# AWS麻雀 Ver1.1

JAWS-UG

# ルール（麻雀）

- 基本的に麻雀ルールを踏襲
- ポン・チー・カンあり
- 喰いタン・後付けなし
- 東風のみ　※親の連荘なし
- 持ち点は25000点の30000点返し
  - 〇　符計算は行わない
  - 〇　4ゲームするか、誰かの持ち点がなくなった時点で終了
- 誰かが上がる、もしくは18回ツモで終了（流局）
- JAWS-UG牌はドラとして扱う
- CDPがない場合は点数は0点 ※役満は除く
- CDPがある毎に、1飜増えていく ※同じ牌を再利用してCDPは作れない

# ルール（点数計算）

| | 子(ロン) | 子（ツモ） | 親（ロン） | 親（ツモ） |
|---|---|---|---|---|
| 1飜 | 1000 | 500(500) | 1500 | 1000オール |
| 2飜 | 2000 | 1000(1000) | 3000 | 1500オール |
| 3飜 | 3000 | 1000(2000) | 5000 | 2000オール |
| 4飜 | 5000 | 1500(3000) | 8000 | 3000オール |
| 5飜（満貫） | 8000 | 2000(4000) | 12000 | 4000オール |
| 6,7飜(跳満) | 12000 | 3000(6000) | 18000 | 6000オール |
| 8,9,10飜（倍満） | 16000 | 4000(8000) | 24000 | 8000オール |
| 11,12飜（三倍満） | 24000 | 6000(12000) | 36000 | 12000オール |
| 13飜(役満) | 32000 | 8000(16000) | 48000 | 18000オール |

# ルール（作り方）

- 刻子


同じ種類の牌を3つ集める。 4枚集めた場合カンが可能となる

- 順子


同じ種類の数字を順番に集める。 ※ただし、891などはなし

- 対子


同じ種類の牌を2つ集める

# ルール（上がり方）

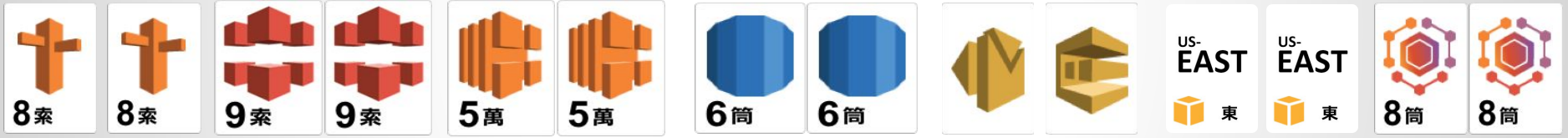
- 手配で刻子or順子を4つ、対子を1つ作れば上がり

例）


# ルール（上がり方例外）

- 例外パターン

例）七対子




例）国士無双

# 役一覧(通常役)

- 1飜
  - 〇　立直(門前)、一発(門前)、門前清模和(門前)、平和(門前)、断ヤオ、一盃口(門前)、役牌(白、發、中、門風牌、荘風牌)、嶺上開花、海底撈月、河底撈魚、槍槓、ドラ
- 2飜
  - 〇　ダブル立直、全帯(鳴き1飜)、混老頭、三色同順(鳴き1飜)、一気通貫(鳴き1飜)、対々和、三色同刻、三暗刻、三槓子、小三元、七対子(門前)
- 3飜
  - 〇　二盃口(門前)、純全帯(鳴き2飜)、混一色(鳴き2飜)
- 6飜
  - 〇　清一色(鳴き5飜)

# 役一覧(役満)

- 四暗刻
- 四暗刻単騎(ダブル)
- 清老頭
- 四槓子
- 大三元
- 字一色
- 小四喜
- 大四喜
- 国士無双
- 国士無双十三面待ち(ダブル)
- 天和
- 地和

※九蓮宝燈、緑一色、大車輪はありません。

# AWS役一覧（役満）

- リファレンスWebアーキテクチャ（役満）

Route53　CloudFront　ELB　EC2　RDS (Muliti-AZ)


- スシロー（役満）

# 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）


- スシロー（役満）

# 特殊役

- Kinesushi
  暗カンで2飜、明カンで1飜
- Docomo
  暗カンで2飜、明カンで1飜
- セキュリティカン
  暗カンで2飜、明カンで1飜
- AWSロボ
  面前のみ2飜

# 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


Elatic
Beanstalk


Auto Scaling


Instances


Elastic Load
Balancing


Amazon Lamda


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

※白に相当

SNS

※撥に相当

API Gateway

※中に相当

# 牌の説明　風牌（ファンパイ）

US-EAST 東

リージョン
バージニア

JAWS-UG

ap-SOUTH east 南

リージョン
シンガポール

JAWS-UG
エンタープライズ

US-WEST 西

リージョン
カリフォルニア

JAWS-UG
中央線

ap-NORTH east 北

リージョン
東京

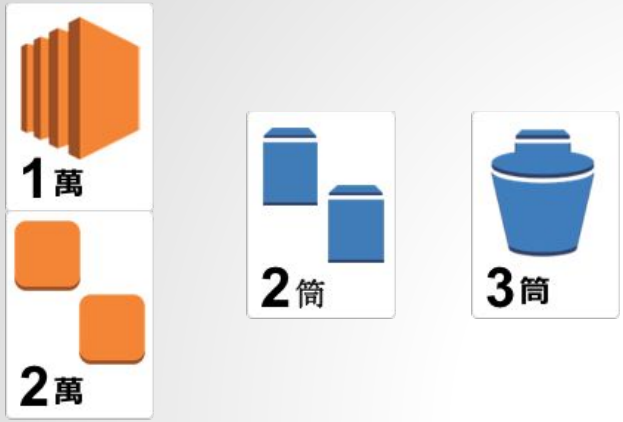
JAWS-UG
女子会

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

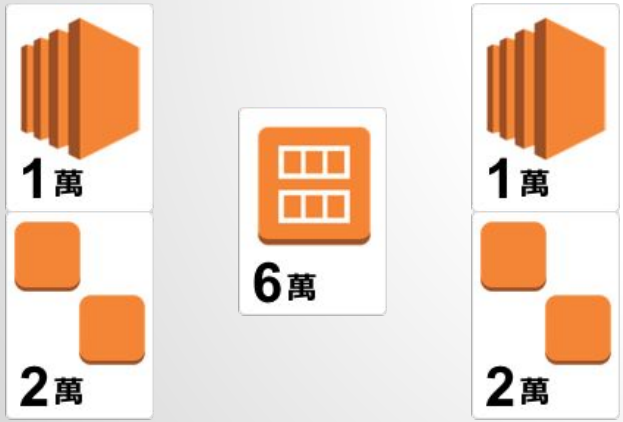
# CDP一覧 Ver1.1

# 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ても便利になる。

# CDP


### 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。


### 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、CloudWatche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。





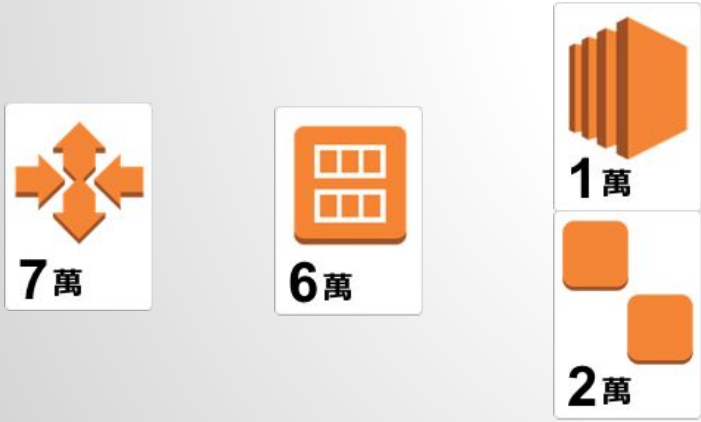

**State Sharing**

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。



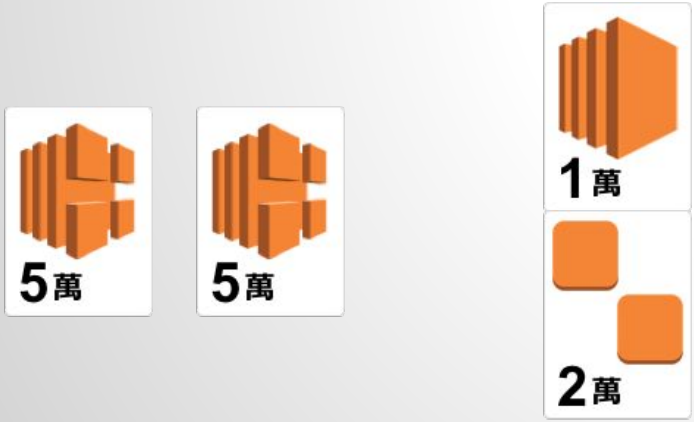

## Scheduled Autoscaling

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Dymamo 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。

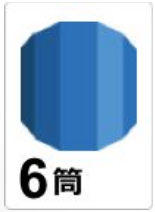
# CDP　追加




**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ー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

Lamda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）


- スシロー（役満）

# 特殊役

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