- 不正アクセスやアプリケーションレベルの攻撃をモニター・検知するWAF(Web Application Firewall)の仕組みを公開
- EC2がより少なくて済むようなコストメリットが高く、運用しやすいアーキテクチャー、ミドルウェアの提供

© CROOZ,Inc.

ご清聴ありがとうございました

© CROOZ,Inc.