

AWSドンジャラ Ver1.1

JAWS-UG

# ルール(作り方)

● CDP



CDPを揃える(パターンは後述)

● 数牌



同じ色のパイを3つ集める。

● リージョン or 三元牌



リージョン or 三元牌を2枚揃える

同じリージョン2枚同じだと+1000点(マルチデータセンター)

# ルール(上がり方)

- 手配でCDPを1つ以上、リージョン(三元牌)を1つ、のこりを数牌の合計11枚で作れば上がり

あがり例)



- CDPが2つ (DirectHostingパターン・BIパターン) マルチデータセンター
- 点数  $3000(\text{CDP} \times 2) + 1000(\text{マルチ}) = 4000$  点

# ルール(その他)

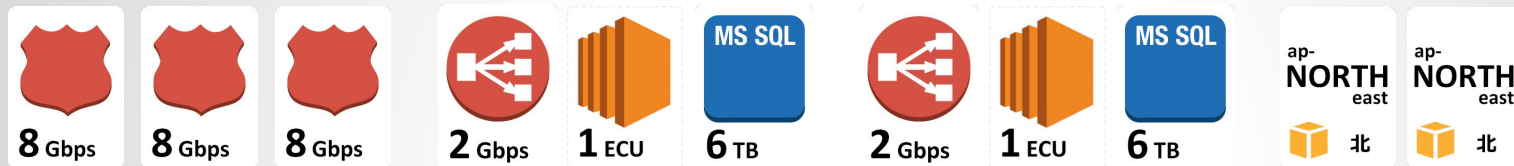
- 持ち点は10000点
- 4ゲームするか、誰かの持ち点がなくなった時点で終了
- 親は順に回っていく
- 手配は11枚、ドラなし
- ポン・チーはなし。
- 同じハイが4つそろったらカンと宣言する(自分で4枚揃えたら+2000点、他人からもらったら+1000点)
- JAWS-UG牌はオールマイティ牌として扱う
- 牌は全員オープンで行う

# ルール(点数計算)

- CDPが一つの場合は1000点、CDPが2つの場合は3000点
- カンをした場合は+1000点
- 同一リージョンだと+1000点
- ツモあがりをした場合は、トータル点数に+2000点し、トータルを3で割って10の位を切り上げた点数を3人が払う(親子関係なし)
- 親の場合は、トータル点数を1.5倍して計算する
- 上がり時にスタッフがCDPの説明を行う事でCDPへの理解を深める

# 特殊あがり(ドンジャラ限定) 10000点

## ● リファレンスWebアーキテクチャ



+ ECU  
2x

## ● スシロー



# 特殊あがり(ドンジャラ限定) 10000点

## ● リファレンスWebアーキテクチャ



## ● スシロー(役満)



# 特殊役



Kinesushi

Kinesis暗カンで+3000、明カンで+2000



Docomo

Redshift暗カンで+3000、明カンで+2000



























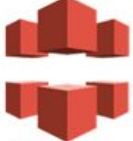











# AWS麻雀・ドンジャラ CDP役一覧 Ver1.1

JAWS-UG

# 牌の種類

 1 萬	 2 萬	 3 萬	 4 萬	 5 萬	 6 萬	 7 萬	 8 萬	 9 萬
 1 筒	 2 筒	 3 筒	 4 筒	 5 筒	 6 筒	 7 筒	 8 筒	 9 筒
 1 索	 2 索	 3 索	 4 索	 5 索	 6 索	 7 索	 8 索	 9 索
US- EAST  東	ap- SOUTH east  南	US- WEST  西	ap- NORTH east  北					

# 牌の説明 萬子(マンズ)



EC2



Elastic  
Beanstalk



Auto Scaling



Instances



Elastic Load  
Balancing



Amazon Lambda



EC2  
ContainerService



AMI



Amazon Kinesis

※青字は変更

# 牌の説明 筒子(ピンズ)



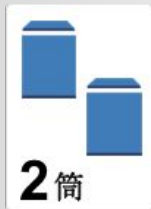
Dynamo DB



bucket



AWS IoT



Amazon EBS



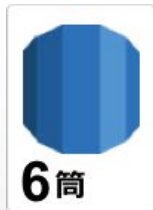
ElastiCache



Mobile Hub



snapshot



RDS



Amazon Redshift

※青字は変更

# 牌の説明 索子(ソーズ)



Direct Connect



EMR



QuickSight



CloudWatch



Machine Learning



Route53



elastic network  
instance



S3



Cloud Front

※青字は変更

# 牌の説明 三元牌(サンゲンハイ)



WAF



Cognito



Amazon  
CloudSearch



CloudTrail



Device Farm



Amazon SES



Inspector



MobileAnalytics



Amazon SQS



IAM



SNS



API Gateway

※白に相当

※撥に相当

※中に相当

# 牌の説明 風牌(ファンパイ)



リージョン  
バージニア



JAWS-UG



リージョン  
シンガポール



JAWS-UG  
エンタープライズ



リージョン  
カリフォルニア



JAWS-UG  
中央線



リージョン  
東京



JAWS-UG  
女子会

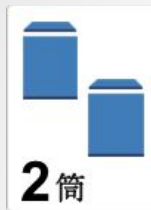
JAWS-UG牌について

麻雀の場合はドラ  
ドンジャラの場合はオールマイティ  
として扱う

# CDP一覽 Ver1.1



# CDP



## Snapshot

ある瞬間のデータをスナップショット(バックアップ)として作成しS3に保存する事でいつでも復元できるようにする。APIを利用して自動バックアップ作成がよくある使い方。



## Stampパターン

AMIを利用する事で、簡単に同じ環境を用意する事が可能。同じ環境を複数台構築する場合にとっても便利になる。

# CDP



4 筒



2 萬

## Web Storage

大容量のファイルや静的コンテンツなどをS3から配信する事でEC2への負荷を減らす。動的コンテンツはEC2より配信する。



2 萬



6 索

## Cache Distribution

Cloudfrontを利用する事で、世界中にあるオリジンサーバーから遅延なくコンテンツを配信できる。  
まとめるとサイトが早くなり、ユーザーへのレスポンスが良くなり、EC2へのアクセス負荷も減ります。

# CDP



## Direct Hosting

R53、Cloudfront、S3を利用する事で、絶対に落ちない静的サイトを構築する事が可能となる。



## Job Observer

SQSを利用して、CloudWatchで指定した閾値を超えた場合、自動でAutoscalingを行う。  
負荷に応じて、EC2の台数を増減(スケールアウト・スケールイン)する。

# CDP



## Back Net

EC2に対して、2つのENI(仮想ネットワークインタフェース)を用意する事で、公開用ネットワークインタフェースと管理用ネットワークインタフェースを利用する。



## State Sharing

ステート情報(セッション情報、ユーザー情報)などをDyanamoDB、Redisに保持することで、サーバー増減時にステート情報の喪失を防ぐ。



# CDP



## Inmemory DB Cache

頻繁に読み込まれるデータをRedisにキャッシュする事で、DBから呼び出すことなくRedisからキャッシュデータを取り出す。



## Scheduled Autoscaling

アクセスが急増するタイミングがわかってる場合、スケジューリングからスケールアウトする事で、サービスを止めずに運用が可能となる。

# CDP



## Storage Index

インターネットストレージにデータを格納する際、同時に検索性能の高いKVS(Dynamo DB)へメタ情報を格納し、その情報をインデックスとして利用する。検索時はKVS(Dynamo DB)を用い、得られた結果を基にインターネットストレージへアクセスする。



## Multi Load Balancer

ELBを複数台用意する事で、同一サイトでELB毎に挙動を変える事ができる。  
PCサイト、スマホサイトをELBを利用してアクセス先を変更できる。

# CDP 追加



## サムライIoT辻

UG京都が誇るAWSサムライ2016の辻さんがこよなく愛する(デモLTでよく事故る)IoTの王道パターン。Kinesisが受けたセンサーデータをLambdaでよろしく加工してDynamo DBに。あとは煮るなり、焼くなり、可視化するなり。



## IoTスターターパックパターン

AWSでIoTを始めるならまず最初に使いたい構成。デバイスからMQTTで受けたセンサーデータをAWS IoTがDynamo DBに直接投入。あとはQuickSiteで簡単可視化。でも残念ながらQuickSiteはまだプレビュー。

# CDP 追加



## マルチリージョンパターン

東と西のリージョンにあるシステムにRoute53でバランシングすると勝手に近い方のリージョンに振り分けてくれるから低レイテンシーをキープできる。



## クラウド移行鉄板パターン

オンプレシステムを構成そのまま少しずつお引越し。エンタープライズ王道CDP。しばらくはハイブリッドでもいいじゃない。先には明るい未来が待っている。



# CDP 追加



## ブルーグリーンデプロイメントパターン

Route53とBeanstalkを利用して安全なリリースを。ダメだったらロールバックすればいいので、どんどん新機能をリリースしちゃいましょう。



## ブルーグリーンデプロイメントECSパターン

上記の進化系。Route53とECSを利用して安全なリリースを。リリースがうまくいったら古いコンテナは捨てて、新しいコンテナに。

# CDP 追加



## BIパターン

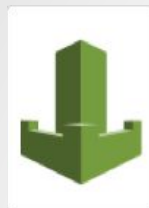
AWSを利用したBI構成の王道パターン。あのデータもこのデータもとにかくKinesisで集めてRedshiftに投入すればQuickSightが真実を見せてくれるはず！



## ディープラーニングパターン

AWSを利用したBI構成の進化系。あのデータもこのデータもとにかくRedshiftに投入すれば、Machine Learningが未来を見せてくれるはず！

# CDP 追加



## セキュアWebサイト三兄弟

この3人が揃えば、DDoSだろうがXSSだろうが、どんな攻撃も怖くない。最強の3兄弟。



## モバイル三兄弟

この3人が揃えば、業務アプリだろうがソーシャルゲームアプリだろうがどんなアプリ開発も怖くない。最高の3兄弟。

# CDP 追加



## 監視パターン

AWS麻雀限定CDP。

CloudWatchで各サービスの挙動を監視しSESでアラート通知。CloudTrailを使えばAWS APIの呼び出し履歴も取得可能。S3に保存されたログを使えば稼働状況の分析もできます。



## サーバーレスAPIパターン

AWS麻雀限定CDP。

REST API公開の新常識となりつつある構成。Lambdaで稼働中のコードをAPIとして簡単に公開、管理することが可能。

# CDP 追加



## スケジュールバックアップパターン

Lambdaを利用して、EC2 or RDS のバックアップを作成し snapshotに保存。今までのバックアップの悩みがこれであっさり解決。



## とあるアプリリリースパターン

Mobile Hubを利用してモバイルアプリ経由で CloudSearchを叩いて、ワードの検索と登録が可能。

# CDP 追加



## ガチ分析パターン

Redshiftだけじゃ物足りない！ やっぱHadoopでしょ！ さらに機械学習もやっちゃうでしょ！ というガチ分析系エンジニアのためのCDP。



## クラウドネイティブパターン

「EC2は使わない。」そう、これがクラウドネイティブエンジニアの合言葉。でも、EC2を憎んでいるわけではありません。

# 特殊あがり(麻雀限定)

## ● リファレンスWebアーキテクチャ(役満)

Route53

CloudFront

ELB

EC2

RDS  
(Multi-AZ)



## ● スシロー(役満)



# 特殊役

- Kinesushi
- Docomo
- セキュリティカン
- AWSロボ

