mijin デプロイメントガイド

Release 1.3（日本語）

Techbureau, Corp

2025 年 07 月 28 日

Contents:

About

1.1 mijin Catapult(v.2) について

1.1.1 mijin Catapult(v.2) とは?

mijin Catapult(v.2) は最小限のリソースで実行可能なプライベートブロックチェーンを構築することができます。

既存のデータベースを置き換えるプライベートブロックチェーン製品として、複数の暗号署名機能を活用し、多くのプロジェクトで利用されています。
取引の原子性を損なうことなく、mijin のアカウントエンジンは、高いセキュリティを前提とした完全分散型、ゼロダウンタイムのネットワークとして稼働し、アプリケーションの開発やメンテナンスの必要性を最小化することができます。
簡単な定義であらゆるデータや資産をトークン化し、すべての取引を安全なワンタイムのスマートコントラクトとして処理し、1 秒間に数千件の取引を行う速度を実現します。

mijin Catapult(v.2) で新たに提供されるのは、**アグリゲートトランザクション**（1,000 トランザクションを原子性を持って同時に実現）や **マルチレイヤーシグネチャ**（3 段階のマルチシグ認証を実現）による原子性が保証された完全分散型のアトミックスワップ技術です。
それら先進的な機能が最低限の知識と工数で実装可能となったのは、すべて V1 における実用ケースで 5 年以上の年月をかけてお客様から得てきたフィードバックの賜物です。
それに NEM という長年の稼働実績があるパブリックブロックチェーンでの経験と技術が合わさったものが、今回 AWS マーケットプレイスにて初めて日本発の商用ブロックチェーン製品として提供開始される mijin Catapult(v.2) です。
以下のサイトでも mijin Catapult(v.2) の説明を詳しく紹介しています。

mijin Catapult(v.2) 説明ページ
https://mijin.io/product/

1.1.2 ユースケース

mijin Catapult(v.2) はさまざまな分野で有効活用可能です。

注釈:

詳しく知りたい方は mijin Site のページの多様な適用分野の項目、又は問い合わせより実例資料を入手することができます。


多様な適用分野
金融系
決済、為替・送金・貯蓄等、
証券取引、BITCOIN取引、
海外送金、ソーシャルバンク
認証
ID、著作権、所有権、各種証明
医療
医療情報
ポイント
ギフトカード交換、アーティスト
向けリワード、プリペイド
カード、リワードトークン
シェアリング
ライドシェア、スペースシェア
データストレージ
資金調達
アーティストエクイティ取引、
クラウドファンディング
商流・物流管理
サプライチェーン、トラッキング管
理、マーケットプレイス、デジタル
アセット管理、移転
資産管理
コミュニケーション
SNS,メッセンジャー取引
コンテンツ
ゲーム、電子書籍、
ストリーミング
教育・人材
学習履歴、履歴・職務経歴書
IoT
製造、センシング、マイニング
チップ
公共
市政予算可視化、投票、
マイナンバー管理
その他
メディア、インフラなど

1.1.3 パブリックブロックチェーン Symbol との違い

mijin Catapult(v.2) はパブリックブロックチェーンとしてローンチされている Symbol と同じコアエンジン (Catapult) を使用しているため、Symbol でできることは mijin Catapult(v.2) でも利用可能です。
ここでは主に Symbol との違いを示します。

表 1: mijin と symbol の比較

項目	mijin Catapult(v.2)	Symbol
チェーン	プライベートブロックチェーン	パブリックブロックチェーン
ネットワーク名	MIJIN, MIJIN_TEST	MAIN_NET, TEST_NET
ジェネシスブロックエポック	1560294000s (Tue 11 Jun 2019 23:00:00 UTC)	1615853185s (Tue 16 Mar 2021 00:06:25 AM UTC)
基軸通貨名	cat.currency	symbol.xym
基軸通貨 ID	mijin Catapult(v.2) 構築時に ID 作成	6BED913FA20223F8
基軸通貨発行量	8,998,999,998.000000	8,164,233,299.724038
ハーベスト通貨名	cat.harvest	symbol.xym（基軸通貨と兼用）
ハーベスト通貨 ID	mijin Catapult(v.2) 構築時に ID 作成	6BED913FA20223F8
ハーベスティング通貨発行量	15,000,000	8,164,233,299.724038（symbol.xym）
ブロック生成間隔	10〜60 秒（構築時にカスタム可能）	30 秒
1 ブロックあたりの最大トランザクション数	6,000 10,000 20,000 50,000 100,000	5,000
トランザクション手数料	あり なし（構築時に指定可能）	あり
Mosaic レンタル手数料	あり（50cat.currency） なし（構築時に指定可能）	あり（50symbol.xym）
Namespace レンタル手数料	あり（期間によって変動） なし（構築時に指定可能）	あり（期間によって変動）
VotingKey ファイルの期限	約 547〜3285 日（ブロック生成間隔により異なる）	約 180 日
1 アカウントあたりの最大署名数	25、50、100、1000（選択可能）	25
ファイナライズ対応	決定的、確率的（構築時に選択可能）	決定的

1.2 主要データベース製品との比較と用途

1.2.1 主要データベース製品との比較

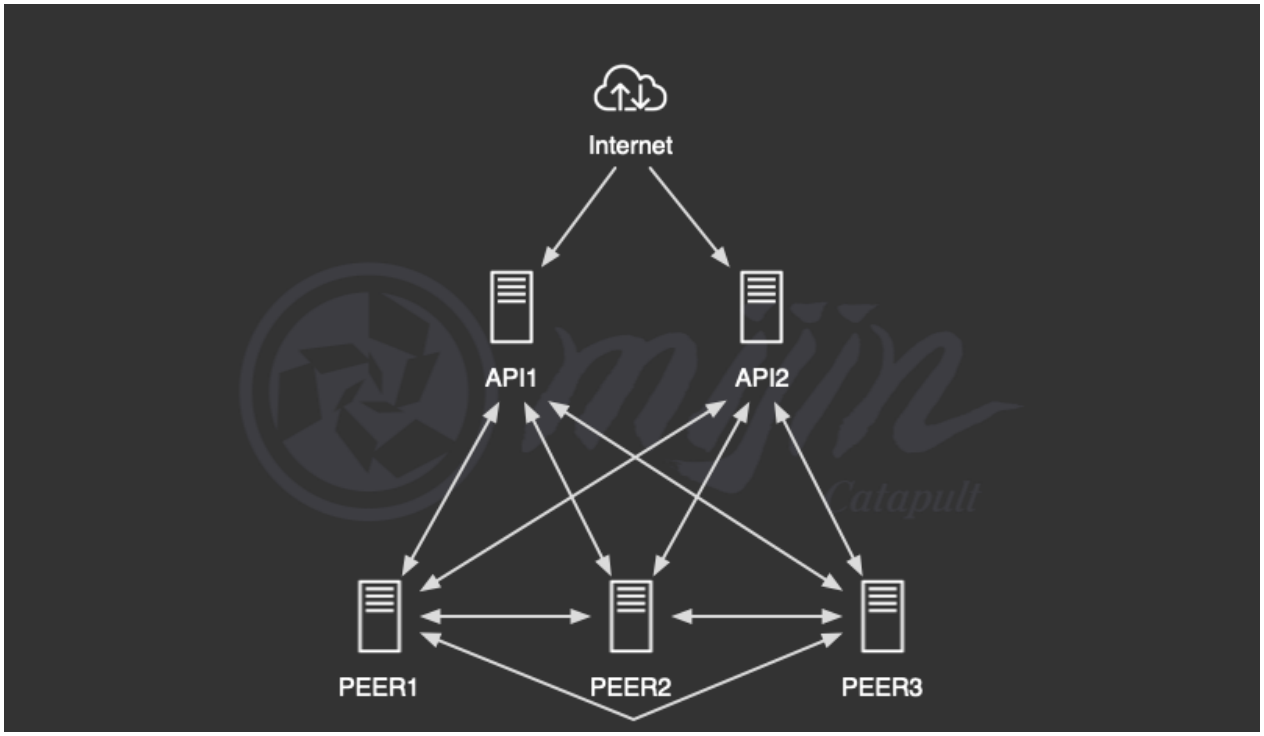
表 2: 各種データベースの比較

項目	mijin v.1	mijin Catapult (v.2)	MySQL	Redis	Mongo	Neo4j	Apache Hbase
種類	ブロックチェーン	ブロックチェーン	RDBMS	NoSQL（KV 型）	NoSQL（ドキュメント型）	NoSQL（グラフ型）	NoSQL（ワイドカラム型）
概要	ブロック単位でデータを格納し、チェーンのように連結する DB	同左	表形式で整合性を保ち、クエリで容易に検索可能	キーバリュー構造で高速アクセスが可能	JSON 形式などをスキーマレスで保存・運用できる DB	関係性をグラフ構造で表現し、高速な結合が可能	列ごとに異なる構造を持ち、高速な集計が可能
ライセンス	商用	商用	GPL / 商用	BSD / 商用	SSPL / 商用	GPL / AGPL / 商用	Apache 2.0
リリース日	2015 年 9 月	2019 年 6 月	1995 年	2009 年	2009 年	2007 年	2010 年
開発言語	Java	C++	C++	C	C++	Java	Java
対応 OS	Linux, Windows	Linux, Windows (要 Docker)	Linux, Windows	Linux, Windows	Linux, Windows	Linux, Windows	Linux
クライアント	REST API, nem-sdk	REST API, symbol-sdk (Java, TS)	mysql-client, 各言語ライブラリ	redis-client, 各言語ライブラリ	mongo-client, 各言語ライブラリ	REST API, WEB UI 各言語ライブラリ	HBase client, RPC 対応ライブラリ
利点	改ざん耐性が高く、勘定系アセット構築が容易	上記同様	クエリが強力、整合性保証 情報量が多い	シンプル API、メモリ常駐で高速	スキーマレス、水平スケーラブル	複雑構造でも高速検索、クエリ対応	列志向で集計が高速、ビッグデータに強い
主な用途	ポイント管理、監査ログ	ポイント管理、P2P 取引	顧客管理	セッション/メッセージ中継	ログ、ゲーム、アンケート	レコメンド、関係性分析	分析、統計、集計用途

1.3 アーキテクチャとスペック要件

1.3.1 mijin Catapult(v.2) の構成について

mijin Catapult (v.2) はノードと呼ばれるサーバ郡を構成し、プライベートブロックチェーンネットワークを構成します。
mijin では、2 台の API ノード、3 台の PEER ノードの構成を最小の推奨構成としています。



1.3.2 PEER ノードの役割とスペック要件

PEER ノードは、ブロックチェーンデータを生成、ブロックのコンセンサス機能を提供します。
PEER ノードだけではクライアントからはアクセスできず、API ノードが必要になります。

CPU	3.1 GHz 程度のプロセッサ (CPU コア 2 以上)
メモリ	4GB RAM 以上
ディスク	root 30GB 以上 block 500GB 以上 3000IOPS 以上推奨
OS	Docker が稼働する Linux(推奨 Ubuntu 20.04 以上)

注釈:

AWS MarketPlace の mijin Catapult(v.2) では、推奨以上の使用可能なスペックのみを選択することができます。
ディスク容量は、ブロックチェーンへの保存量に比例して増加します。

1.3.3 API ノードの役割とスペック要件

API ノードは、ブロックチェーンデータを mongodb に書き込み、高速で読み込める API を提供します。
API ノード上にも、ブロックチェーンデータはありますが、ブロック生成機能はなく、単純なバックアップとして扱われます。
このブロックチェーンデータの生成機能を有効にすることもでき、PEER ノードの機能を有した DUAL モードとして機能することができます。

CPU	3.1 GHz 程度のプロセッサ (CPU コア 2 以上)
メモリ	8GB RAM 以上
ディスク	root 30GB 以上 mongo 300GB 3000IOPS 以上推奨 block 500GB 3000IOPS 以上推奨
OS	Docker が稼働する Linux(推奨 Ubuntu 20.04 以上)

注釈:

AWS MarketPlace の mijin Catapult(v.2) では、推奨以上の使用可能なスペックのみを選択することができます。
ディスク容量は、ブロックチェーンへの保存量に比例して増加します。

Deploy

2.1 mijin Catapult(v.2) のデプロイ方法

2.1.1 mijin のデプロイ方法について

mijin Catapult(v.2) を構築する方法は、現在二通りとなります。

1. AWS MarketPlace を使用したデプロイ
2. テックビューロによる構築

2.1.1.1 AWS MarketPlace を使用したデプロイ

クラウドベンダー最大手の一つである Amazon Web Service(AWS) のマーケットプレイスにて、mijin Catapult(v.2) を容易にデプロイすることができる製品を提供しています。

AWS MarketPlace
https://aws.amazon.com/marketplace

詳しくは、AWS MarketPlace を使ったデプロイ準備 を参照してください。

2.1.1.2 テックビューロによる構築

テックビューロにて、mijin 構築を行います。
詳しくは、mijin Site の問い合わせページよりお問い合わせください。

2.2 AWS MarketPlace

AWS MarketPlace は、サードパーティーのソフトウェア、サービス、およびデータの調達、プロビジョンする AWS のサービスとなります。
この章では、AWS MarketPlace に展開する mijin Catapult(v.2) の説明をしています。

2.2.1 AWS Marketplace を使ったデプロイ準備

AWS Marketplace にて提供している mijin Catapult(v.2) を使用するには、Amazon Web Services(AWS) のアカウントおよび操作の知識が必要になります。
本章では、デプロイまで事前に準備が必要となる内容を理解することができます。

2.2.1.1 AWS アカウントの準備

Amazon Web Services(AWS) はクラウド上にコンピューティング、ストレージ、データベースなどのインフラストラクチャテクノロジーを作成するプラットフォームになります。
操作を行うには、AWS アカウント作成が必要になるため、以下のサイトを参照し、AWS アカウントを作成してください。

https://aws.amazon.com/jp/register-flow/

警告:

mijin Catapult(v.2) をデプロイする際は　ルートユーザーではなく管理用 IAM アカウントを作成しデプロイすることを推奨します。
管理 IAM ユーザーの作成は以下を参照してください。
https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/getting-started_create-admin-group.html
ルートユーザーに関して知りたい方は、以下を参照してください。
https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/id_root-user.html

2.2.1.2 AWS の知識

mijin が使用する AWS サービスは主に以下となります。
Marketplace で提供している mijin Catapult(v.2) はデプロイ時に自動で以下を作成されます。

- **Amazon VPC**

  仮想ネットワークを構築します。mijin Catapult(v.2) はこの VPC 上のネットワークに配置されます。
  詳しくは以下のドキュメントを確認してください。
  https://docs.aws.amazon.com/ja_jp/vpc/latest/userguide/what-is-amazon-vpc.html

- **Amazon EC2**

  コンピューティングマシンを作成します。mijin Catapult(v.2) のノード (サーバ) を作成します。
  詳しくは以下のドキュメントを確認してください。
  https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/concepts.html

- **Amazon EBS**

  コンピューティングマシンにストレージを作成します。

mijin Catapult(v.2) では、ブロックチェーンデータおよび mongo データを EBS 上に配置します。
詳しくは以下のドキュメントを確認してください。
https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/AmazonEBS.html

- **Elastic Load Balancing**

  コンピューティングマシンに中継する負荷分散装置を作成します。
  mijin Catapult(v.2) では API ノードへのアクセスをロードバランサ経由にすることで冗長性を確保します。
  詳しくは以下のドキュメントを確認してください。
  https://docs.aws.amazon.com/ja_jp/elasticloadbalancing/latest/userguide/what-is-load-balancing.html

- **Amazon Route 53**

  コンピューティングマシンの名前解決に使用する DNS サービスを作成します。
  mijin Catapult(v.2) では、各ノード間を DNS で名前解決して接続しています。
  詳しくは以下のドキュメントを確認してください。
  https://docs.aws.amazon.com/ja_jp/Route53/latest/DeveloperGuide/Welcome.html

- **AWS IAM**

  AWS の各サービス間など、AWS のサービスやリソースにアクセスできるユーザーやグループを指定し、
  きめ細かいアクセス許可を一元管理します。
  mijin Catapult(v.2) デプロイ時、AWS アカウントは IAM 権限が必要となります。(AWS にデプロイするためのアカウント権限 にて説明)

  mijin Catapult(v.2) デプロイ後に作成されたリソースに、
  ・EC2 インスタンスから Secrets Manager への権限
  ・EC2 インスタンスから SSM ログイン権限
  を API,PEER ノードである EC2 インスタンスのプロファイルに割り当てます。

  EC2 の IAM ロールに関しては以下を参照してください。
  https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html

  SSM ログインについて詳しくは以下を参照してください。
  https://docs.aws.amazon.com/ja_jp/systems-manager/latest/userguide/session-manager-getting-started-instance-profile.html

- **AWS Systems Manager Parameter Store**

  設定データ管理と機密管理のための安全な階層型ストレージとして利用します。
  mijin Catapult(v.2) では、各ノードの設定情報や共通設定を保存します。
  パラメータストアに保存することで、万が一障害が発生した場合でも復元することが可能です。
  パラメータストアに関しては、以下を参照してください。
  https://docs.aws.amazon.com/ja_jp/systems-manager/latest/userguide/systems-manager-parameter-store.html

- **AWS CloudFormation**

  AWS サービスを自動で構築するオーケストレーションを提供します。
  AWS Marketplace は AWS CloudFormation を使用して、AWS に mijin Catapult(v.2) を構築します。
  詳しくは以下のドキュメントを確認してください。
  https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/Welcome.html

2.2.1.3 AWS にデプロイするためのアカウント権限

AWS に mijin Catapult(v.2) をデプロイするためには、AWS アカウントの準備 で準備したアカウントに権限を付与する必要があります。
権限としてはAWS の知識 の AWS リソースを作成する権限及び Marketplace で展開する mijin Catapult(v.2) のイメージを使用するための Subscribe する権限が必要です。

AWS Marketplace 権限およびデプロイ権限のみに絞った IAM ポリシーは、以下となります。
以下を参考に、IAM ポリシーを作成し、デプロイに使用する IAM アカウントに付与してください。
https://docs.aws.amazon.com/ja_jp/apigateway/latest/developerguide/api-gateway-create-and-attach-iam-policy.html

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lambda.amazonaws.com",
                        "ec2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "iam:*RolePolicy",
                "route53:*HostedZone",
                "iam:List*",
                "aws-marketplace:*",
                "elasticloadbalancing:RegisterTargets",
                "ec2:*RouteTable*",
                "iam:CreateRole",
                "elasticloadbalancing:DeleteLoadBalancer",
                "ec2:DescribeInternetGateways",
                "elasticloadbalancing:DescribeLoadBalancers",
                "ec2:*KeyPairs",
```

(continues on next page)

(continued from previous page)

```
                "lambda:GetFunction*",
                "ec2:DescribeAccountAttributes",
                "elasticloadbalancing:ModifyTargetGroupAttributes",
                "cloudformation:ListStackResources",
                "iam:GetRole",
                "elasticloadbalancing:DescribeTarget*",
                "elasticloadbalancing:CreateTargetGroup",
                "route53:ChangeResourceRecordSets",
                "iam:DeleteRole",
                "cloudformation:GetTemplate*",
                "ec2:RunInstances",
                "elasticloadbalancing:*Listener",
                "ec2:*Addresses",
                "cloudformation:GetStackPolicy",
                "cloudformation:DeleteStack",
                "ec2:DeleteVpc",
                "iam:GetAccountSummary",
                "ec2:*Tags",
                "route53:GetChange",
                "iam:*InstanceProfile*",
                "ec2:*InternetGateway",
                "ec2:CreateVpc",
                "sns:ListTopics",
                "route53:ListQueryLoggingConfigs",
                "cloudformation:*ChangeSet*",
                "cloudformation:EstimateTemplateCost",
                "elasticloadbalancing:DescribeListeners",
                "ec2:DescribeAvailabilityZones",
                "ec2:ModifyVpcAttribute",
                "ec2:*Route",
                "ec2:ReleaseAddress",
                "cloudformation:ListStacks",
                "elasticloadbalancing:CreateLoadBalancer",
                "ec2:TerminateInstances",
                "elasticloadbalancing:DeleteTargetGroup",
                "ec2:AllocateAddress",
                "cloudformation:DescribeStack*",
                "lambda:AddPermission",
                "cloudformation:CreateStack",
                "ec2:DescribeVpcs",
                "lambda:*Function",
                "ec2:*NatGateway*",
                "lambda:RemovePermission",
                "ec2:*SecurityGroup*",
                "ec2:*Subnet*",
                "ec2:DescribeInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

2.2.1.4 AWS における mijin Catapult(v.2) ライセンス

AWS Marketplace の mijin Catapult(v.2) 製品版のライセンスは、Marketplace にある mijin Catapult(v.2) の EULA に同意することで使用可能になります。
EULA に関しては、以下を参照してください。
https://d7umqicpi7263.cloudfront.net/eula/product/d6b2653b-ee61-4a62-8fef-a9fa7930892e/c255cb3f-6c72-412a-a899-42fa3f83fd71.pdf

AWS Marketplace の mijin Catapult(v.2) 製品版のライセンス費は、AWS 利用料金に合算され、**時間単位**で従量課金されます。
スペックや CPU コア数は関係なく、1 ノード単位です。

項目	値
mijin ライセンス費	$0.40/1 時間

注釈:

AWS MarketPlace Trial Version は mijin Catapult(v.2) ライセンス費用は発生しません。
AWS MarketPlace Enterprise x86_64 Version のノード起動は最低 5 台のため、1 ヶ月にかかるおよそのライセンス費の計算は以下となります。

■ 1 時間単位
$0.4/1h * 5(台) = $2.0/1h
■ 1 日単位
$2.0/1h * 24(h) = $48.0/1d
■ 1 ヶ月 (30 日) 単位
$48/1d * 30(h) = $1,440.0/1m

2.2.1.5 AWS 利用料金

mijin のライセンス費の他、AWS リソース使用料として、

・Amazon EC2
・Amazon EBS
・Elastic Load Balancing
・Amazon Route53
・Amazon VPC(Nat Gateway)
・パラメータストア

の従量課金の費用が発生します。
また、データ転送量などでも、料金が変動します。
AWS 利用料金に関しては、以下を参照してください。

https://aws.amazon.com/jp/pricing/

注釈:

AWS Marketplace で提供している mijin Catapult(v.2) は指定したパラメーターによって構成が変わり、費用が変わります。
主にパラメーター指定によって、以下が変動することに留意してください。
・インスタンスタイプ
・ノード数 (インスタンス数)
・Elastic Load Balancing の必要可否
・VPC の作成有無
・EBS のブロックサイズ・IOPS

AWS Marketplace の製品説明

現在、テックビューロから提供している製品は以下となります。
各製品ごとに、AWS Marketplace 上に製品ページがあります。

- トライアル版 (AWS MarketPlace Trial Version)
- 製品版 x86_64 版 (AWS MarketPlace Enterprise x86_64 Version)
- 製品版 arm64 版 (AWS MarketPlace Enterprise arm64 Version)

AWS Marketplace で提供している mijin Catapult(v.2) は、構築に複雑な操作をする必要がありません。
デプロイ時に選んだパラメータによって、自動でネットワークを含んだ環境を構築し、mijin を含んだイメージから安全かつ堅牢なブロックチェーンネットワークを構築します。

環境の構築される内容については、各デプロイ方法のページにて説明しています。

2.2.1.6 トライアル版

トライアル版は、簡易に起動できる mijin Catapult(v.2) として、ライセンス無償でお試しいただくことが可能です。
冗長性はなく、単一稼働の mijin Catapult(v.2) になります。
製品版に比べ、カスタム性がなく、シングル AZ の配置ですが、試しに使ってみる環境や開発環境に適しています。

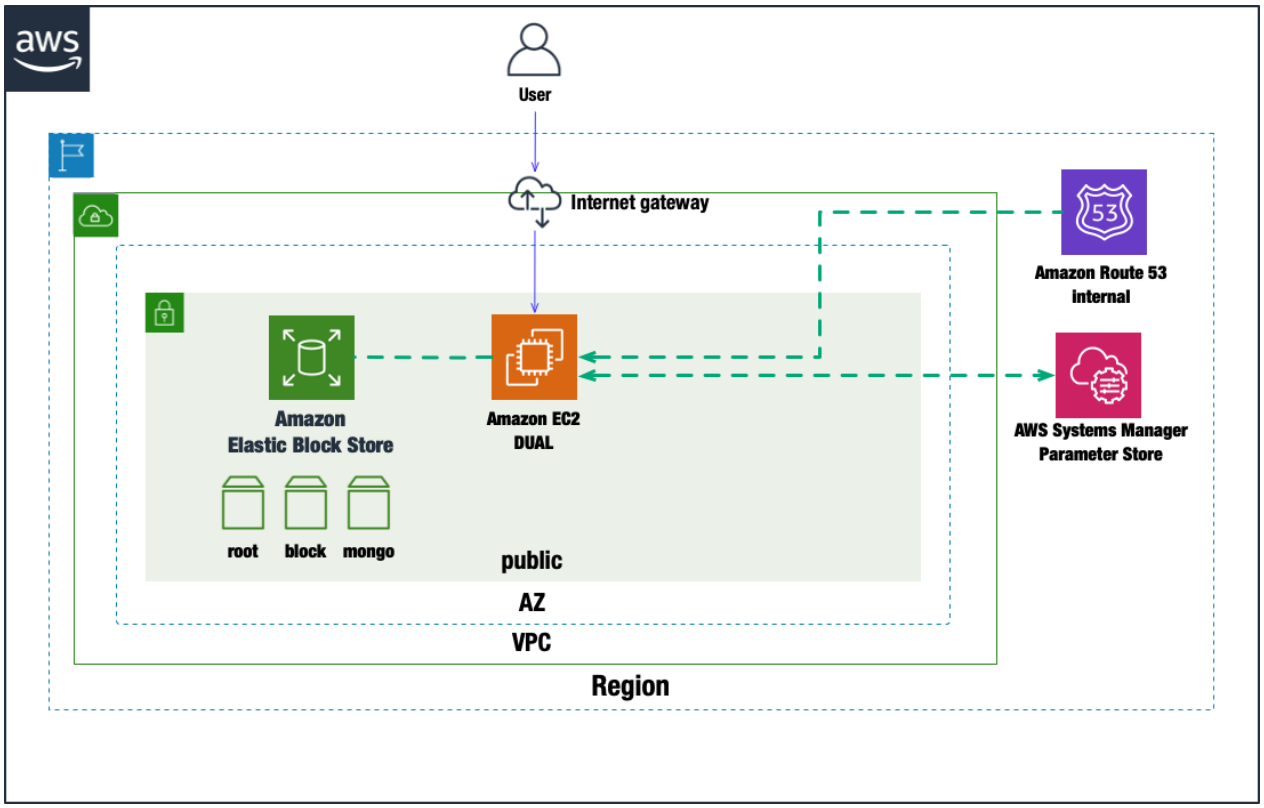


表 1: 無償トライアル環境概要

項目	説明
マーケットプレイス製品ページ	AWS MarketPlace Trial Version
デプロイ方法	新規の VPC に mijin Catapult(v.2) を構築 トライアル版の mijin をデプロイする を参照
デプロイにかかる時間	約 15 分
サポート	基本サポートなし AWS に関する問題は以下で問い合わせ： https://aws.amazon.com/jp/premiumsupport/
使用可能リージョン	世界 21 リージョンで使用可能 ap-northeast-1, us-west-1, us-west-2, us-east-1, us-east-2, eu-north-1, eu-west-1, eu-west-2, eu-west-3, eu-south-1, af-south-1, ap-south-1, ap-east-1, ap-northeast-2, ap-northeast-3, ap-southeast-1, ap-southeast-2, sa-east-1, ca-central-1, eu-central-1, me-south-1
ノード数	1 台（DUAL モード）
配置アベイラビリティゾーン	シングル AZ（1 つ）
ロードバランサー	なし

2.2.1.7 製品版

製品版は、エンタープライズ向けに本番運用を想定したカスタム性がある構成を自由に設定することができ、耐障害性、高可用性のある構成を標準とし、且つ API へのアクセス方法もセキュアなアクセスを容易に構築することが可能です。
また、mijin Catapult(v.2) を既存・新規環境を問わず設置することができ、ロードバランサを使った負荷分散など様々な環境に適用することができます。

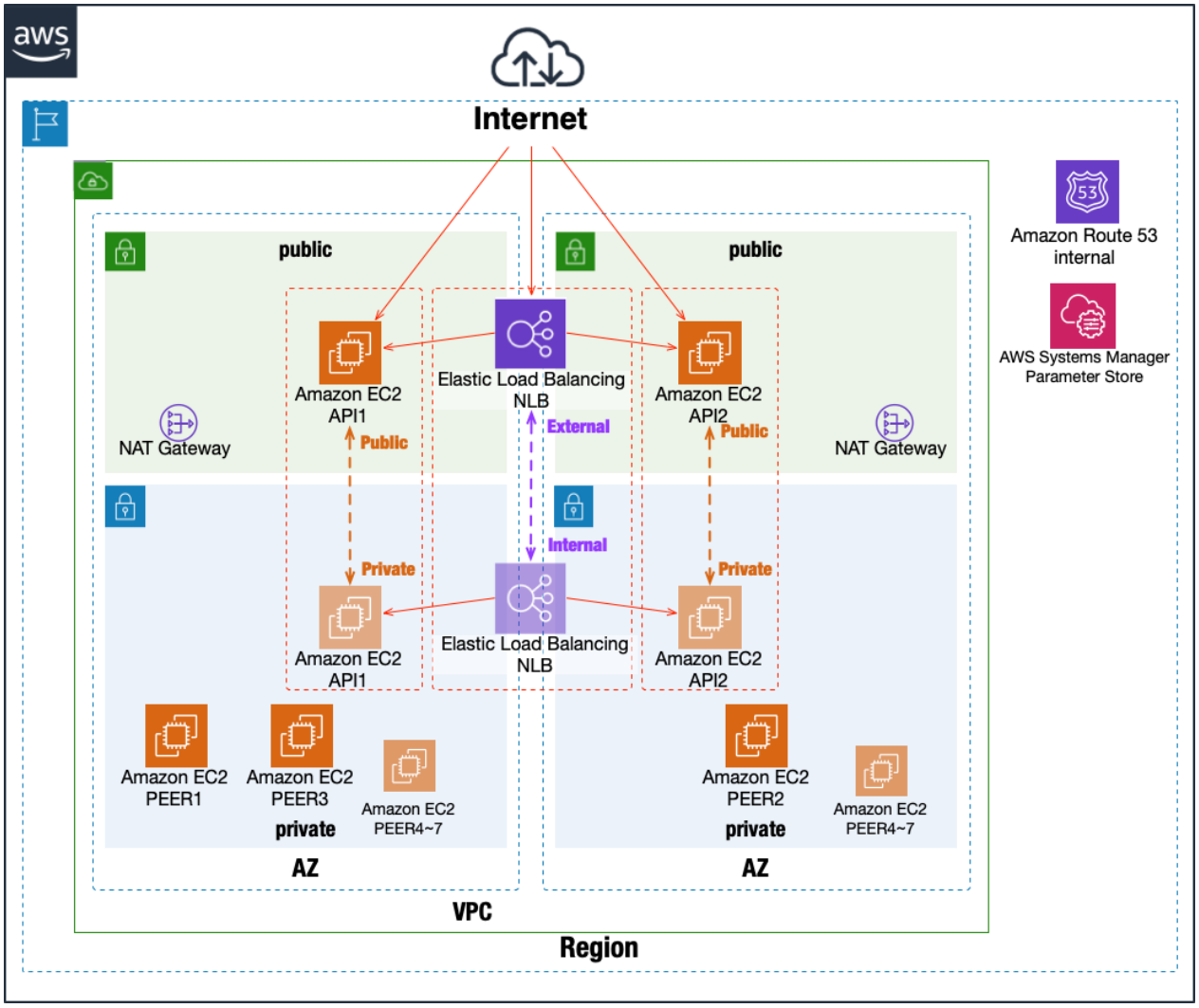


表 2: AWS デプロイ情報

項目	説明
マーケットプレイス製品ページ	x86_64 版：AWS MarketPlace Enterprise x86_64 Version arm64 版：AWS MarketPlace Enterprise arm64 Version
既存環境へのデプロイ方法	すでに既存サービスがあり、 同じ VPC 内で mijin Catapult(v.2) を管理したい場合は 既存 VPC 上に、mijin をデプロイする を参照
新規環境へのデプロイ方法	mijin Catapult(v.2) を新規に構築する、 初めて AWS を使う場合は 新規 VPC を作成し、mijin をデプロイする を参照
デプロイにかかる時間	約 30 分
サポート	デプロイ関連のお問い合わせは https://mijin.io/aws_contact/ ※ mijin Catapult(v.2) に関する技術的な質問は 有償サポート（チケット）で対応 AWS 関連は AWS サポートケースへ： https://aws.amazon.com/jp/premiumsupport/
使用可能リージョン	世界 21 リージョンで使用可能 ap-northeast-1, us-west-1, us-west-2, us-east-1, us-east-2, eu-north-1, eu-west-1, eu-west-2, eu-west-3, eu-south-1, af-south-1, ap-south-1, ap-east-1, ap-northeast-2, ap-northeast-3, ap-southeast-1, ap-southeast-2, sa-east-1, ca-central-1, eu-central-1, me-south-1
ノード数	API ノード ×2、PEER ノード ×3〜7（合計 5〜9 台）
配置 AZ（アベイラビリティゾーン）	マルチ AZ（2 つ） ※ シングル AZ は設定不可
ロードバランサー	必要に応じて設定可能

2.2.1.8 有償サポートについて

mijin Catapult(v.2) のデプロイに関するサポートは無償で行いますが、mijin Catapult(v.2) に関する技術問合せが必要な場合、有償のサポート (チケット制) を購入することでテックビューロ社より支援することが可能です。
以下より、サポート購入の旨をお問合せください。

https://mijin.io/aws_contact/

サポート内容	mijin Catapult(v.2) に関する技術問合せ ノードの障害サポート (営業時間内) バージョンアップの通知・手順公開 AWS 環境におけるインフラ支援 など

2.2.1.9 AWS のサービスクォータによる制限について

AWS Marketplace mijin Catapult(v.2) は、以下の AWS サービスを使用するため、すでに環境が構築されている AWS アカウントを利用する場合、サービスクォータによる、起動失敗する可能性があります。
サービスクォータに関しては、以下に説明があります。

https://docs.aws.amazon.com/ja_jp/general/latest/gr/aws_service_limits.html

以下に、mijin Catapult(v.2) のサービス作成数を説明します。

表 3: AWS サービスと制限

AWS サービス	制限に関連する内容
Amazon VPC	VPC: 1 サブネット: 4（トライアルは 1） インターネットゲートウェイ: 1 NAT Gateway: 2（トライアルは 0） ルートテーブル: 1 セキュリティグループ: 5 参考: https://docs.aws.amazon.com/ja_jp/vpc/latest/userguide/amazon-vpc-limits.html
Amazon EC2	EC2 インスタンス: 5〜9（トライアルは 1） インスタンスタイプによりクォータ変動あり EC2 制限: https://docs.aws.amazon.com/ja_jp/general/latest/gr/ec2-service.html オンデマンド制限: https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/ec2-on-demand-instances.html#ec2-on-demand-instances-limits EBS: 630GB〜 https://docs.aws.amazon.com/ja_jp/general/latest/gr/ebs-service.html
Elastic Load Balancing	Network Load Balancer: 1（有効時） https://docs.aws.amazon.com/ja_jp/elasticloadbalancing/latest/network/load-balancer-limits.html
Amazon Route53	Internal ゾーン: 1 https://docs.aws.amazon.com/ja_jp/Route53/latest/DeveloperGuide/DNSLimitations.html
AWS IAM	IAM ロール: 2（トライアルは 1） IAM ポリシー: 2（トライアルは 1） https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/reference_iam-quotas.html
Systems Manager Parameter Store	パラメーター数: 48〜(トライアルは 15) https://docs.aws.amazon.com/ja_jp/general/latest/gr/ssm.html

2.2.2 新規 VPC を作成し、mijin をデプロイする

本章では、新規ネットワーク (VPC) に mijin Catapult(v.2) をデプロイする方法を示します。
製品版では、パラメーターを変更することで環境にあったネットワークを柔軟に構築することが可能です。

2.2.2.1 デプロイによって AWS 上に構築するサービス一覧

- Amazon EC2 (API ノード x 2 PEER ノード x 3~)
- Amazon EBS
- Elastic Load Balancing
- Amazon Route53
- Amazon VPC(Nat Gateway)
- パラメータストア

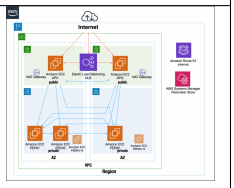
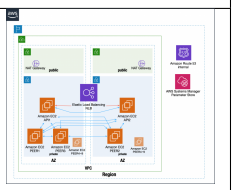
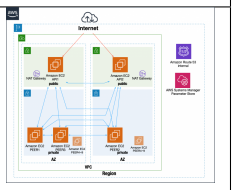
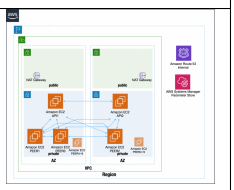
注釈:

Elastic Load Balancing は、デプロイ時に指定するパラメーターの設定により、作成しないケースがあります。

2.2.2.2 View Network

製品版では、パラメーターを変更することで環境にあったネットワークを構築することが可能です。以下にパターン例を示します。

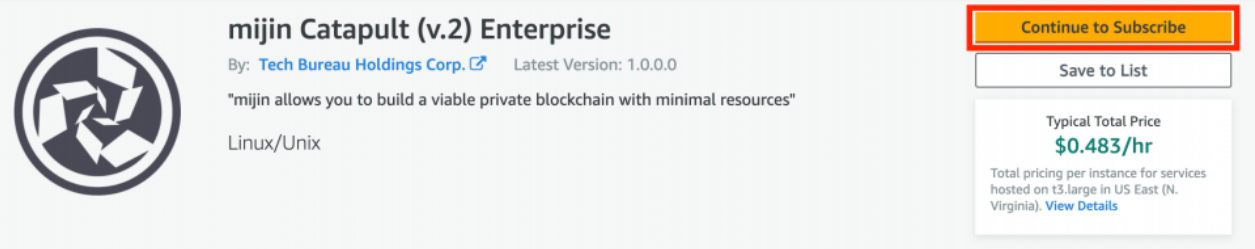
表 4: ロードバランサーとノード配置パターン

No	ロードバランサー	API ノード配置	PEER ノード配置	図
1	**あり / 公開 NW** UseLoadBalancer: `Yes` LoadBalancerType: `external`	**公開 NW** ApiPlacementNetwork: `Public`	非公開 NW	
2	**あり / 非公開 NW** UseLoadBalancer: `Yes` LoadBalancerType: `internal`	**非公開 NW** ApiPlacementNetwork: `Private`	非公開 NW	
3	**なし** UseLoadBalancer: `No` LoadBalancerType は設定無効	**公開 NW** ApiPlacementNetwork: `Public`	非公開 NW	
4	**なし** UseLoadBalancer: `No` LoadBalancerType は設定無効	**非公開 NW** ApiPlacementNetwork: `Private`	非公開 NW	

注釈:

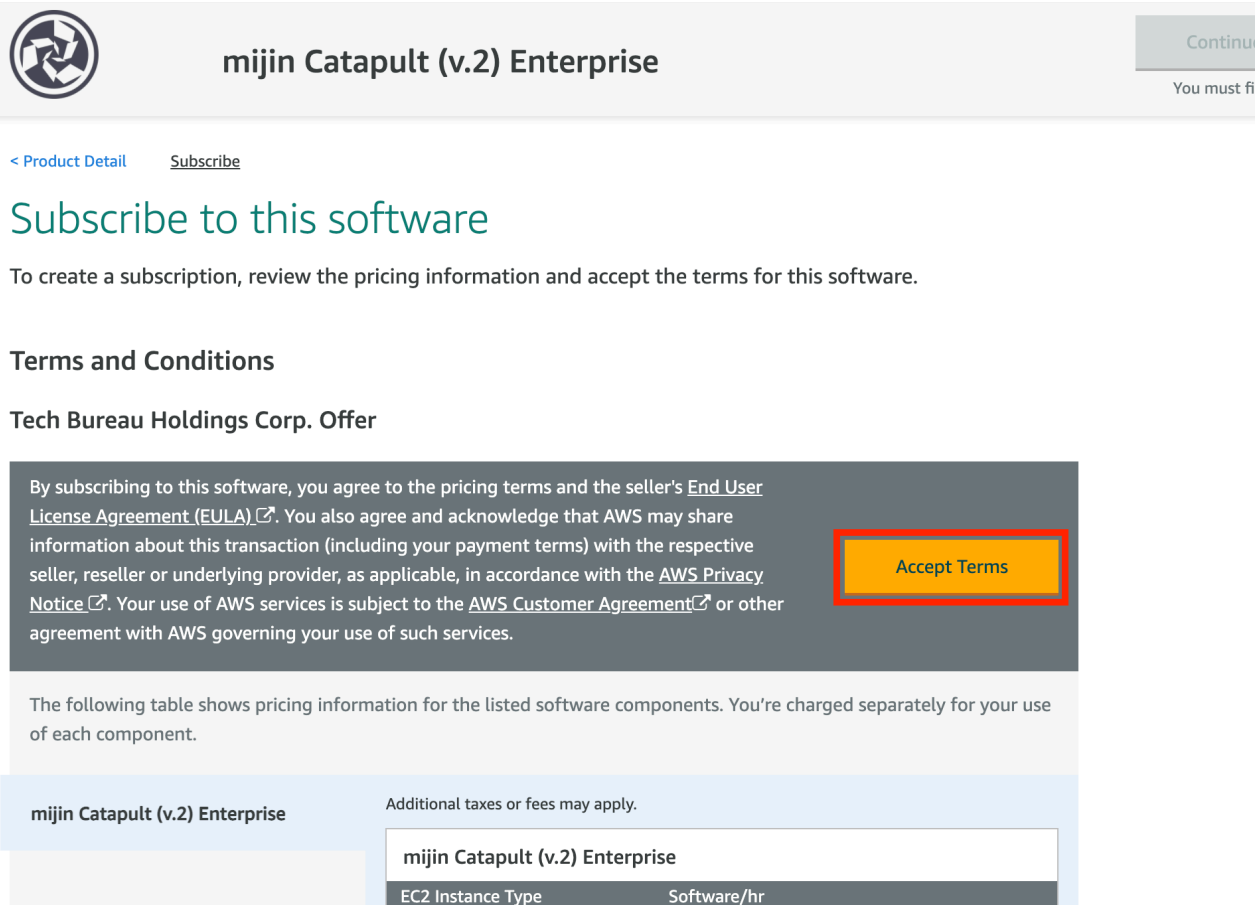
製品版では、高可用性を実現するため、シングル AZ で構成はできず、マルチ AZ のみの構成となっています。
必ずパブリックネットワーク 2 つ、プライベートネットワーク 2 つが必要になることに留意してください。
配置はシングルリージョンとなりますが、世界 20 リージョン毎にデプロイすることが可能です。

2.2.2.3 Step.1



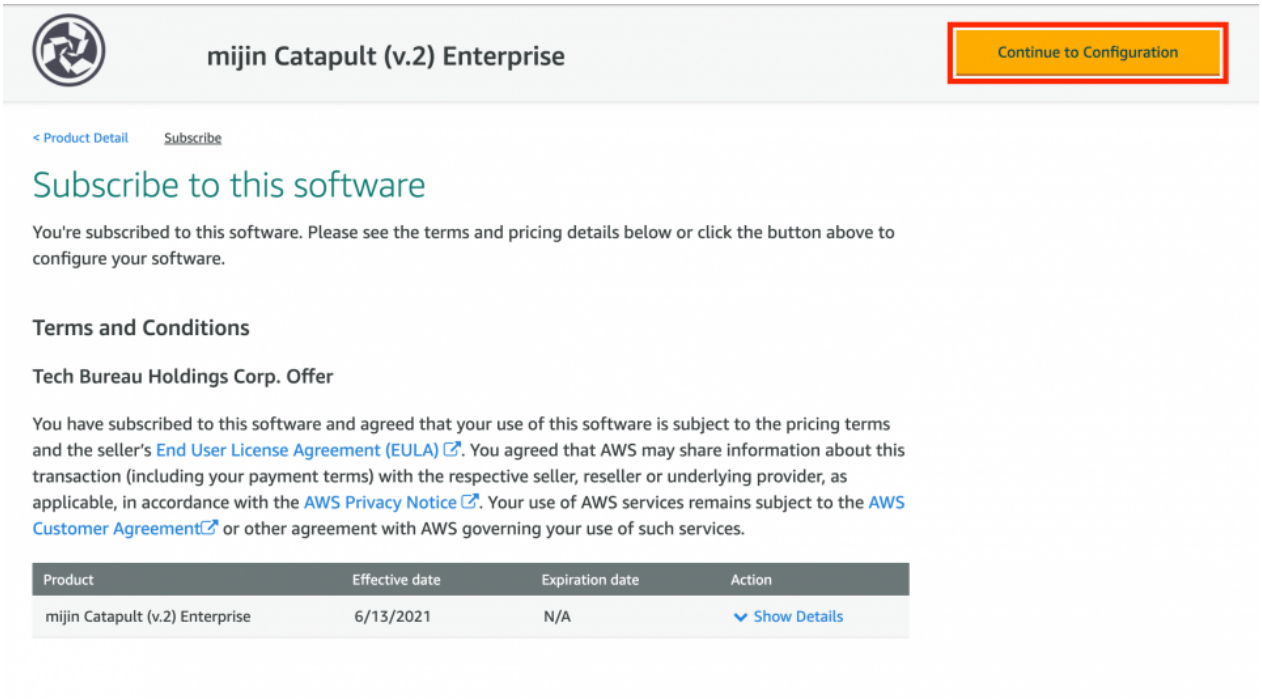
mijin Catapult Enterprise の AMI を使用するためにサブスクライブする必要があります。赤枠のボタンを押してください。

2.2.2.4 Step.2



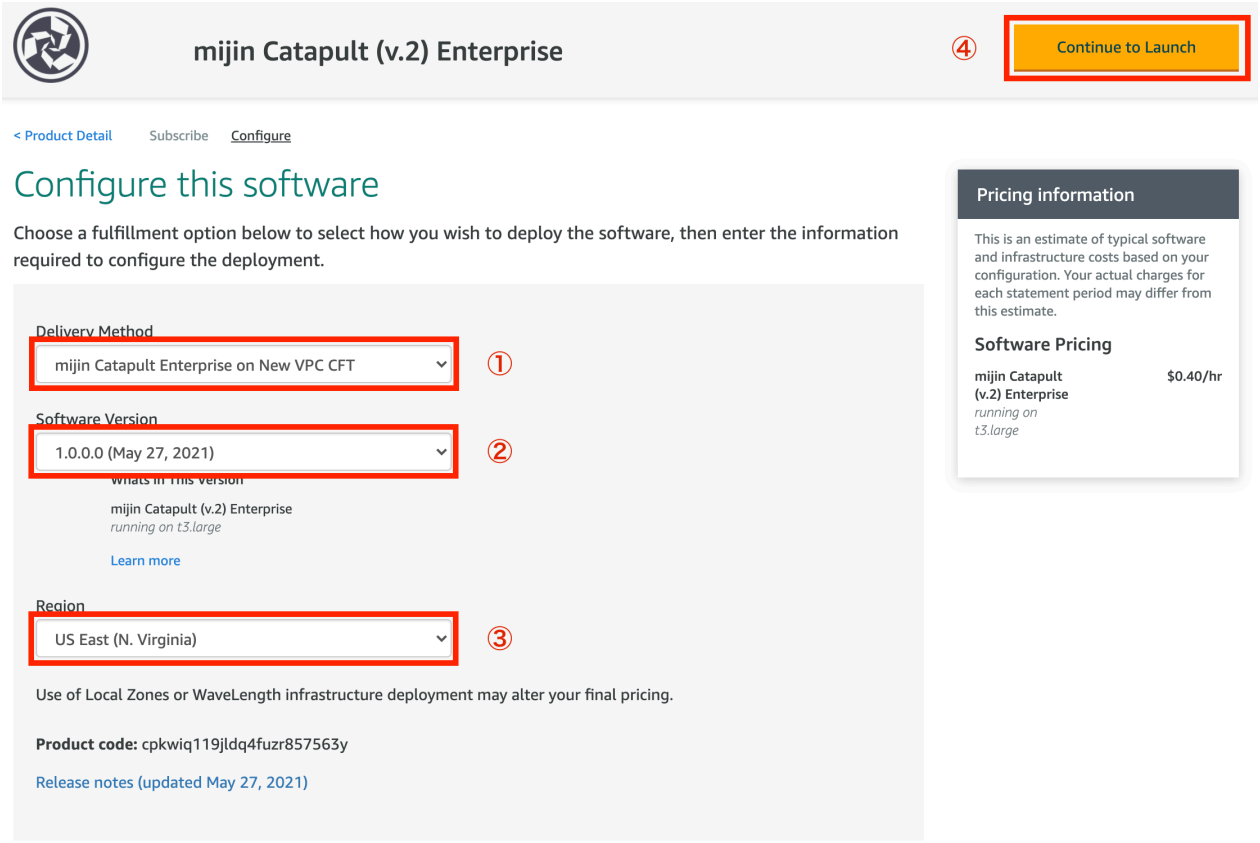
mijin Catapult Enterprise AMI を使用するため、使用の承認をしてください。

2.2.2.5 Step.3



日: Config から起動するために、赤枠の文字をクリックしてください。

2.2.2.6 Step.4



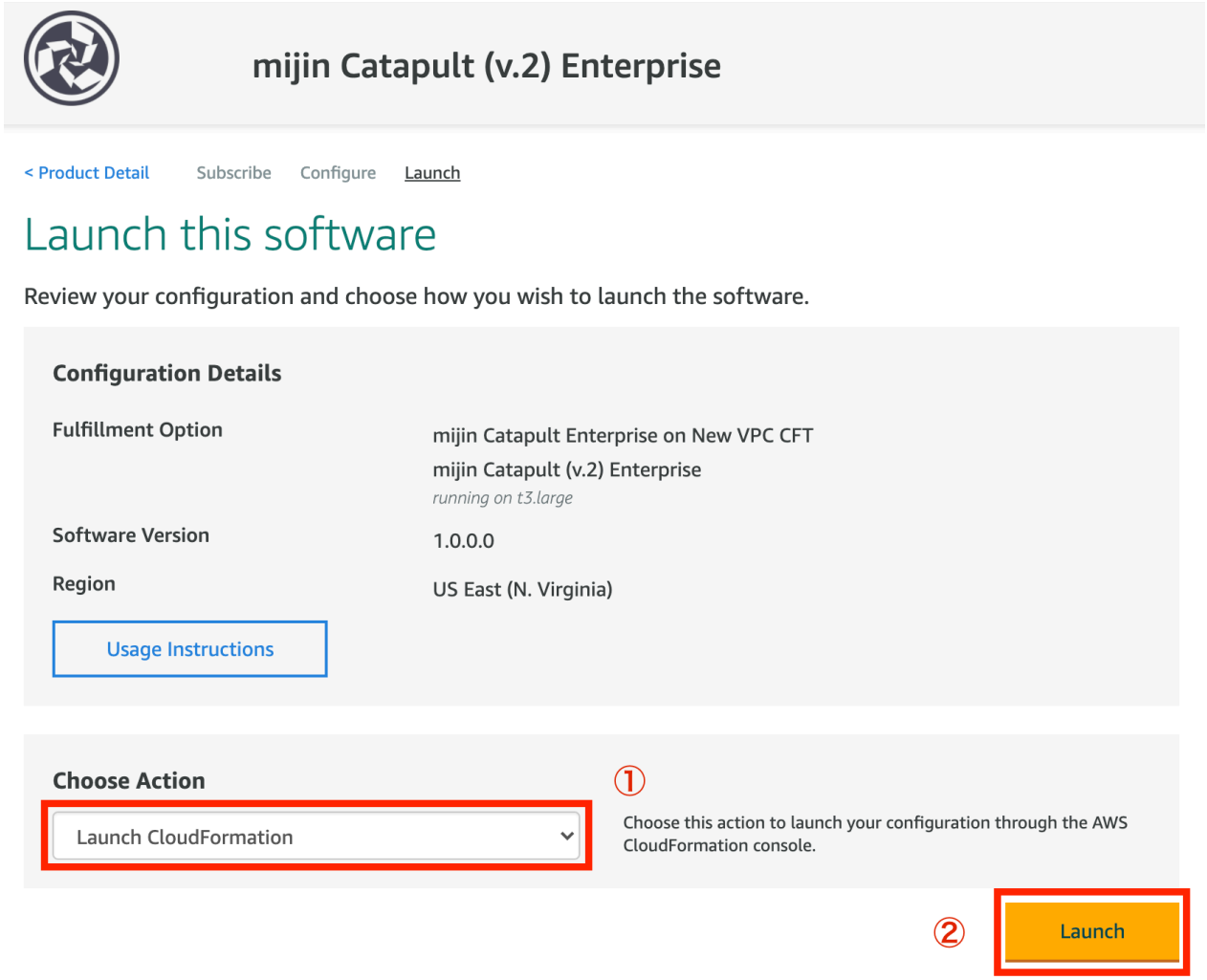
① の赤枠にて使用するテンプレートを選択します。ここでは新規ネットワークを作成するため「mijin Catapult Enterprise on New VPC CFT」を指定します。

② の赤枠ににて mijin のバージョンを指定します。

③ mijin を展開するリージョンを指定します。

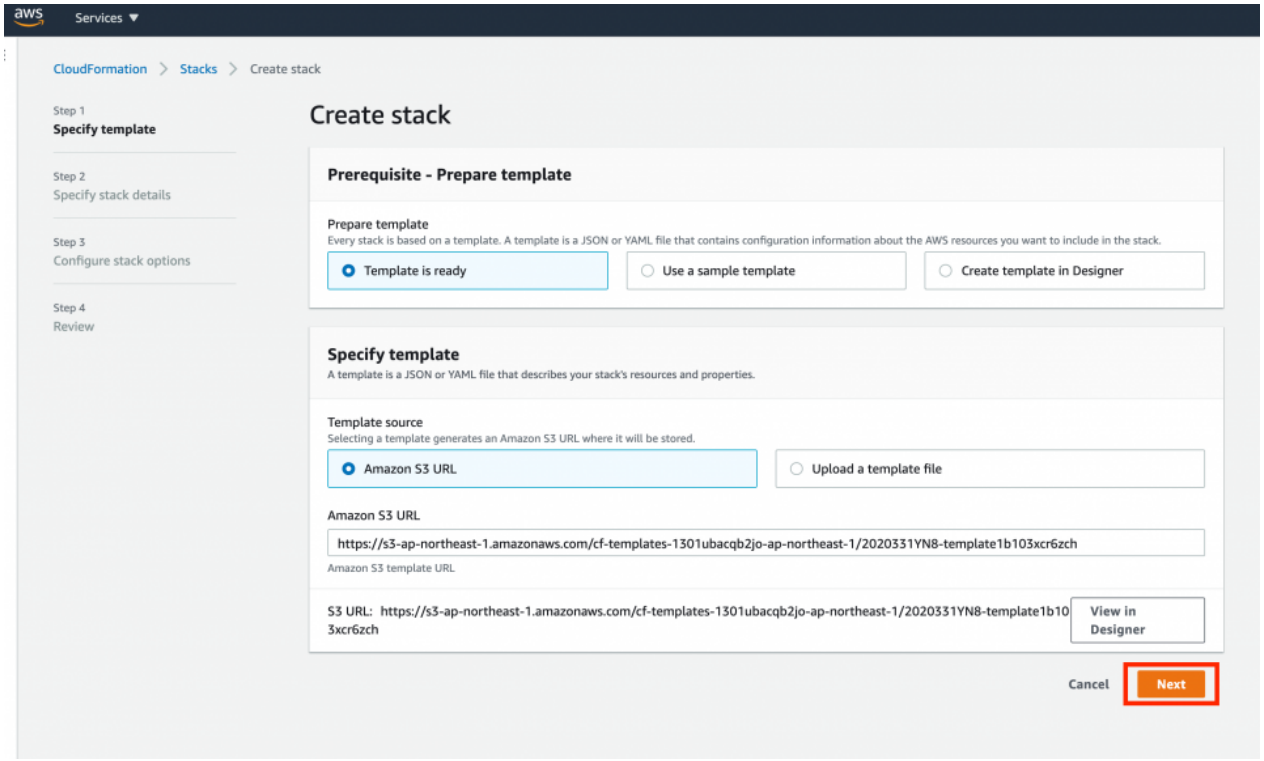
④ の赤枠の「Continue to Launch」を押します。

2.2.2.7 Step.5



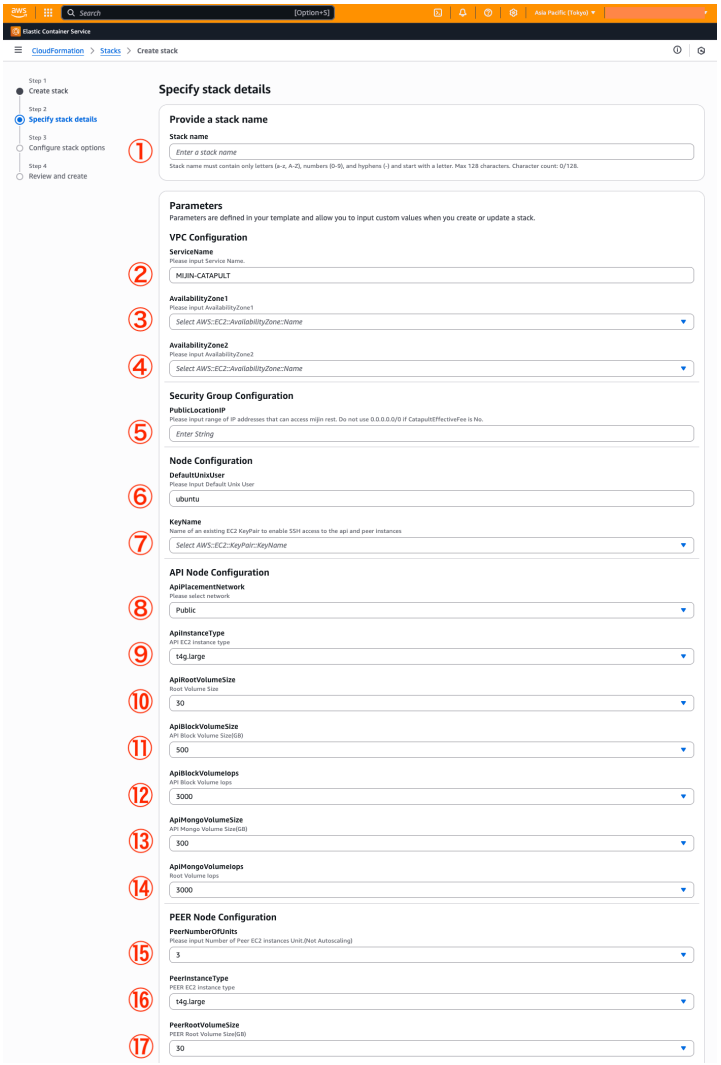
日: ① の赤枠にて、mijin の構築するサービスを指定します。ここでは「Launch CloudFormation」を指定します。日: ② の赤枠の、「Launch」を押します

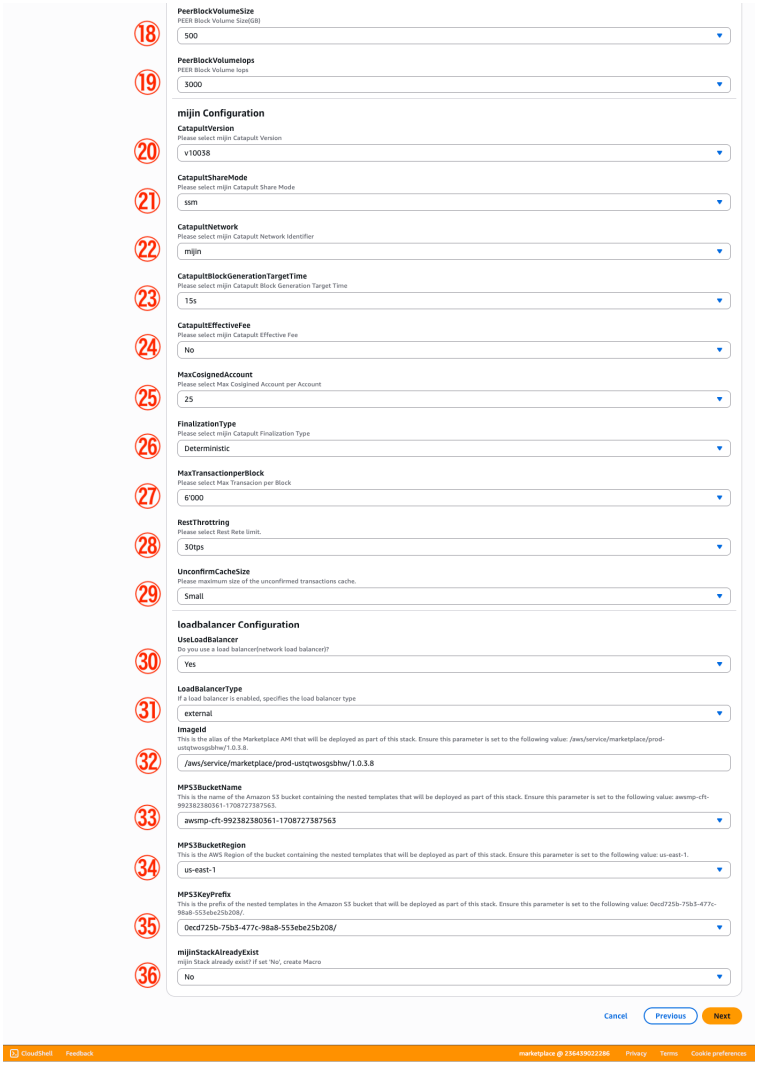
2.2.2.8 Step.6



日: とくに編集せず、赤枠の「Next」を押します。

2.2.2.9 Step.7


Search
[Option+S]
Asia Pacific (Tokyo)
Elastic Container Service
CloudFormation
Stacks
Create stack
Step 1
Create stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review and create
Specify stack details
Provide a stack name
Stack name
Enter a stack name
Stack name must contain only letters (a-z, A-Z), numbers (0-9), and hyphens (-) and start with a letter. Max 128 characters. Character count: 0/128.
Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.
VPC Configuration
ServiceName
Please input Service Name.
MIJIN-CATAPULT
AvailabilityZone1
Please input AvailabilityZone1
Select AWS::EC2::AvailabilityZone::Name
AvailabilityZone2
Please input AvailabilityZone2
Select AWS::EC2::AvailabilityZone::Name
Security Group Configuration
PublicLocationIP
Please input range of IP addresses that can access mijin rest. Do not use 0.0.0.0/0 if CatapultEffectiveFee is No.
Enter String
Node Configuration
DefaultUnixUser
Please Input Default Unix User
ubuntu
KeyName
Name of an existing EC2 KeyPair to enable SSH access to the api and peer instances
Select AWS::EC2::KeyPair::KeyName
API Node Configuration
ApiPlacementNetwork
Please select network
Public
ApiInstanceType
API EC2 instance type
t4g.large
ApiRootVolumeSize
Root Volume Size
30
ApiBlockVolumeSize
API Block Volume Size(GB)
500
ApiBlockVolumeIops
API Block Volume Iops
3000
ApiMongoVolumeSize
API Mongo Volume Size(GB)
300
ApiMongoVolumeIops
Root Volume Iops
3000
PEER Node Configuration
PeerNumberOfUnits
Please input Number of Peer EC2 instances Unit.(Not Autoscaling)
3
PeerInstanceType
PEER EC2 instance type
t4g.large
PeerRootVolumeSize
PEER Root Volume Size(GB)
30
①
②
③
④
⑤
⑥
⑦
⑧
⑨
⑩
⑪
⑫
⑬
⑭
⑮
⑯
⑰



日: パラメータを入力します。

表 5: mijin デプロイ用パラメーター一覧（簡易版）

No	パラメータ	説明	推奨値
①	Stack Name	このスタックの名前	•
②	Service Name	全リソースの冠名として利用されるサービス名	•
③	Availability Zone1	使用するアベイラビリティゾーン（Multi-AZ 構成）	•
④	Availability Zone2	AZ1 と異なる AZ を指定（Multi-AZ 構成）	•
⑤	Public Location IP	API 接続許可 IP アドレス（/24 等も可）	例: XX.XX.XX.XX/32
⑥	Default UnixUser	EC2 の標準 Unix ユーザー	ubuntu
⑦	KeyName	EC2 SSH 接続用鍵名	•
⑧	ApiPlacementNetwork	API ノードのネットワーク配置先	Public
⑨	ApiInstanceType	API ノードのインスタンスタイプ	c5n.2xlarge 以上
⑩	ApiRootVolumeSize	API ノードのルートディスク容量（Docker 等に使用）	30GB 以上
⑪	ApiBlockVolumeSize	mijin ブロックデータ格納用ディスク容量	500GB 以上
⑫	ApiBlockVolumeIops	上記ディスクの IOPS 設定	3000 以上
⑬	ApiMongoVolumeSize	mongo データ格納用ディスク容量	300GB 以上

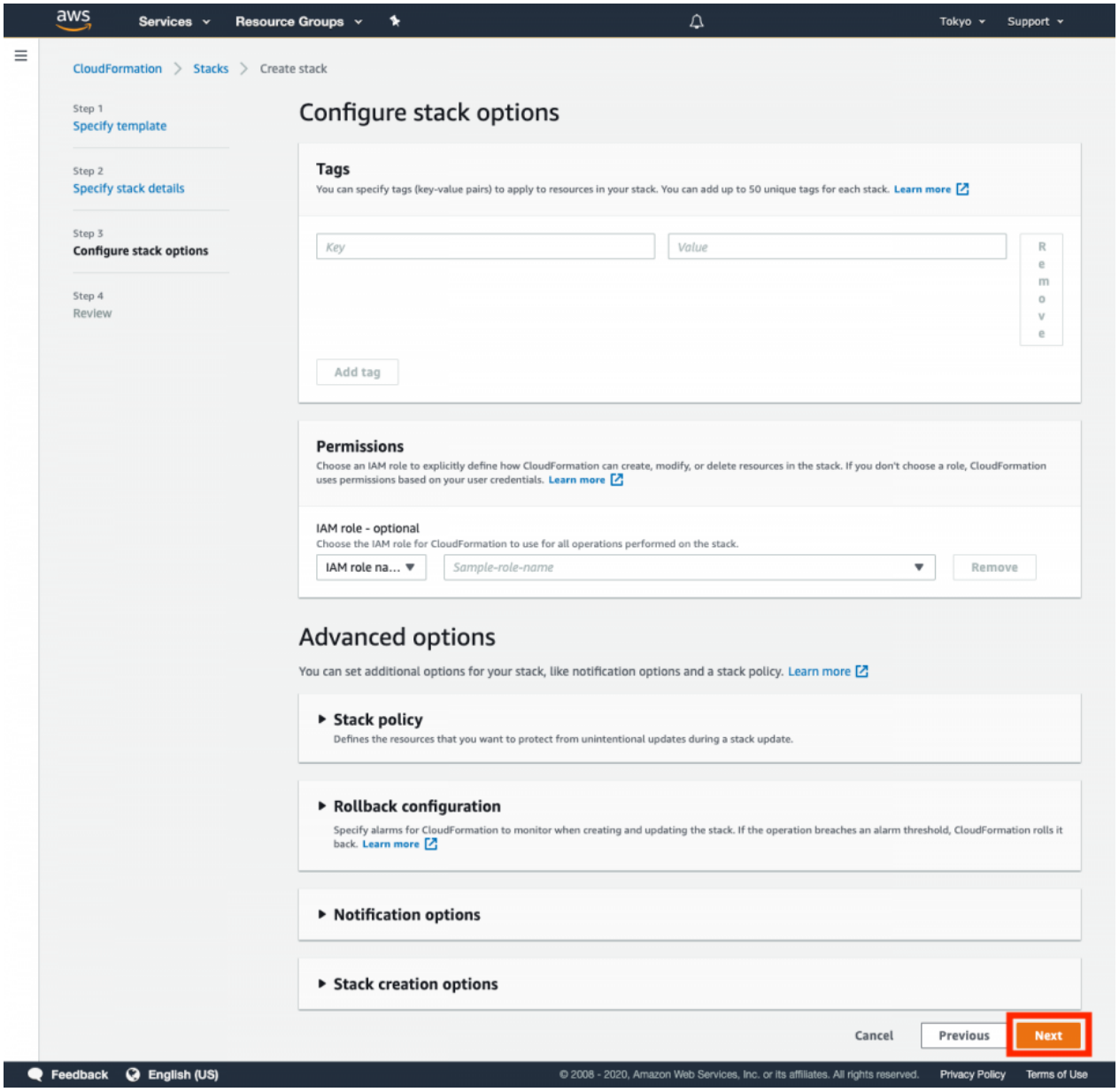
次のページに続く

表 5 – 前のページからの続き

No	パラメータ	説明	推奨値
⑭	ApiMongoVolumeIops	mongo ディスクの IOPS 設定	3000 以上
⑮	PeerNumberOfUnits	PEER ノードの台数	3
⑯	PeerInstanceType	PEER ノードのインスタンスタイプ	c5n.xlarge 以上
⑰	PeerRootVolumeSize	PEER ノードのルートディスク容量	30GB 以上
⑱	PeerBlockVolumeSize	PEER ノードのブロックデータ用ディスク	500GB 以上
⑲	PeerBlockVolumeIops	上記ディスクの IOPS 設定	3000 以上
⑳	CatapultVersion	mijin のバージョン	v10038
㉑	CatapultShareMode	設定保存方式（例：SSM）	ssm
㉒	CatapultNetwork	ネットワーク名	mijin
㉓	BlockGenerationTargetTime	ブロック生成間隔	15s
㉔	EffectiveFee	手数料設定（あり/なし）	No
㉕	MaxCosignedAccount	署名可能最大アカウント数	25
㉖	FinalizationType	ファイナライゼーション方式	Deterministic
㉗	MaxTransactionperBlock	最大トランザクション数（1 ブロック）	6000
㉘	RestThrottling	API 接続上限	30tps
㉙	UnconfirmCacheSize	未承認トランザクションキャッシュサイズ	•
㉚	UseLoadBalancer	NLB の使用有無	Yes
㉛	LoadBalancerType	ロードバランサー種別	External
㉜	ImageId	管理用 AMI の ID（変更不可）	変更不可
㉝	MPS3BucketName	S3 バケット名（変更不可）	変更不可
㉞	MPS3BucketRegion	S3 リージョン（変更不可）	変更不可
㉟	MPS3KeyPrefix	S3 のプレフィックス（変更不可）	変更不可
㊱	mijinStackAlreadyExist	既存スタックの有無	No

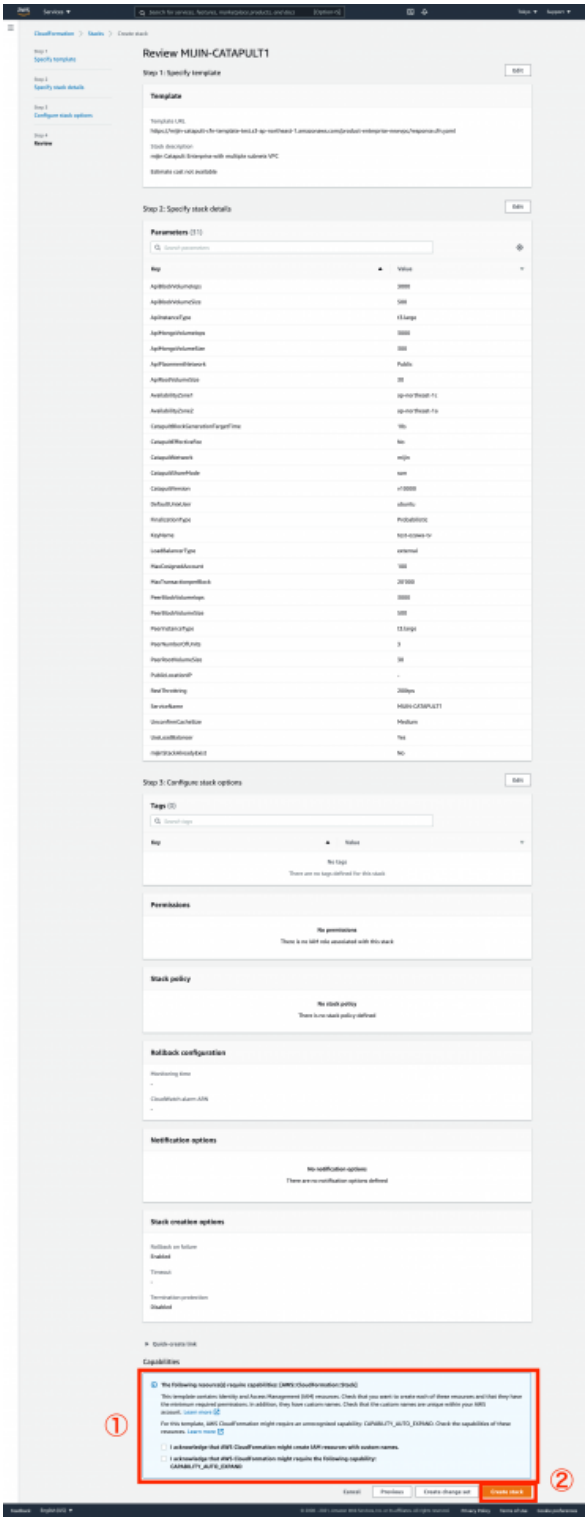
日: パラメータの入力完了後、「Next」を押します。

2.2.2.10 Step.8



日: とくに編集せず、赤枠の「Next」を押します。

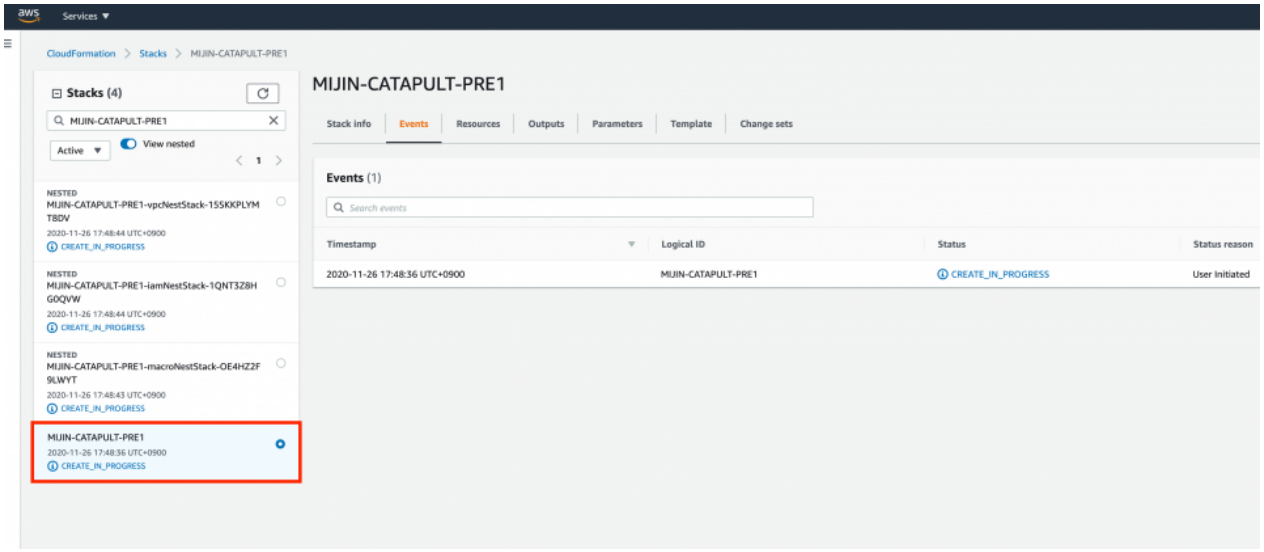
2.2.2.11 Step.9



日: ① の赤枠にて２つの項目にチェックを入れます。

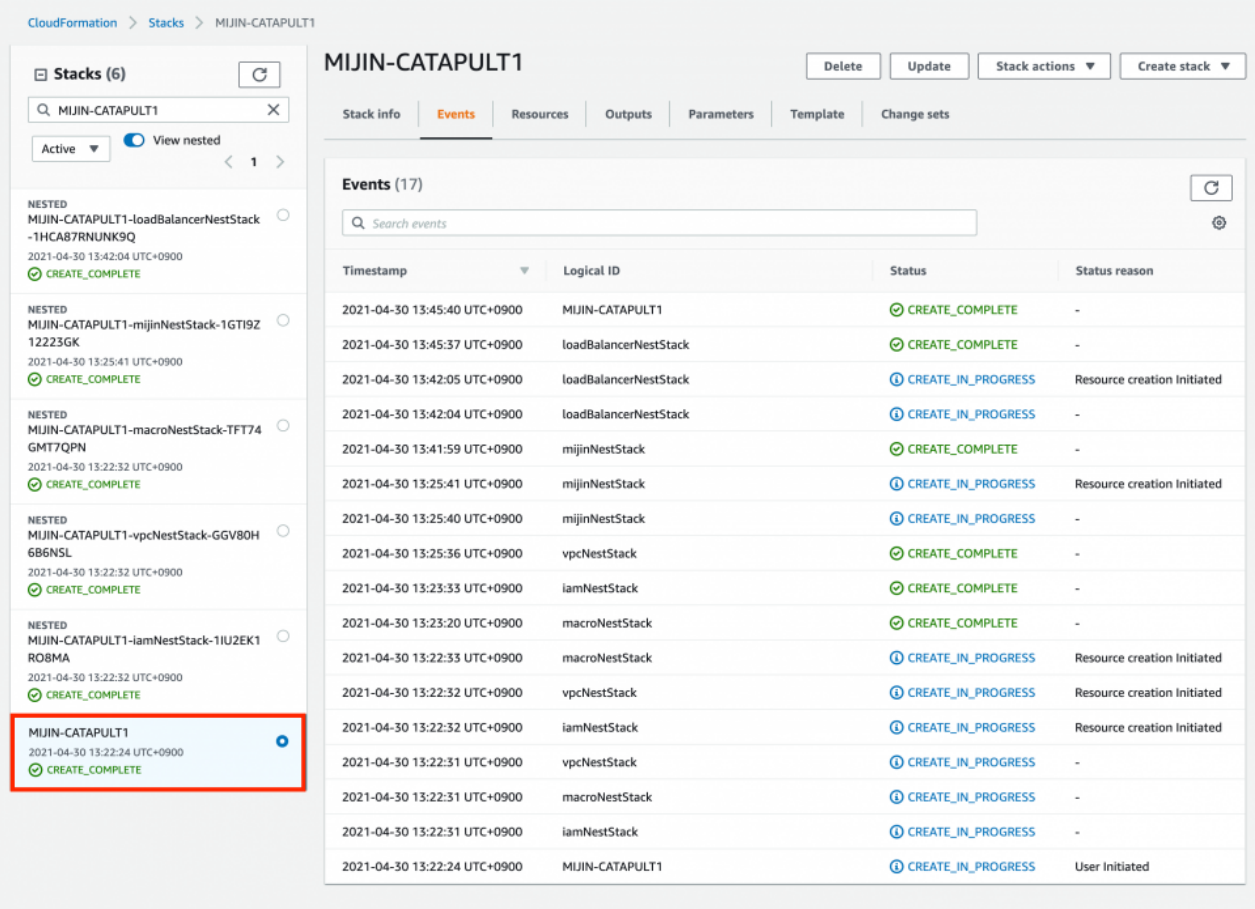
日: ② の赤枠の「Create Stack」を押します。エラーがなければ作成が始まります。

2.2.2.12 Step.10



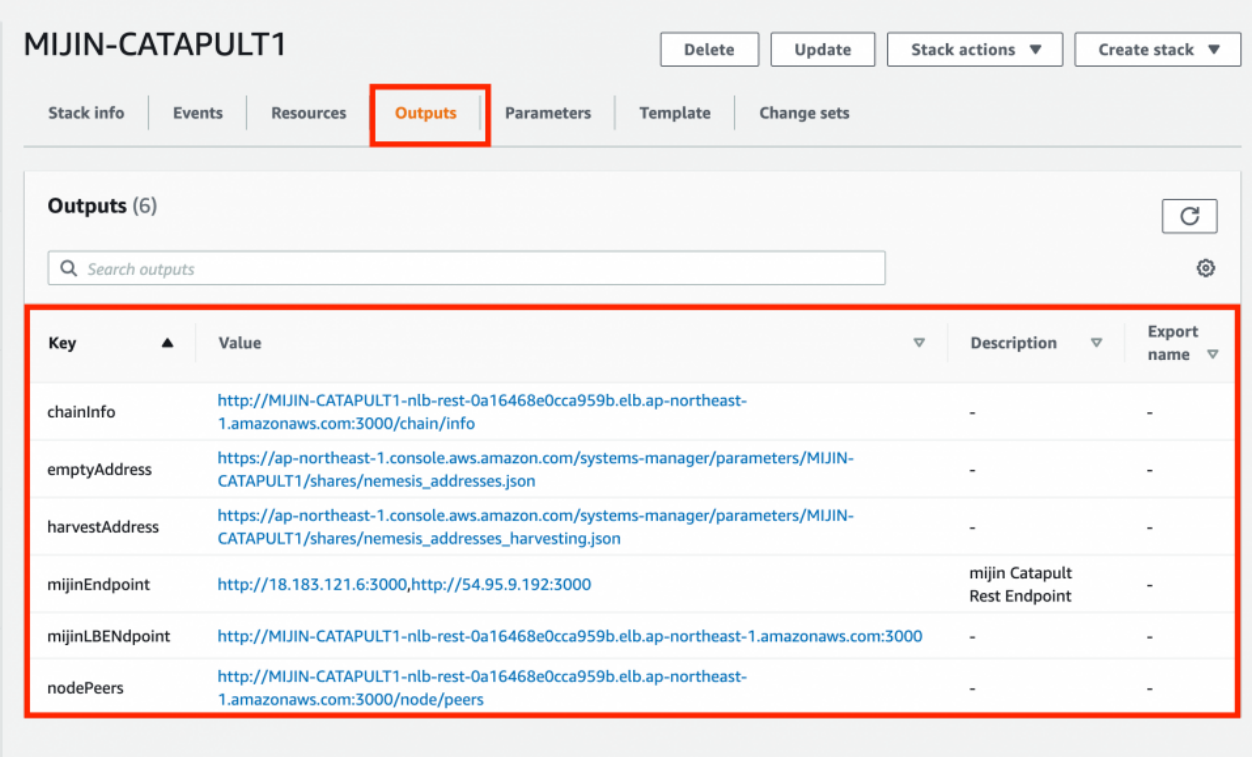
日: Stack が始まり「CREATE_IN_PROGRESS」になっていることを確認してください。この状態はおよそ 20~30 分程度かかります。

2.2.2.13 Step.11



日: 「CREATE_COMPLATE」の状態であれば、mijin の作成が完了しました。

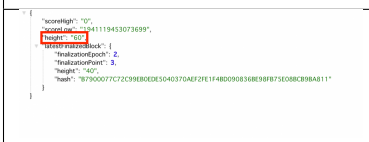
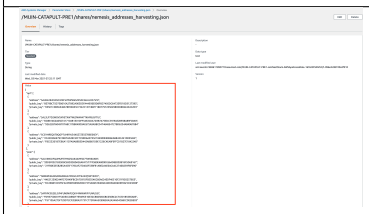
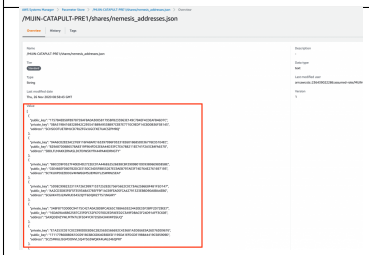
2.2.2.14 Step.12



日: 作成した Stack の「Outputs」を押すと、作成された mijin の設定情報を確認できます。

2.2.2.15 mijin エンドポイントと確認項目

表 6: mijin エンドポイントと確認項目

	mijinLBEndpoint ロードバランサー経由の mijin API エンドポイントです。セッション維持設定が有効で、ソース IP に基づいたスティッキーセッションとなります。詳細はこちら
	mijinEndpoint API ノード (EC2) の直アクセス用の mijin API エンドポイントです。
	**chainInfo** mijin の現在のブロック数を確認できます。ブロック数が「2」以上であれば正常です。
	**harvestAddress** AWS Systems Manager パラメータストアに登録された通貨分配用アドレスのリンクです。
	**emptyAddress** AWS Systems Manager パラメータストアに登録された未使用アドレスのリンクです。
	**nodePeers** mijin API からノードの接続状態が確認できます。API ノード 1 台と、設定済みの PEER 台数が表示されていれば OK です。

日: これで mijin Catapult を使用する準備が整いました。それでは次の項で操作を始めてみましょう!

2.2.3 既存 VPC 上に、mijin をデプロイする

本章では、既存ネットワーク (VPC) に mijin Catapult(v.2) をデプロイする方法を示します。
製品版では、パラメーターを変更することで環境にあったネットワークを柔軟に構築することが可能です。

2.2.3.1 デプロイによって AWS 上に構築するサービス一覧

- Amazon EC2 (API ノード x 2 PEER ノード x 3~)
- Amazon EBS
- Elastic Load Balancing
- Amazon Route53
- パラメータストア

注釈:

Elastic Load Balancing は、デプロイ時に指定するパラメーターの設定により、作成しないケースがあります。

2.2.3.2 既存 VPC のサブネットの作成

mijin をデプロイするうえで、高可用性を実現するため、Public ネットワークと Private ネットワークのサブネットが二つずつ必要になります。
サブネット数が足りない場合、以下を参考に作成してください。

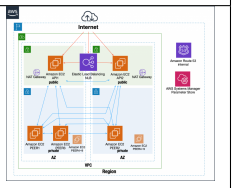
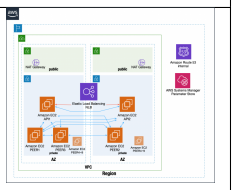
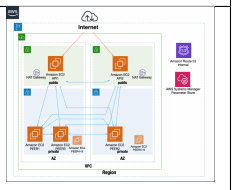
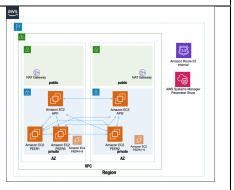
<https://docs.aws.amazon.com/ja_jp/vpc/latest/userguide/working-with-subnets.html#create-subnets>

注釈: 片方の AZ に障害が発生してもサービスを継続できるように、複数のアベイラビリティゾーン (AZ) のサブネット二つを作成してください。

2.2.3.3 View Network

既存ネットワーク用では、パラメーターを変更することですでに存在する環境へ構築することが可能です。
以下にパターン例を示します。

表 7: API/PEER ネットワーク構成一覧

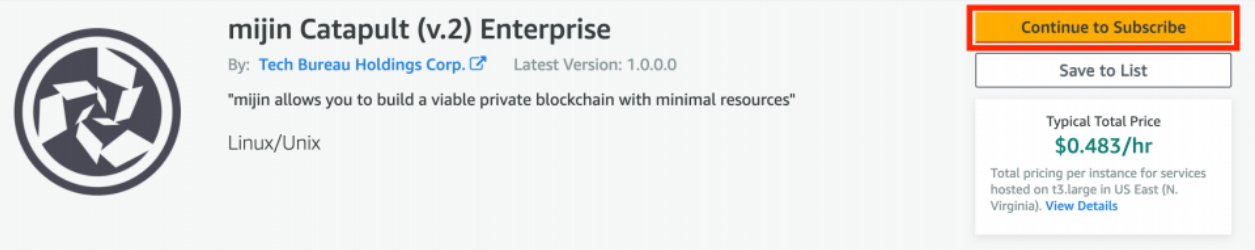
No	ロードバランサー	API ノード配置	PEER ノード配置	図
1	**あり（公開ネットワーク）** VPC: Your vpc VpcCidrBlock: xx.xx.xx.xx/xx Public1/2, Private1/2 UseLoadBalancer: `Yes` LoadBalancerType: `external`	公開 NW ApiPlacementNetwork: `Public`	非公開 NW	
2	**あり（非公開ネットワーク）** UseLoadBalancer: `Yes` LoadBalancerType: `internal`	非公開 NW ApiPlacementNetwork: `Private`	非公開 NW	
3	**なし** UseLoadBalancer: `No` LoadBalancerType は設定効果なし	公開 NW ApiPlacementNetwork: `Public`	非公開 NW	
4	**なし** UseLoadBalancer: `No` LoadBalancerType は設定効果なし	非公開 NW ApiPlacementNetwork: `Private`	非公開 NW	

注釈:

製品版では、高可用性を実現するため、シングル AZ で構成はできず、マルチ AZ のみの構成となっています。

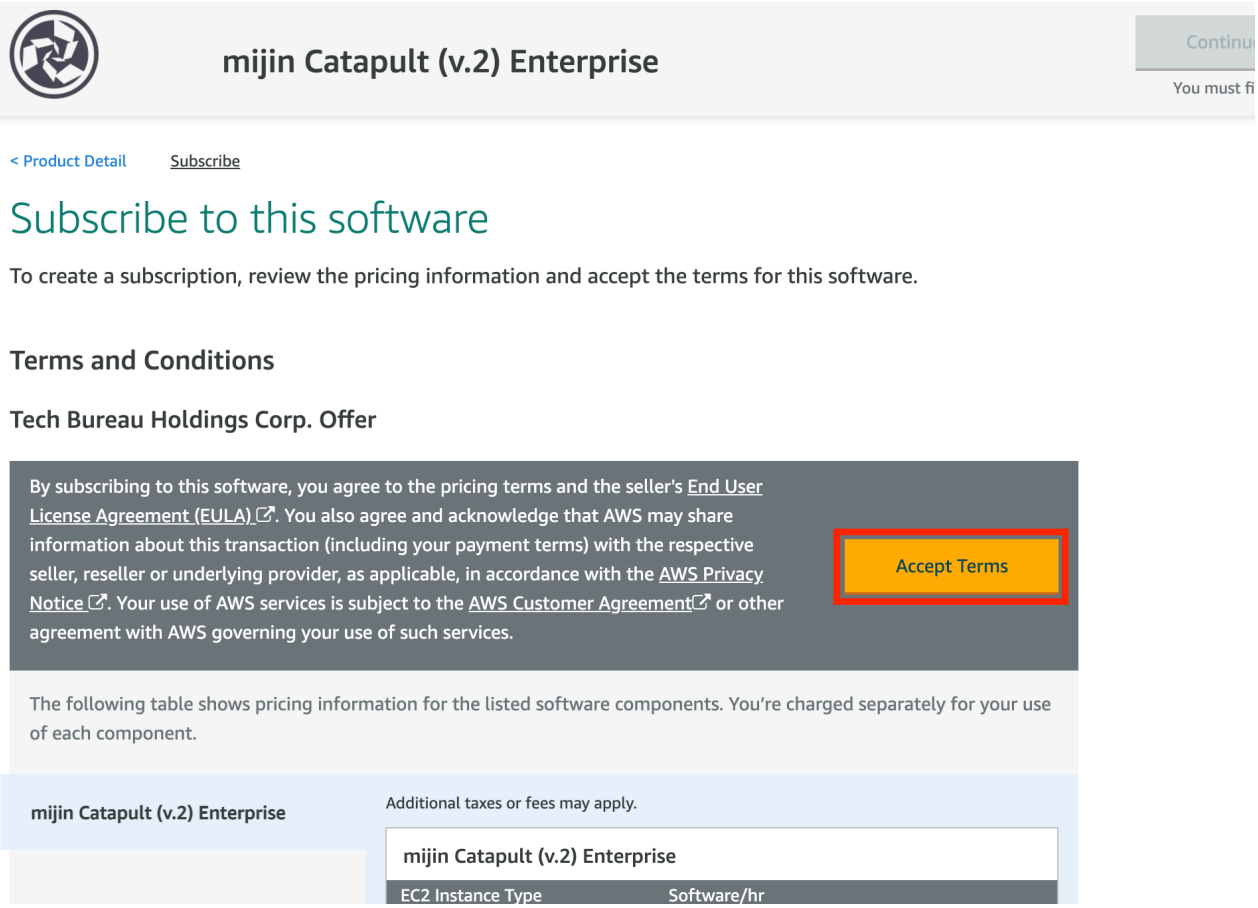
必ずパブリックネットワーク 2 つ、プライベートネットワーク 2 つが必要になることに留意してください。
配置はシングルリージョンとなりますが、世界 21 リージョン毎にデプロイすることが可能です。

2.2.3.4 Step.1



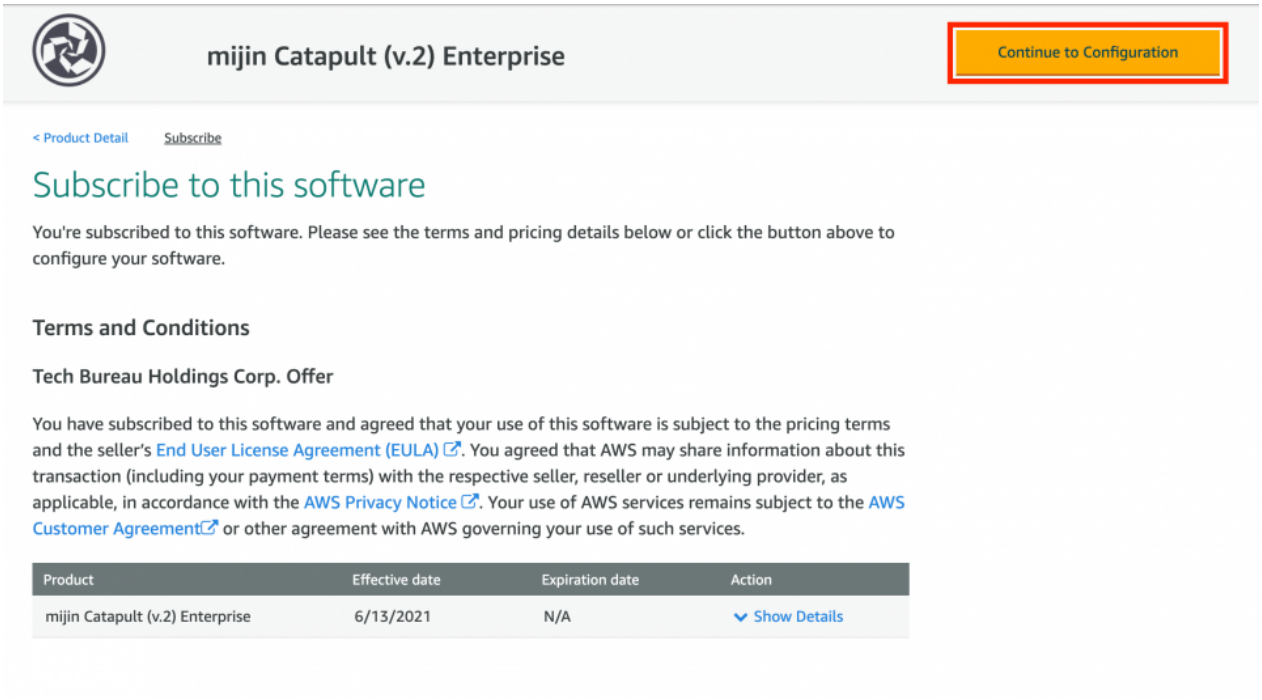
日: mijin Catapult Enterprise の AMI を使用するためにサブスクライブする必要があります。赤枠のボタンを押してください。

2.2.3.5 Step.2



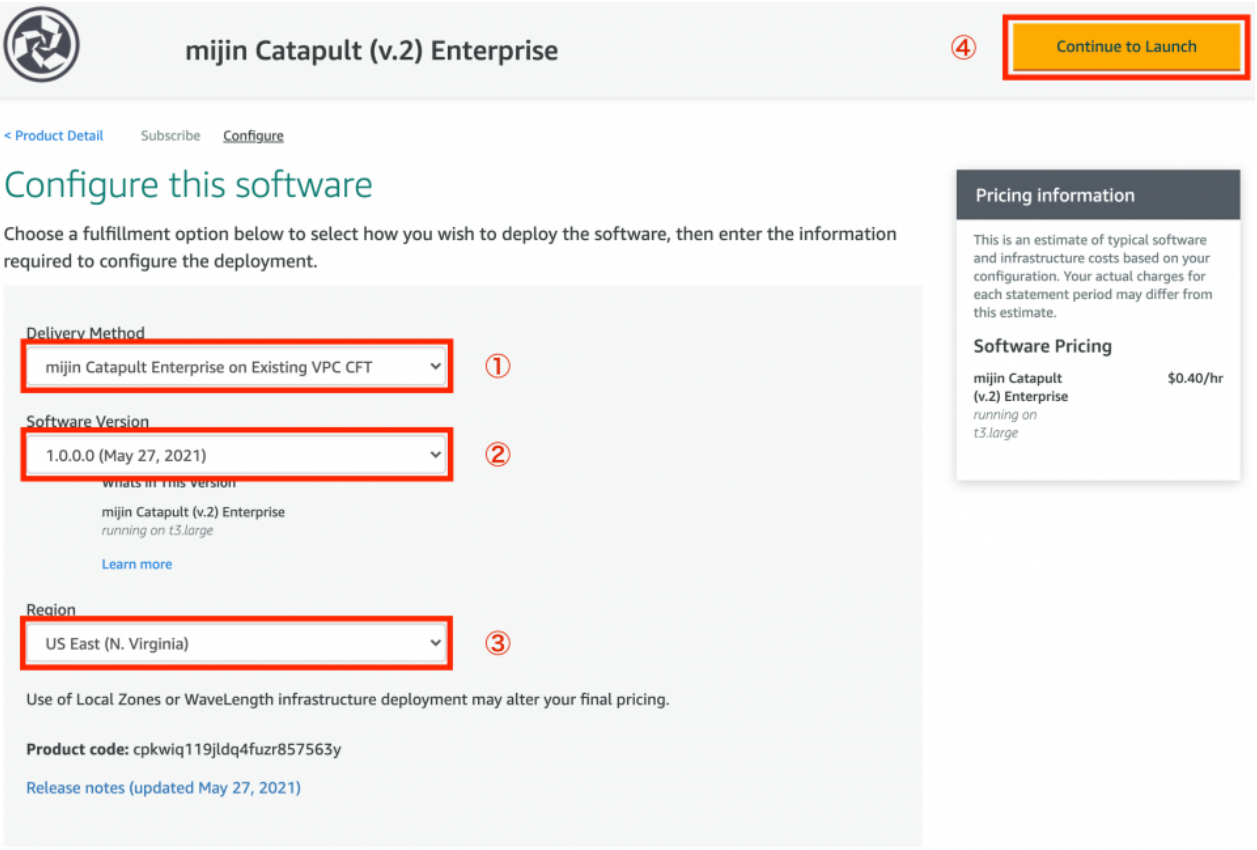
日: mijin Catapult Enterprise AMI を使用するため、使用の承認をしてください。

2.2.3.6 Step.3



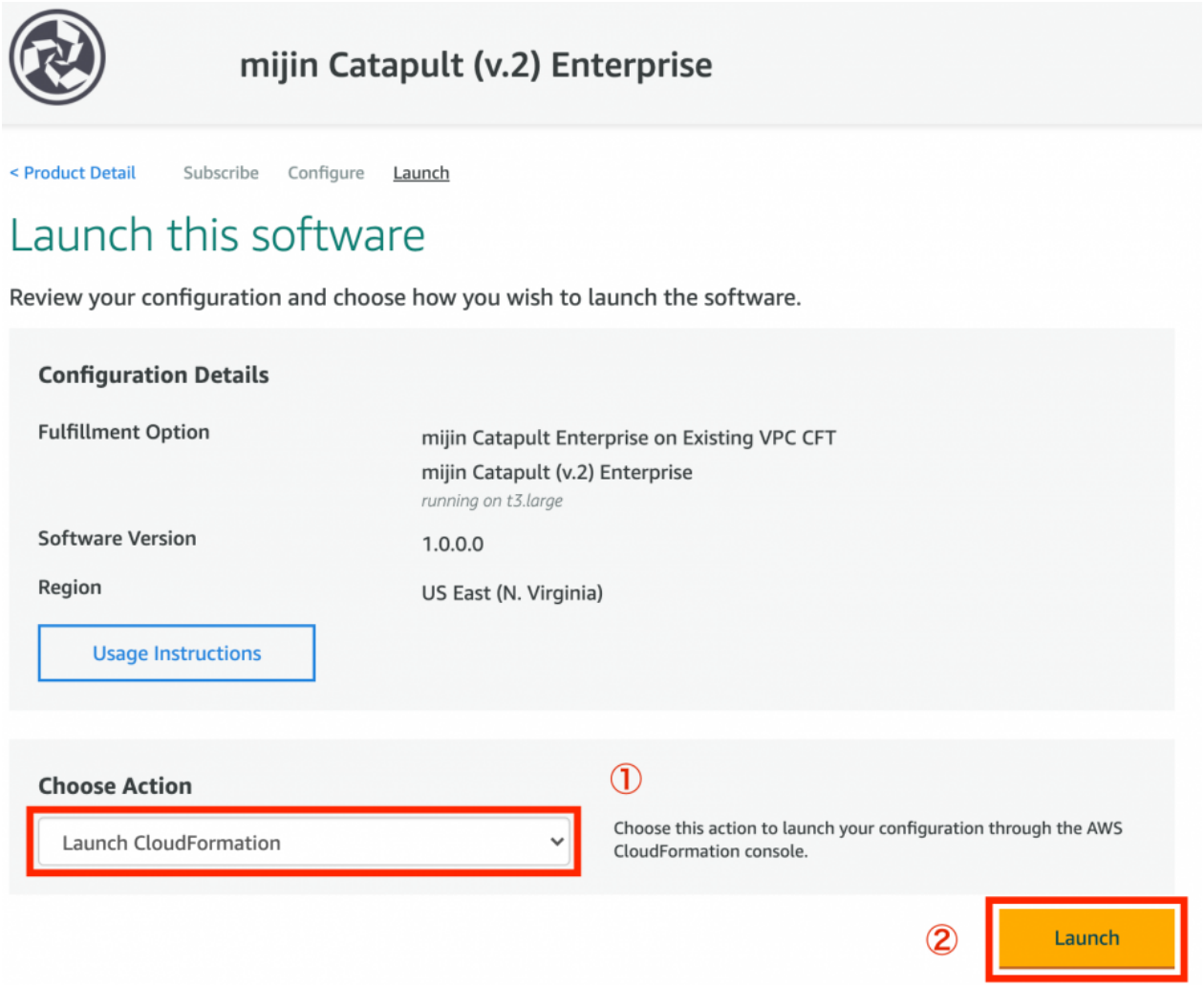
日: Config から起動するために、赤枠の文字をクリックしてください。

2.2.3.7 Step.4



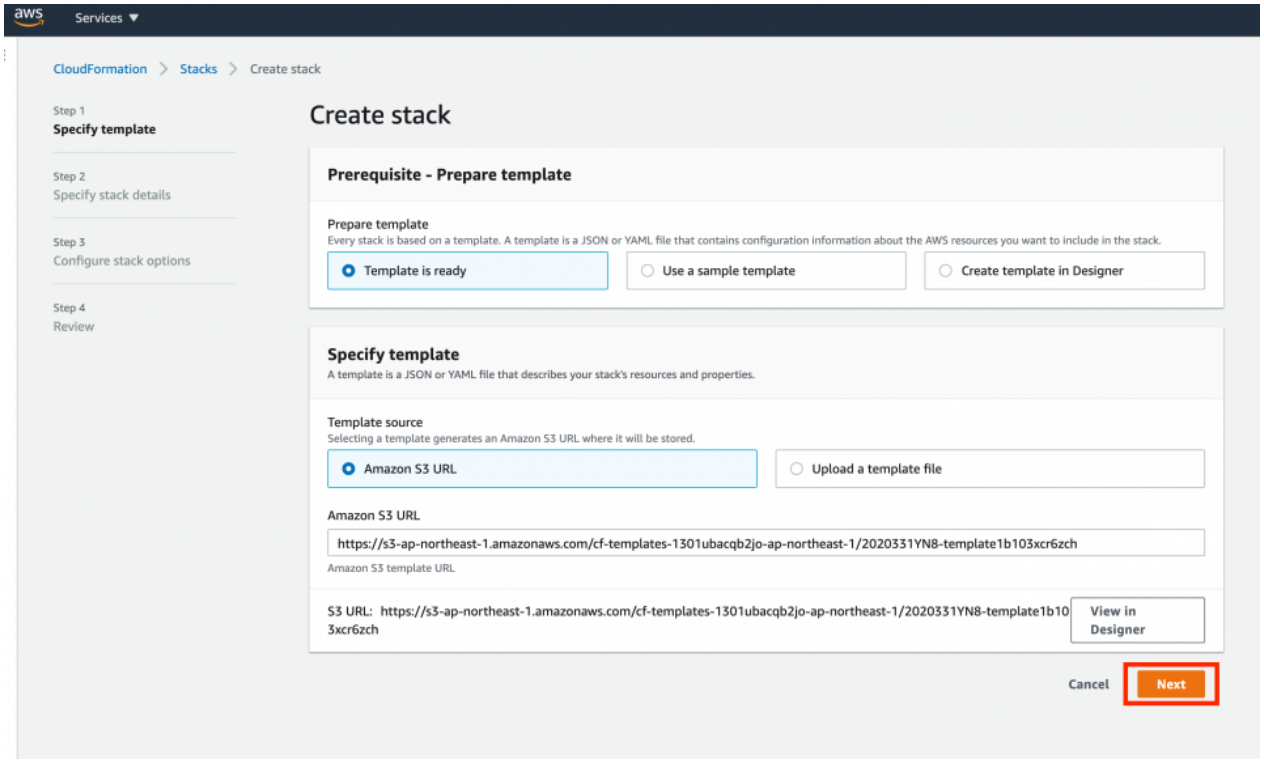
日: ① の赤枠にて使用するテンプレートを選択します。ここでは既存ネットワークに作成するため「mijin Catapult Enterprise on Existing VPC CFT」を指定します。日: ② の赤枠ににて mijin のバージョンを指定します。日: ③ mijin を展開するリージョンを指定します。日: ④ の赤枠の「Continue to Launch」を押します。

2.2.3.8 Step.5



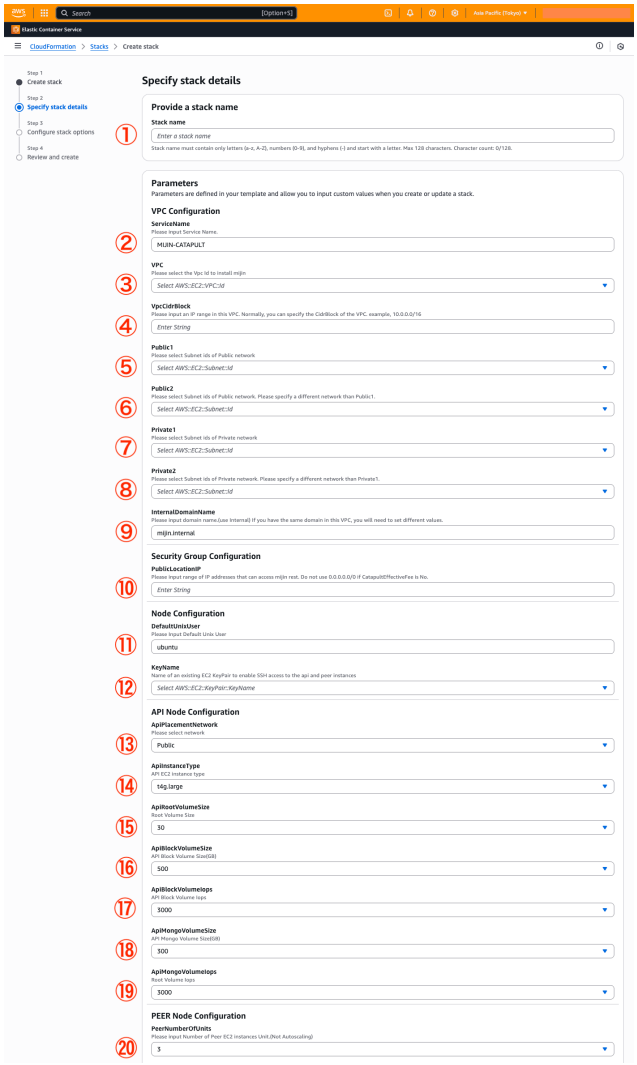
日: ① の赤枠にて、mijin の構築するサービスを指定します。ここでは「Launch CloudFormation」を指定します。日: ② の赤枠の、「Launch」を押します

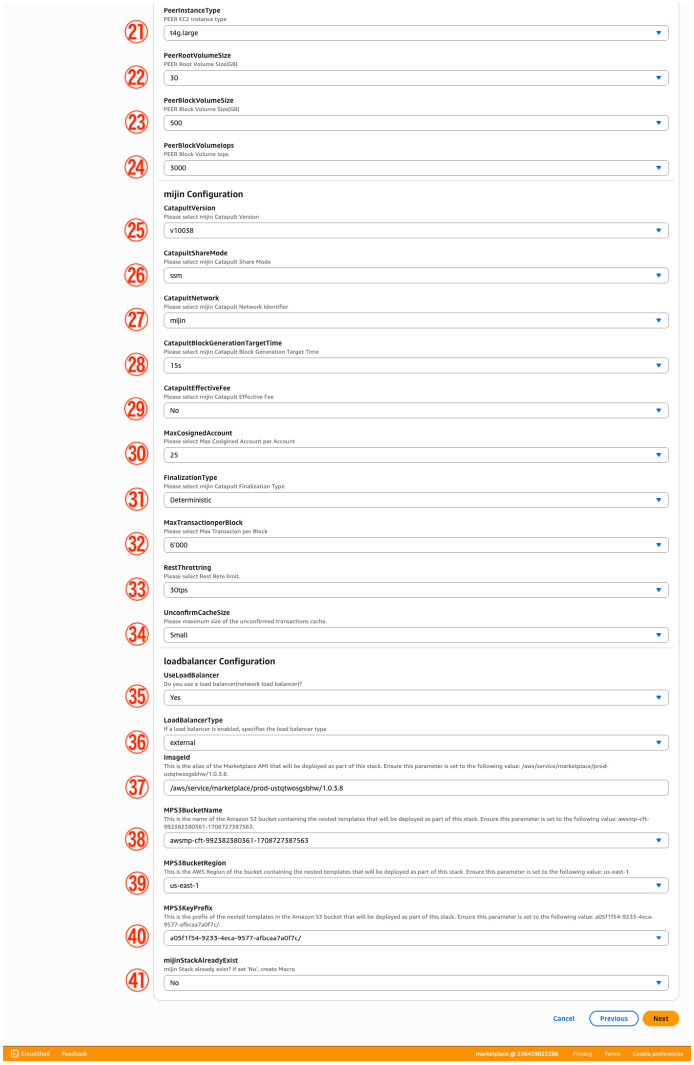
2.2.3.9 Step.6



日: とくに編集せず、赤枠の「Next」を押します。

2.2.3.10 Step.7


Elastic Container Service
CloudFormation > Stacks > Create stack
Step 1
Create stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review and create
Specify stack details
Provide a stack name
Stack name
Enter a stack name
Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.
VPC Configuration
ServiceName
Please input Service Name.
MIJIN-CATAPULT
VPC
Please select the Vpc Id to install mijin
Select AWS::EC2::VPC::Id
VpcCidrBlock
Please input an IP range in this VPC. Normally, you can specify the CidrBlock of the VPC. example, 10.0.0.0/16
Enter String
Public1
Please select Subnet ids of Public network
Select AWS::EC2::Subnet::Id
Public2
Please select Subnet ids of Public network. Please specify a different network than Public1.
Select AWS::EC2::Subnet::Id
Private1
Please select Subnet ids of Private network
Select AWS::EC2::Subnet::Id
Private2
Please select Subnet ids of Private network. Please specify a different network than Private1.
Select AWS::EC2::Subnet::Id
InternalDomainName
Please input domain name.(use Internal) If you have the same domain in this VPC, you will need to set different values.
mijin.internal
Security Group Configuration
PublicLocationIP
Please input range of IP addresses that can access mijin rest. Do not use 0.0.0.0/0 if CatapultEffectiveFee is No.
Enter String
Node Configuration
DefaultUnixUser
Please input Default Unix User
ubuntu
KeyName
Name of an existing EC2 KeyPair to enable SSH access to the api and peer instances
Select AWS::EC2::KeyPair::KeyName
API Node Configuration
ApiPlacementNetwork
Please select network
Public
ApiInstanceType
API EC2 instance type
t4g.large
ApiRootVolumeSize
Root Volume Size
30
ApiBlockVolumeSize
API Block Volume Size(GB)
500
ApiBlockVolumeIops
API Block Volume Iops
3000
ApiMongoVolumeSize
API Mongo Volume Size(GB)
300
ApiMongoVolumeIops
Root Volume Iops
3000
PEER Node Configuration
PeerNumberOfUnits
Please input Number of Peer EC2 instances Unit.(Not Autoscaling)
3
①
②
③
④
⑤
⑥
⑦
⑧
⑨
⑩
⑪
⑫
⑬
⑭
⑮
⑯
⑰
⑱
⑲
⑳



日: パラメータを入力します。

表 8: CloudFormation パラメーター一覧

No	Parameter	Describe	推奨値
①	Stack Name	このスタックにおける名前を記載してください。	•
②	Service Name	スタックによって作成されるサービス名を記載してください。 全リソースの冠名として使用されます。	•
③	VPC	お使いの VPC を指定してください。	•
④	VpcCidrBlock	③ で指定した VPC の IP アドレスの範囲を記載してください。(CIDR Block) ③ で選択した () 内の IP アドレス範囲で問題ありません。	•
⑤	Public1	③ で指定した VPC にある公開ネットワークを指定してください。	•
⑥	Public2	③ で指定した VPC にある公開ネットワークを指定してください。 ⑤ と別のネットワークを指定する必要があります。	•

次のページに続く

表 8 – 前のページからの続き

No	Parameter	Describe	推奨値
⑦	Private1	③ で指定した VPC にある非公開ネットワークを指定してください。	•
⑧	Private2	③ で指定した VPC にある非公開ネットワークを指定してください。 ⑦ とは別のネットワークを指定する必要があります。	•
⑨	InternalDomainName	ノード間で使用する名前解決用の DNS 名を記載してください。 非公開ネットワーク用で、公開はされません。 複数スタック構築時には一意にしてください。	mijin.internal
⑩	Public Location IP	mijin Catapult の API への接続許可 IP アドレスを指定してください。 IP レンジ指定可 (/24 など)	XX.XX.XX.XX/32 など
⑪	Default UnixUser	作成する EC2 の標準 Unix ユーザー	ubuntu
⑫	KeyName	EC2 の SSH 鍵を指定してください。 表示されない場合は事前作成が必要です。 こちら	•
⑬	ApiPlacementNetwork	API ノードの配置ネットワークを選択してください。	•
⑭	ApiInstanceType	API ノードのインスタンスタイプ アーキテクチャとスペック要件 参照	c5n.2xlarge 以上
⑮	ApiRootVolumeSize	API ノードのルートディスクサイズ (GB) Docker ログや System log に使用	30GB 以上
⑯	ApiBlockVolumeSize	mijin ブロックデータ格納用ディスクサイズ (GB) gp3 ディスクを利用	500GB 以上
⑰	ApiBlockVolumeIops	mijin ブロックデータ用 IOPS	3000 以上
⑱	ApiMongoVolumeSize	mongo データ格納用ディスクサイズ (GB) Blockchain データ呼び出し用	300GB 以上
⑲	ApiMongoVolumeIops	mongo データ用 IOPS	3000IOPS 以上
⑳	PeerNumberOfUnits	PEER ノードの台数	3 以上
㉑	PeerInstanceType	PEER ノードのインスタンスタイプ アーキテクチャとスペック要件 参照	c5n.xlarge 以上
㉒	PeerRootVolumeSize	PEER ノードのルートディスクサイズ (GB) Docker ログや System log に使用	30GB 以上
㉓	PeerBlockVolumeSize	PEER ノードのブロックデータ保存用ディスクサイズ	500GB 以上
㉔	PeerBlockVolumeIops	PEER ノードのブロックデータ用 IOPS	3000IOPS 以上
㉕	CatapultVersion	mijin のバージョン	v10038
㉖	CatapultShareMode	ブロック生成情報保存先を指定 現在は AWS Systems Manager に固定	ssm
㉗	CatapultNetwork	mijin ネットワーク指定	mijin
㉘	Catapult BlockGenerationTargetTime	ブロック生成間隔	15s
㉙	CatapultEffectiveFee	手数料有無の設定	No

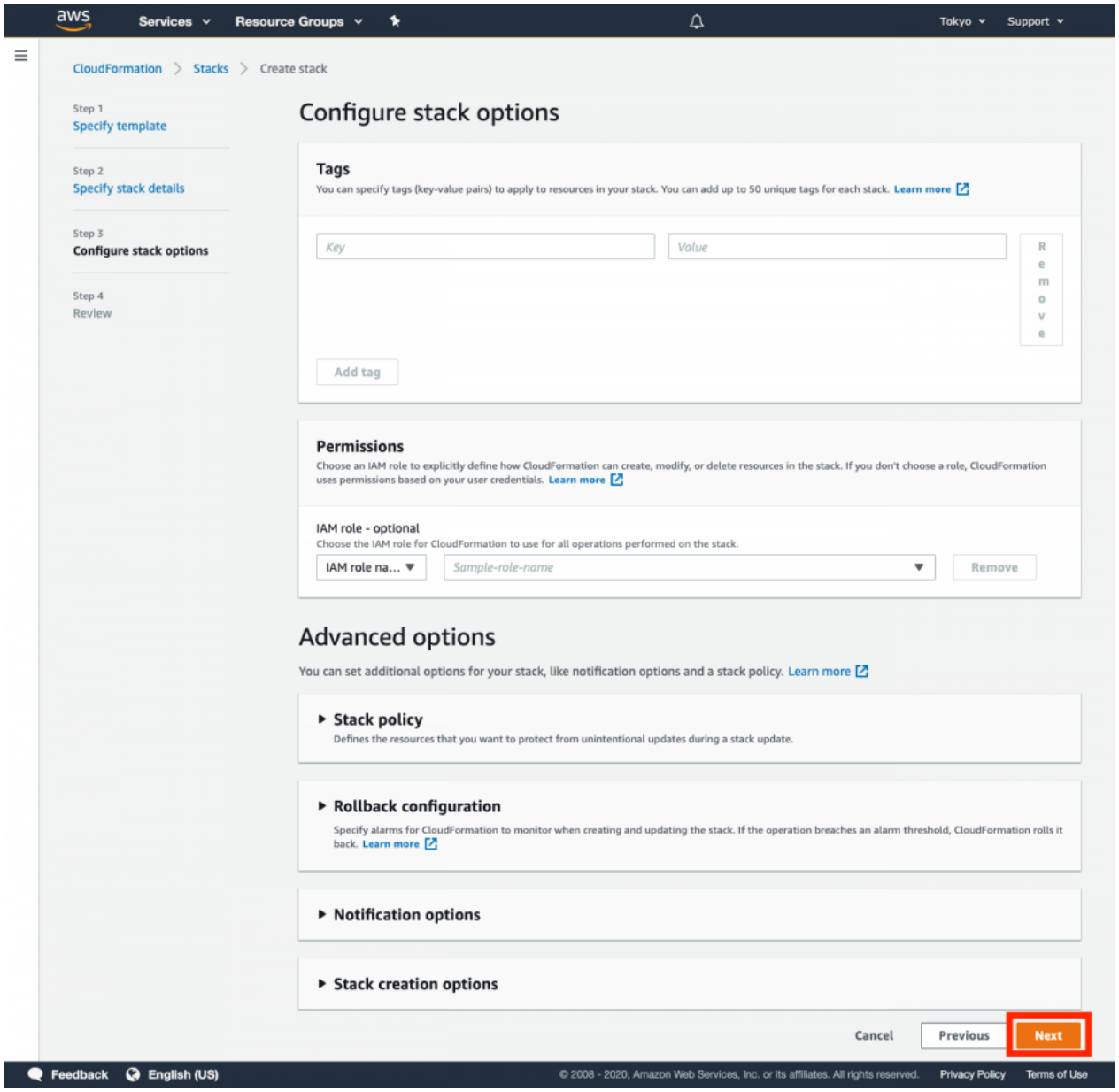
次のページに続く

表 8 – 前のページからの続き

No	Parameter	Describe	推奨値
㉚	MaxCosignedAccount	最大署名数	25
㉛	FinalizationType	ファイナライゼーション方式	Deterministic
㉜	MaxTransactionperBlock	1 ブロックの最大トランザクション数	6000
㉝	RestThrottring	API 接続数上限	30tps
㉞	UnconfirmCacheSize	未承認トランザクション格納数	Small
㉟	UseLoadBalancer	NLB 使用有無（プレビューは Yes 固定）	Yes
㊱	LoadBalancerType	ロードバランサー配置タイプ	External
㊲	ImageId	AMI ID（変更不可）	変更不可
㊳	MPS3BucketName	S3 バケット名（変更不可）	変更不可
㊴	MPS3BucketRegion	S3 リージョン（変更不可）	変更不可
㊵	MPS3KeyPrefix	S3 キー Prefix（変更不可）	変更不可
㊶	mijinStackAlreadyExist	他に mijin スタックがある場合、再作成時は「Yes」を選択	No

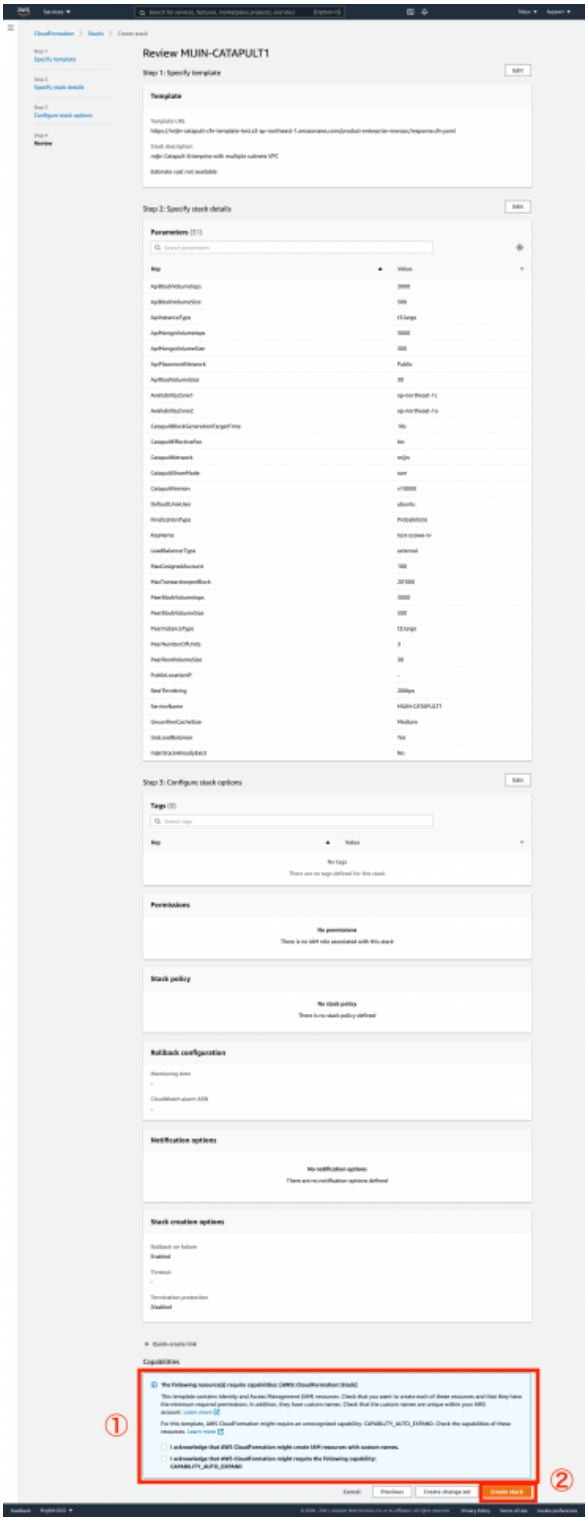
日: パラメータの入力完了後、「Next」を押します。

2.2.3.11 Step.8



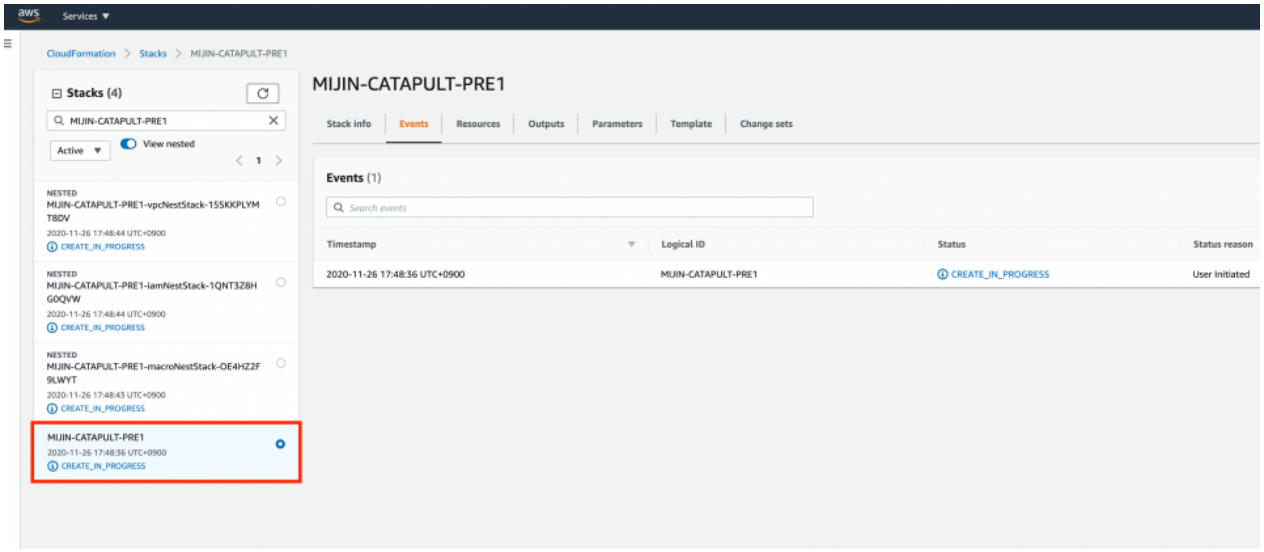
日: とくに編集せず、赤枠の「Next」を押します。

2.2.3.12 Step.9



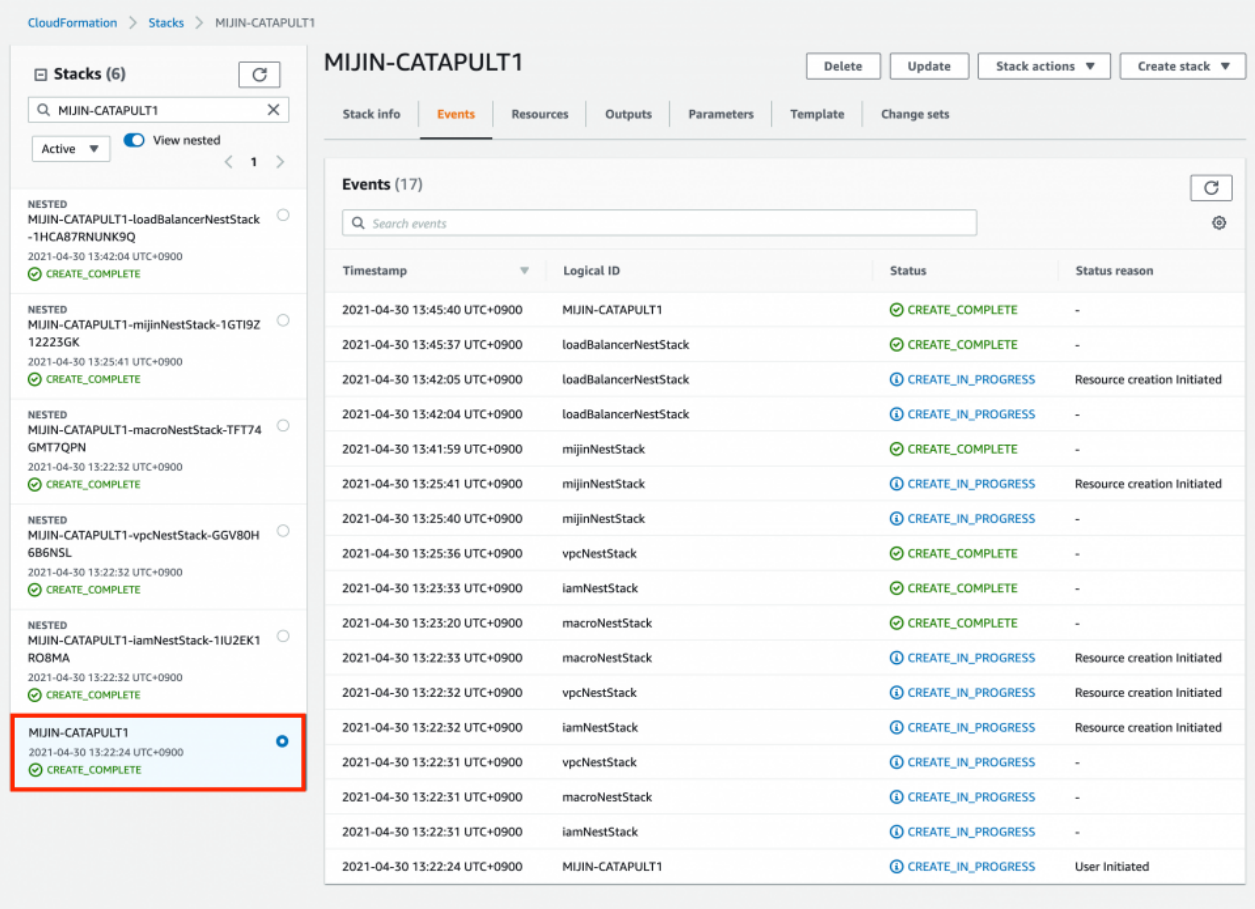
日: ① の赤枠にて２つの項目にチェックを入れます。日: ② の赤枠の「Create Stack」を押します。エラーがなければ作成が始まります。

2.2.3.13 Step.10



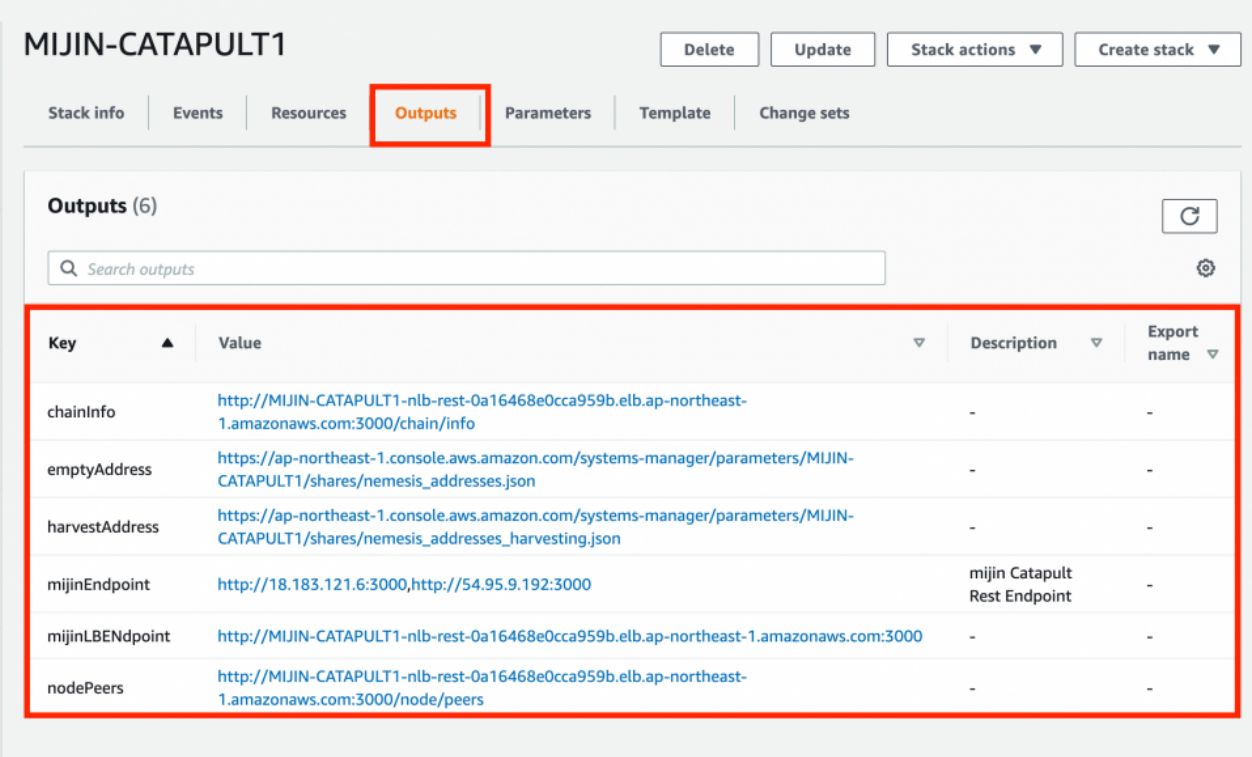
日: Stack が始まり「CREATE_IN_PROGRESS」になっていることを確認してください。この状態はおよそ 20~30 分程度かかります。

2.2.3.14 Step.11



日: 「CREATE_COMPLATE」の状態であれば、mijin の作成が完了しました。

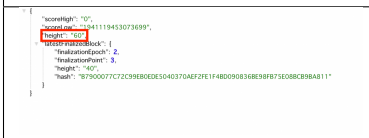
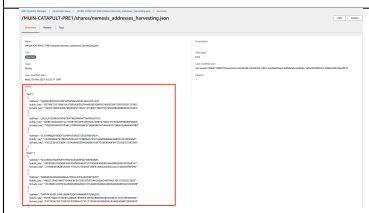
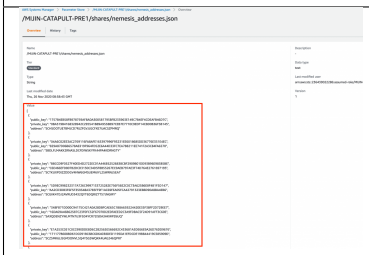
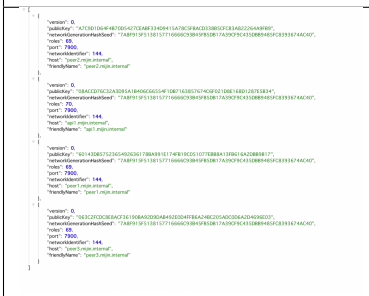
2.2.3.15 Step.12



日: 作成した Stack の「Outputs」を押すと、作成された mijin の設定情報を確認できます。

※ 以下表は、新規ネットワーク版と同じのため省略

表 9: mijin エンドポイントと確認項目

	mijinLBEndpoint ロードバランサーを通した mijin の API エンドポイントです。API ノードを負荷分散しますが、ソース IP によるスティッキーセッションが有効です。詳細はこちら
	mijinEndpoint API ノード（EC2 インスタンス）の直接アクセス用の API エンドポイントです。ロードバランサーを介さずに接続できます。
	**chainInfo** mijin の現在のブロック数を確認できます。ブロック数が「2」以上であることを確認してください。
	**harvestAddress** AWS Systems Manager パラメータストアに登録された、通貨分配用アドレスへのリンクです。
	**emptyAddress** AWS Systems Manager パラメータストアに登録された、未使用アドレスのリンクです。
	**nodePeers** mijin API からノードの接続状態を確認できます。API ノード 1 台と設定された PEER ノード数が表示されていれば正常です。

日: これで mijin Catapult を使用する準備が整いました。それでは次の項で操作を始めてみましょう!

2.2.4 トライアル版の mijin をデプロイする

トライアル版では、パラメーターを変更することで環境にあったネットワークを構築することが可能です。
以下にパターン例を示します。

日: 本ページは AWS マーケットプレイス内にある cloudformation から mijin Catapult トライアル版を起動する手順となります。

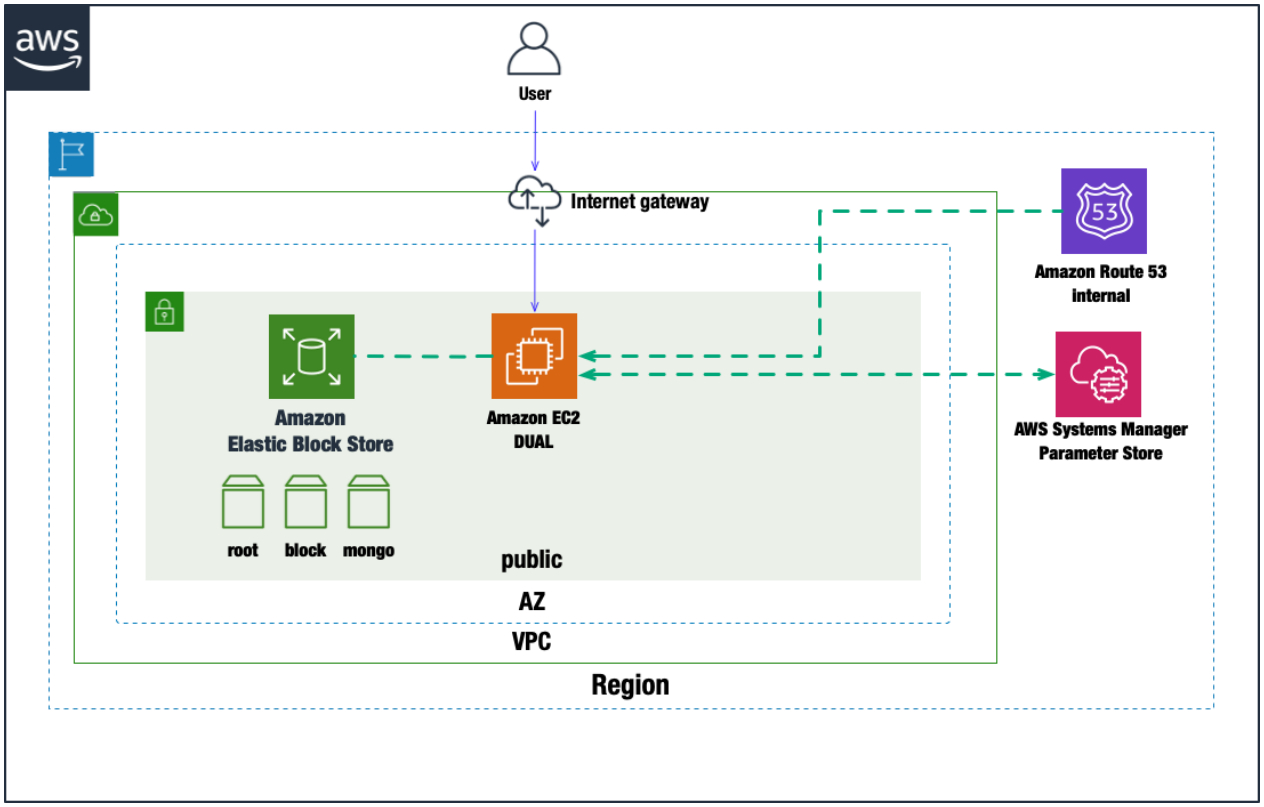
2.2.4.1 AWS 使用サービス

- Amazon EC2
- Amazon EBS
- Amazon Route53
- Amazon VPC(Nat Gateway)
- パラメータストア

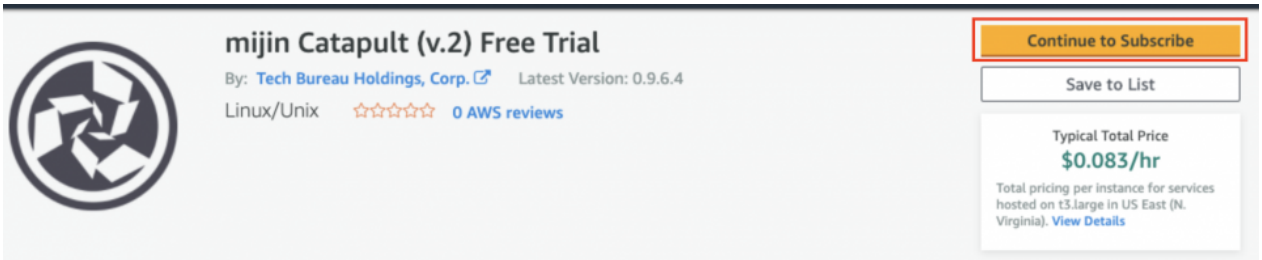
2.2.4.2 View Network

注釈:

トライアル版では、シングル AZ のみの構成となっています。
配置はシングルリージョンとなりますが、世界 20 リージョン毎にデプロイすることが可能です。

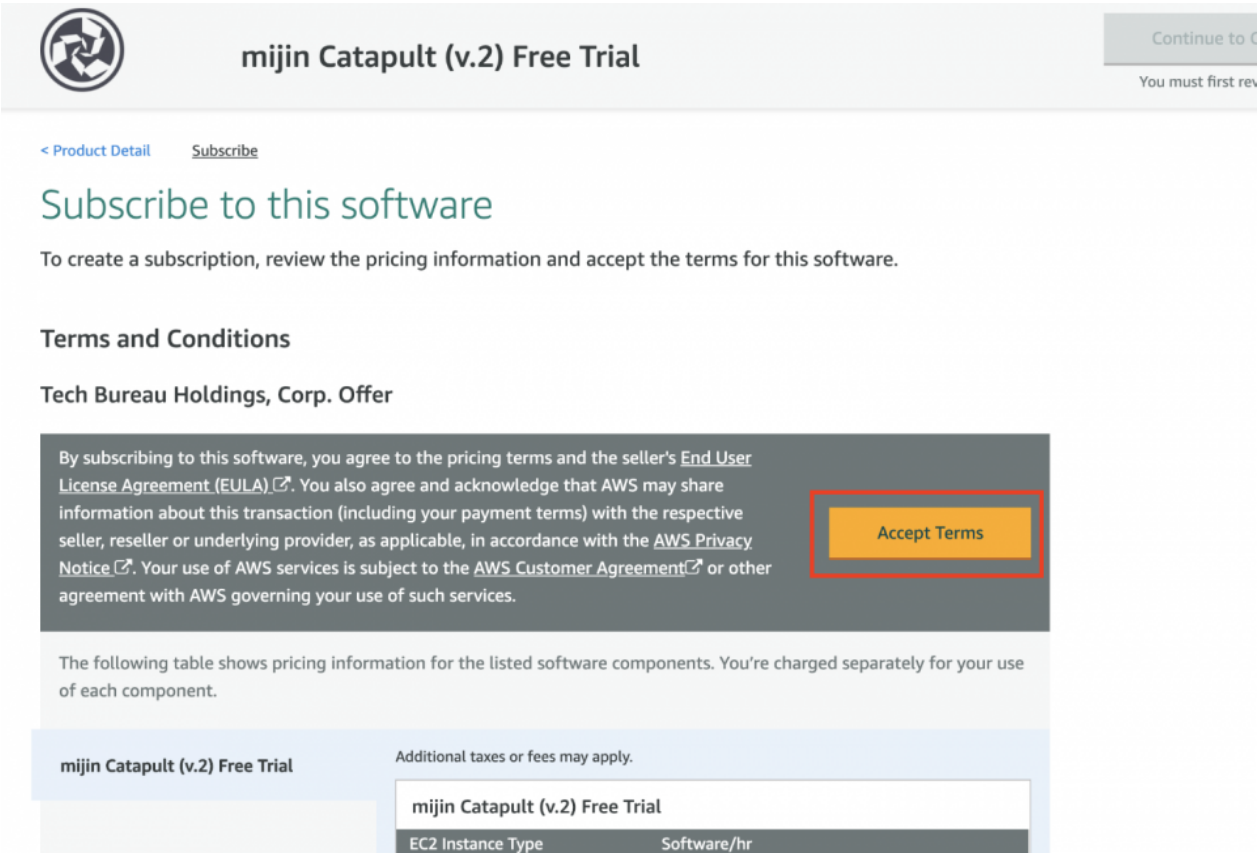


2.2.4.3 Step.1



日: mijin Catapult Free Trial の AMI を使用するためにサブスクライブする必要があります。赤枠のボタンを押してください。

2.2.4.4 Step.2



日: mijin Catapult AMI を使用するため、使用の承認をしてください。

2.2.4.5 Step.3

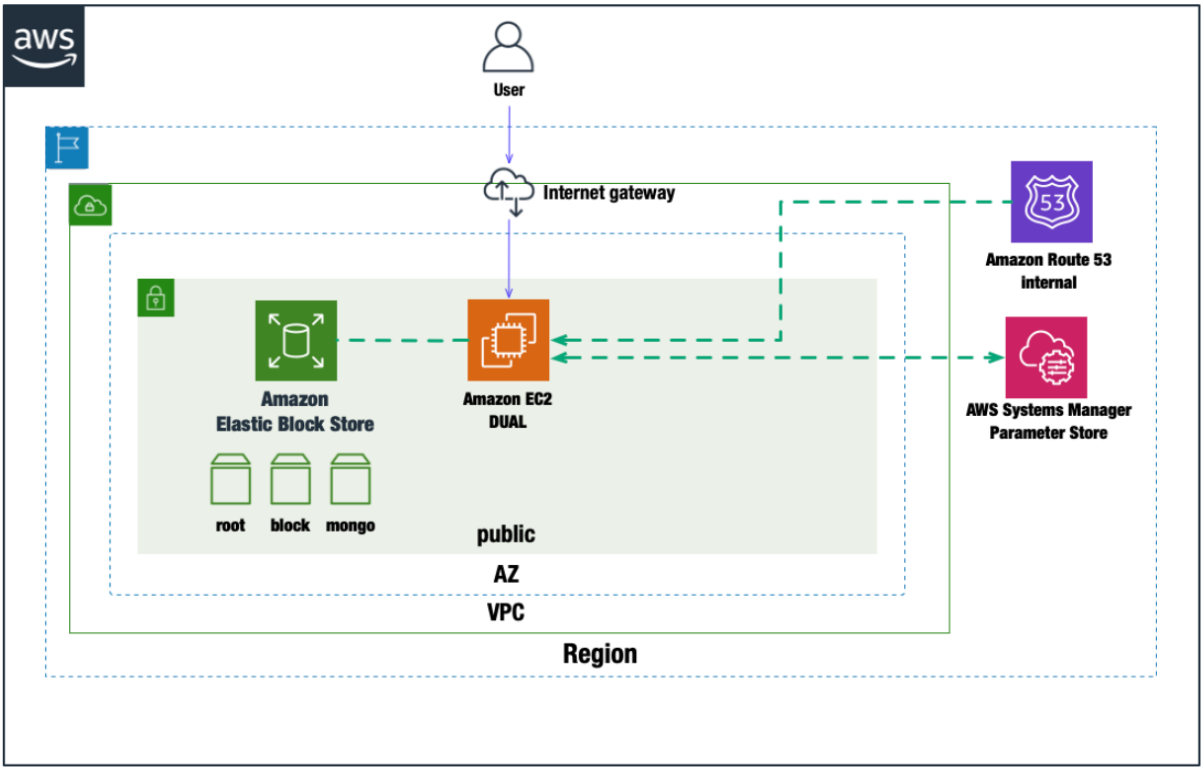
mijin Catapult Free Trial CFT 0.9.6.4
CloudFormation Template

This Cloudformation Stack makes it easy to build mijin.
Users create a new VPC and place a single EC2 instance in a public network.
After the Cloudformation execution is complete, the mijin address data is stored in the System Manager's Parameter Store. Using symbol-cli without having to remotely log in to the server, users are able to use the mijin immediately.

View Template Components

View Usage Instructions

Close CloudFormation Template

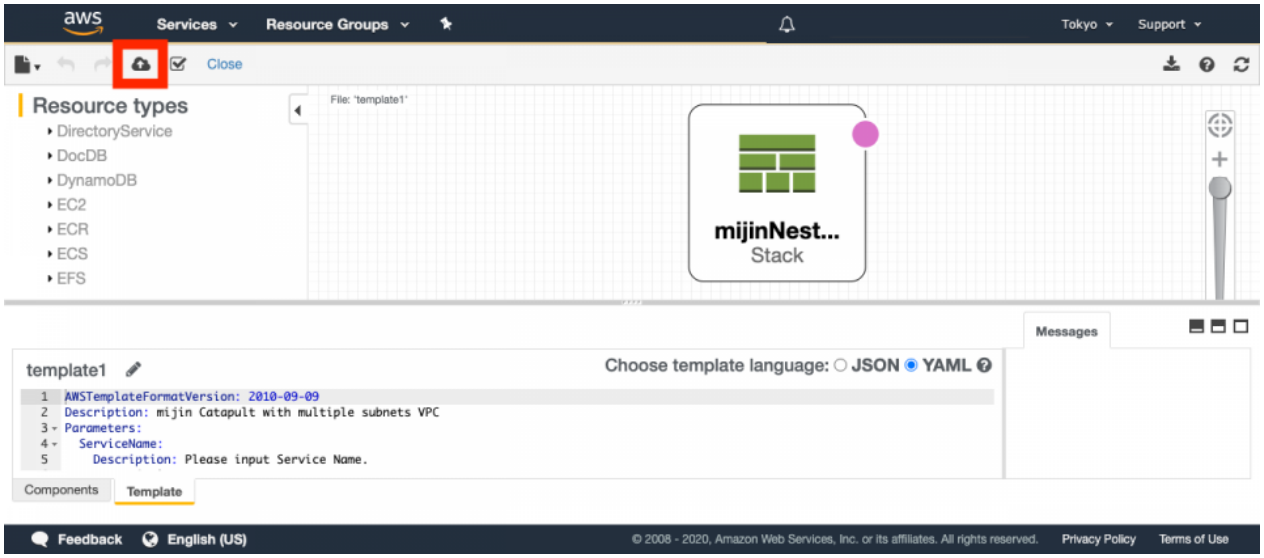


Download CloudFormation Template

View Template in CloudFormation Designer

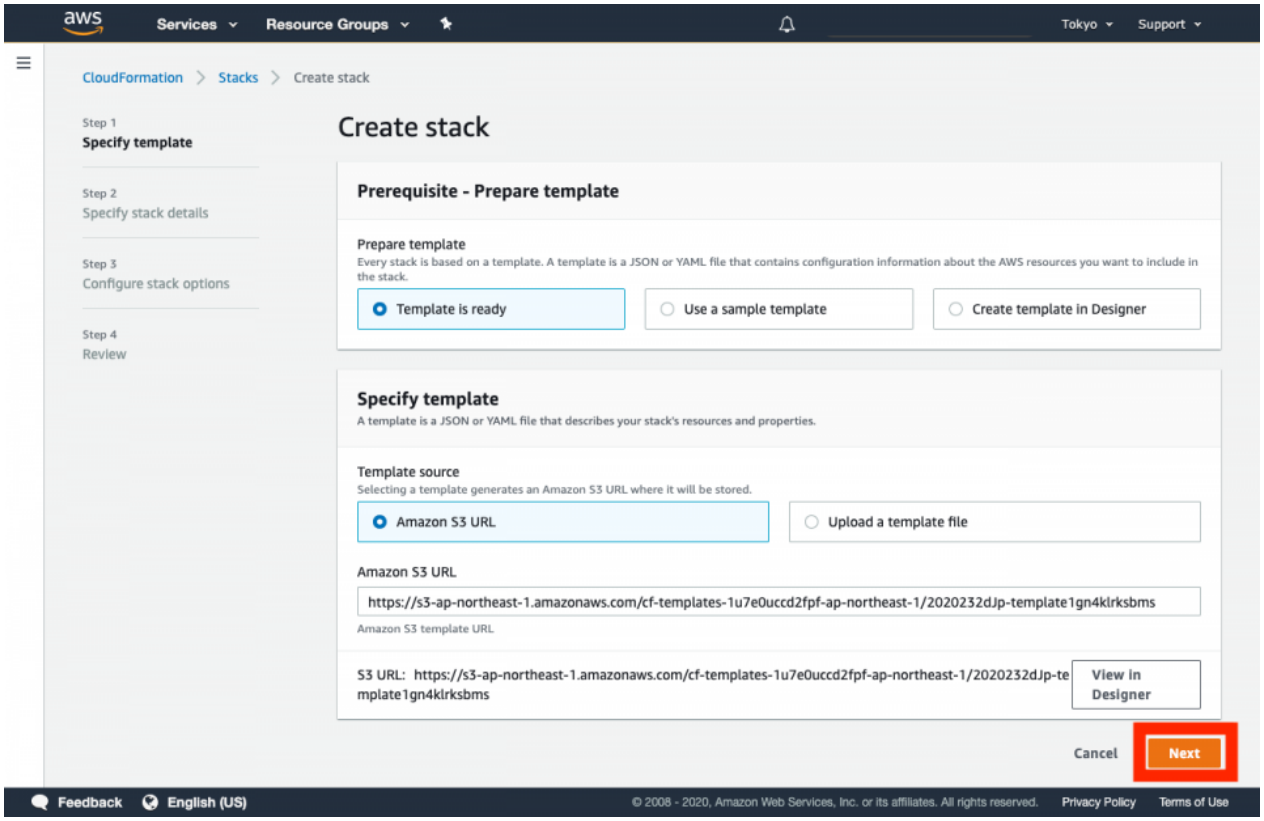
日: 図のようなネットワークを AWS CloudFormation によって作成するため、赤枠の文字をクリックしてください。

2.2.4.6 Step.4



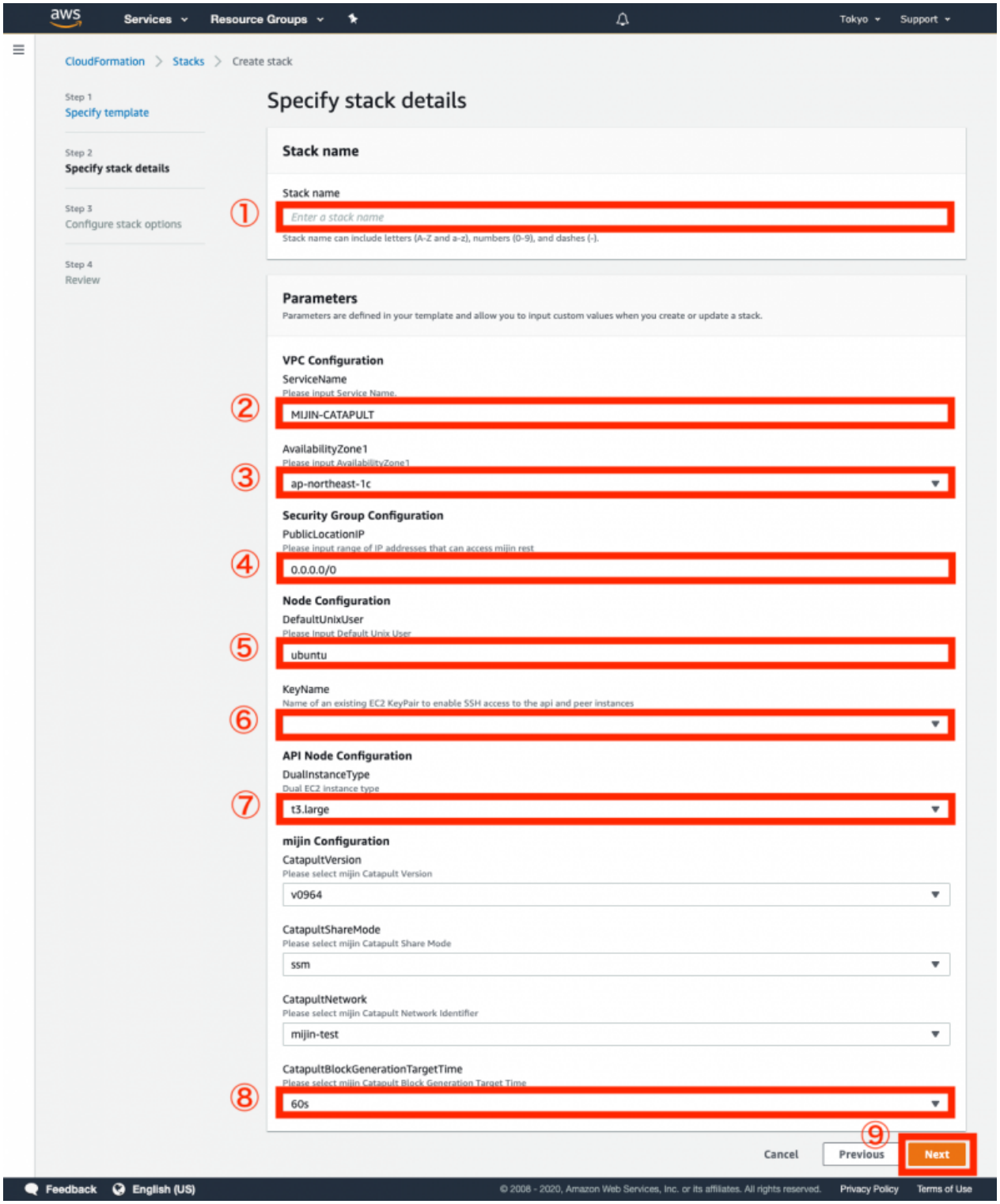
日: とくに編集せず、雲マークの CreateStack を押します。

2.2.4.7 Step.5



日: とくに編集せず、赤枠の「Next」を押します。

2.2.4.8 Step.6

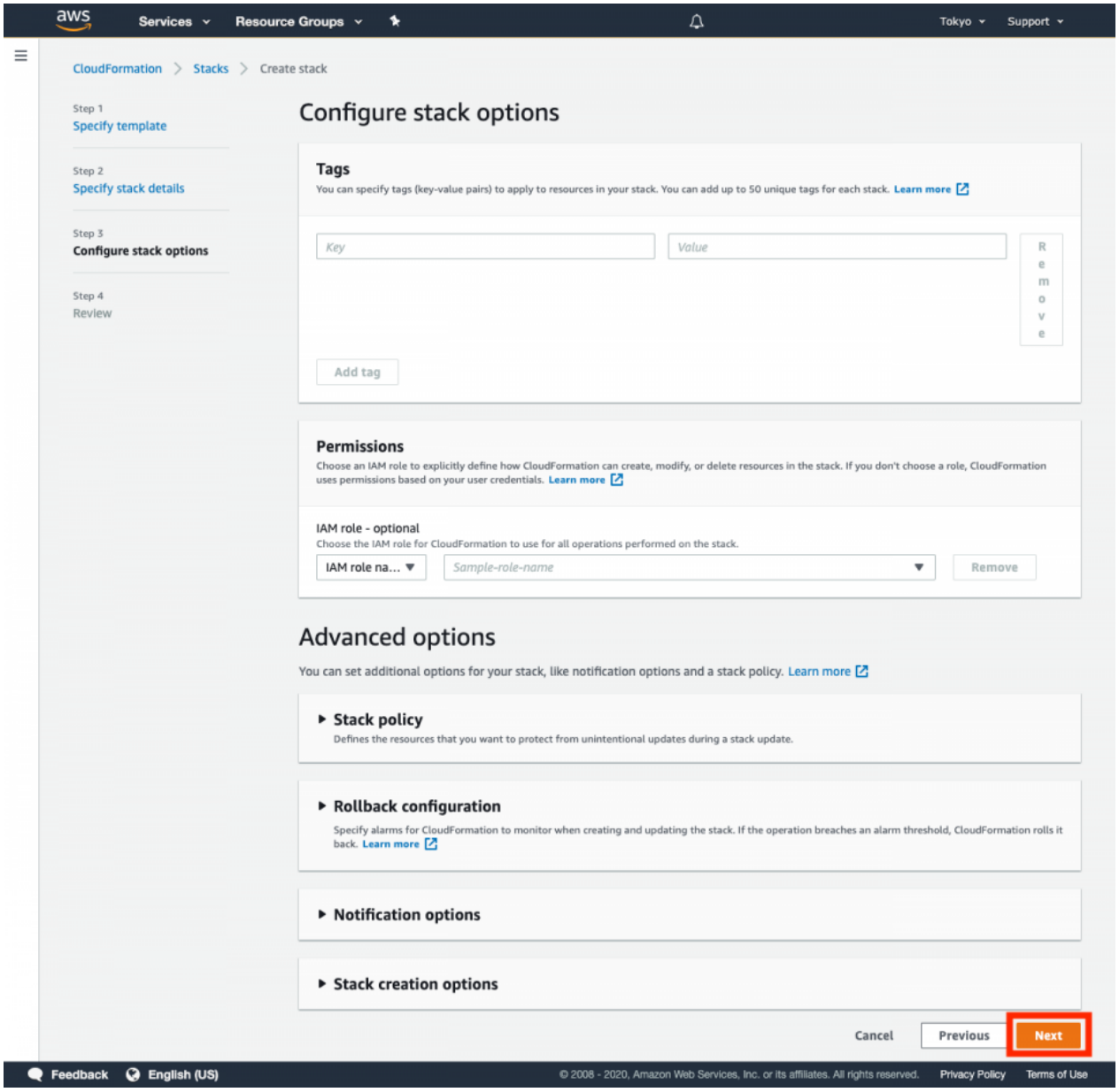


日: パラメータを入力します。

No	Parameter	Describe
①	Stack Name	このスタックにおける名前を記載してください。
②	Service Name	スタックによって作成されるサービス名を記載してください。 全リソースの冠名として使用されます。
③	Availability Zone	お使いのリージョンのアベイラビリティゾーンを選択してください。
④	Public Location IP	mijin Catapult の API への接続許可 IP アドレスを指定してください。 IP アドレスはレンジでも可能です（/24 など）
⑤	Default UnixUser	作成する EC2 インスタンスの標準の Unix ユーザーを記載してください。
⑥	KeyName	作成する EC2 インスタンスのリモート接続用の SSH 鍵を選択してください。 表示されていない場合、事前に鍵を作成する必要があります。 鍵の作成方法は こちら をご確認ください。
⑦	Dual InstanceType	作成する EC2 インスタンスのスペックを選択してください。
⑧	Catapult BlockGenerationTargetTime	mijin のブロック生成時間を選択してください。30 秒か 60 秒のみを選択できます。

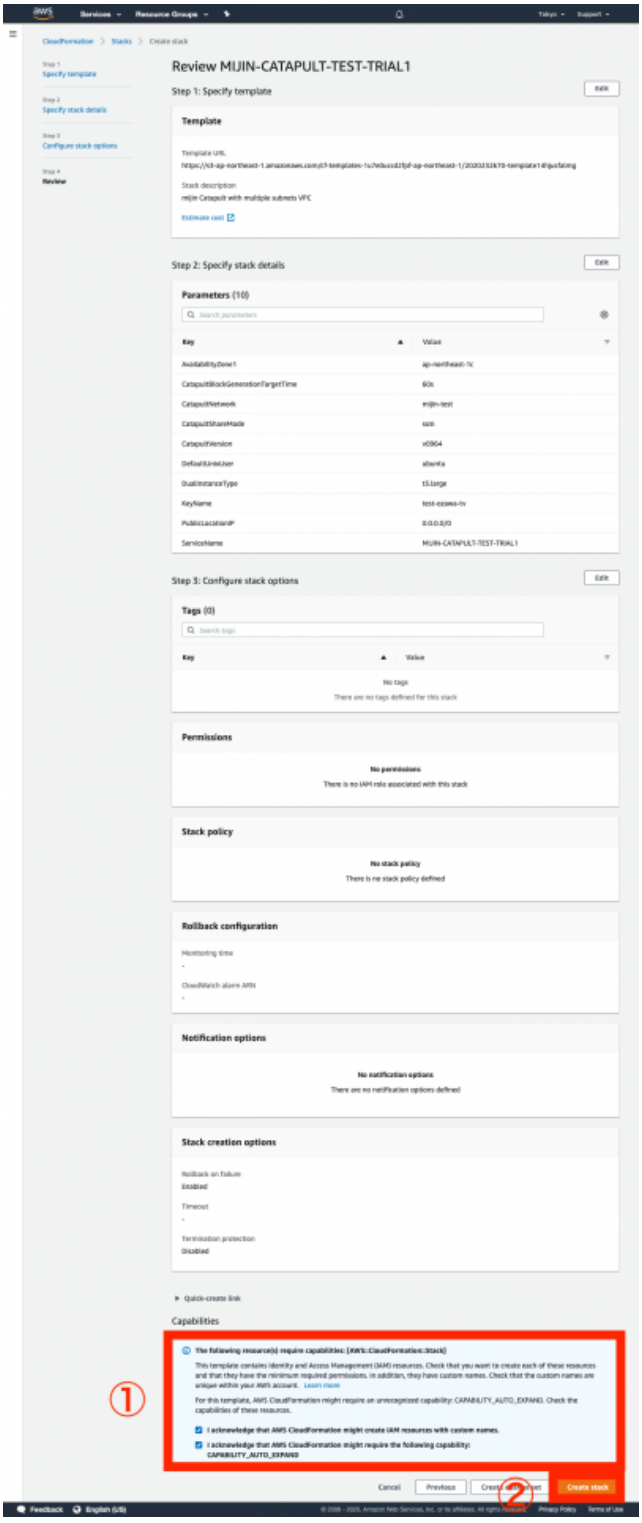
日: パラメータの入力完了後、⑨ の「Next」を押します。

2.2.4.9 Step.7



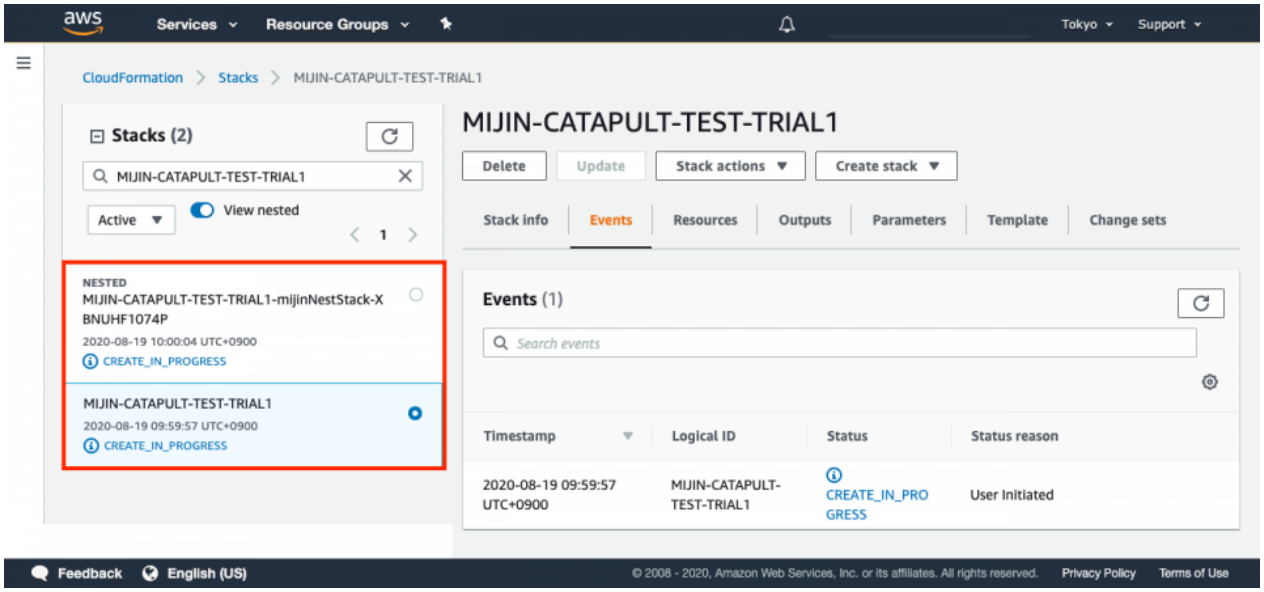
日: とくに編集せず、赤枠の「Next」を押します。

2.2.4.10 Step.8



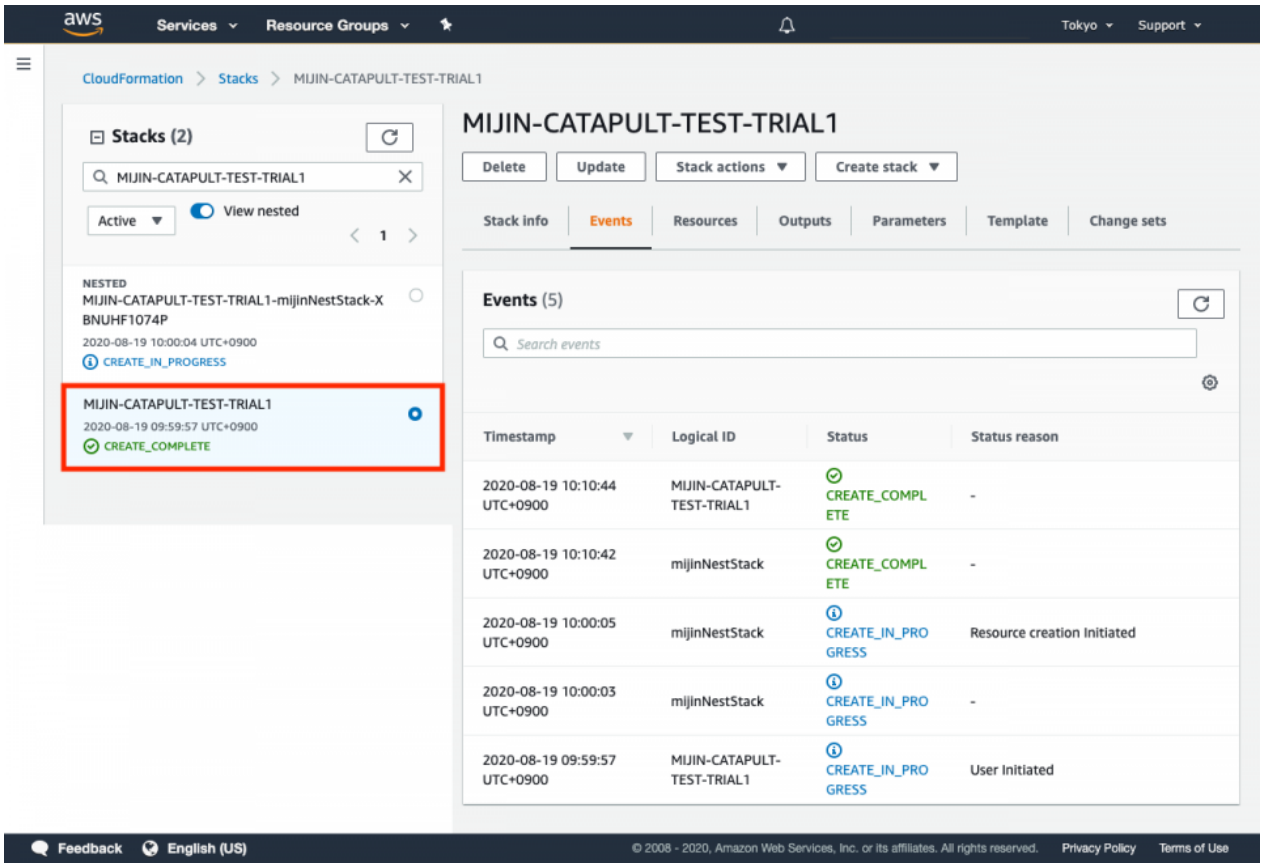
日: ① の赤枠にて２つの項目にチェックを入れます。日: ② の赤枠の「Create Stack」を押します。エラーがなければ作成が始まります。

2.2.4.11 Step.9



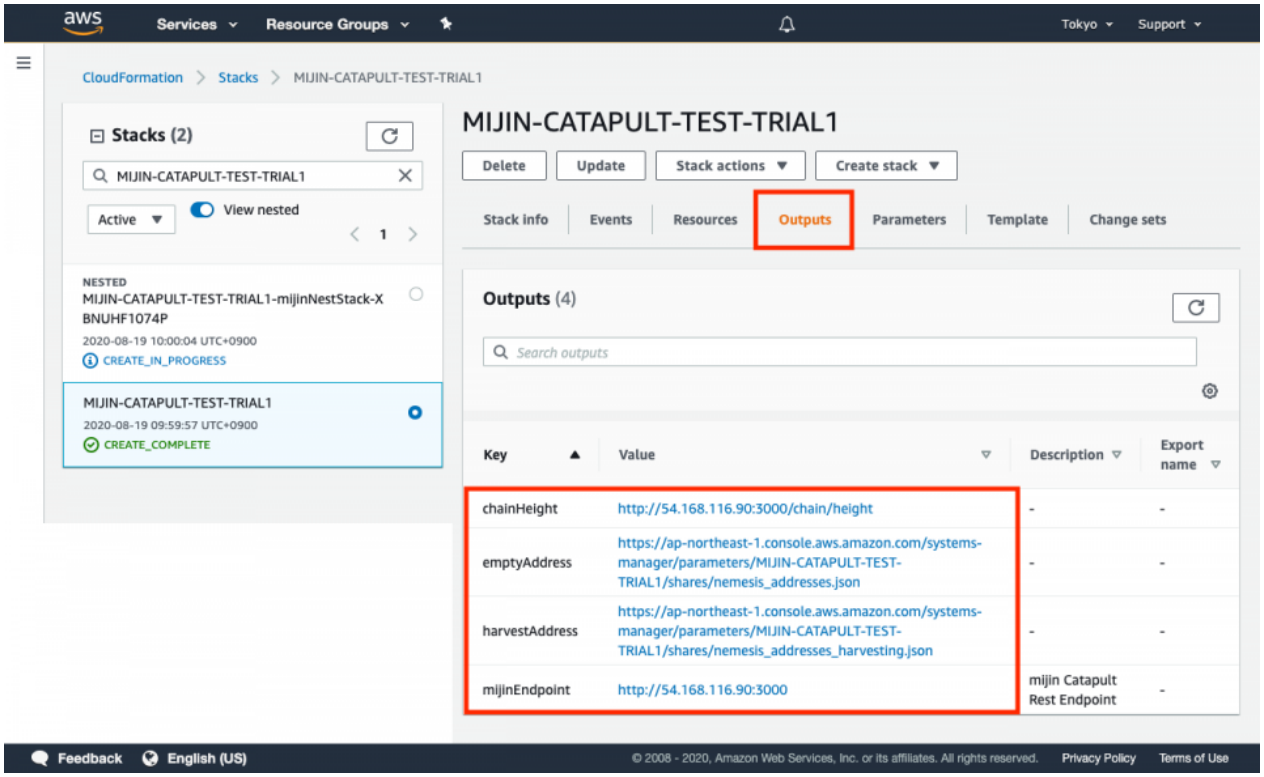
日: Stack が始まり「CREATE_IN_PROGRESS」になっていることを確認してください。この状態はおよそ 15~20 分程度かかります。

2.2.4.12 Step.10



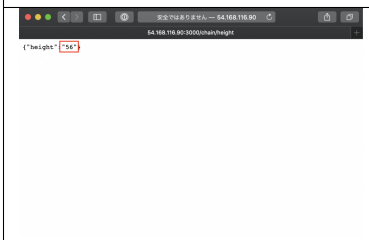
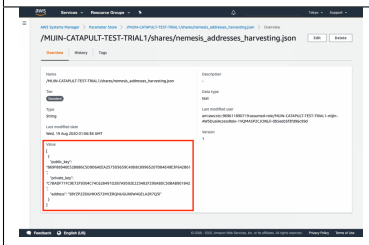
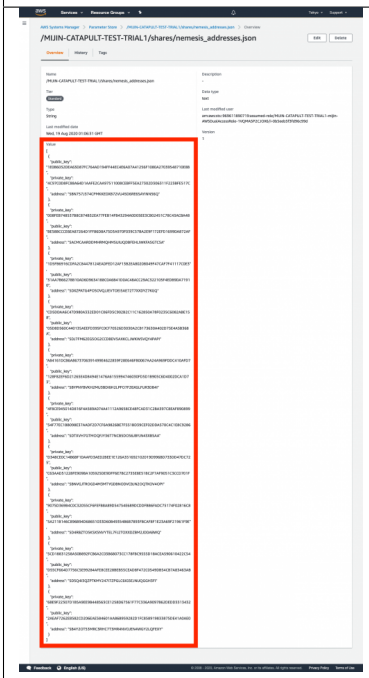
日: 「CREATE_COMPLATE」の状態であれば、mijin の作成が完了しました。

2.2.4.13 Step.11



日: 作成した Stack の「Outputs」を押すと、作成された mijin の設定情報を確認できます。

表 10: mijin エンドポイント確認項目（トライアル）

	mijinEndpoint mijin の API エンドポイントです。
	**chainHeight** mijin の現在のブロック数の確認ができます。ブロック数が「2」以上になっていることを確認してください。
	**harvestAddress** AWS Systems Manager パラメータストアに登録された、基軸通貨（トライアル版のため 2000cat.currency）を分配したアドレスのリンクです。
	**emptyAddress** AWS Systems Manager パラメータストアに登録された、未使用のアドレスのリンクです。

日: これで mijin Catapult を使用する準備が整いました。それでは次の項で操作を始めてみましょう!

2.2.5 AWS MarketPlace テクニカル資料

AWS MarketPlace における様々な技術資料を追加します。

2.2.5.1 AWS Marketplace Cloudformation パラメーター比較表

本章では、AWS Marketplace の mijin Catapult(v.2) をデプロイする際のパラメーターの説明とデフォルト値を説明します。
製品版２製品及びトライアル版が異なるデフォルト値であることを確認することができます。

Cloudformation パラメーター比較表

表 11: Trial / Product 設定一覧

No	カテゴリー	設定名	説明	設定値	制限値	Trial 設定可否	Trial デフォルト値	Product NewVPC デフォルト値	Product ExistsVPC デフォルト値
1	VPC Configuration	ServiceName	リソースの冠名となるサービス名を指定します。	String	記号始まりNG、大文字小文字英数字、ダッシュ（-）使用可能	○	MIJIN-CATAPULT	MIJIN-CATAPULT	MIJIN-CATAPULT
2		AvailabilityZone1	VPC で使用する AZ を指定します。	List	リージョンの AZ	○	リージョンに依存	リージョンに依存	•
3		AvailabilityZone2	VPC で使用する AZ を指定します（AZ1 とは別を指定）	List	リージョンの AZ	•	リージョンに依存	リージョンに依存	•
4		VPC	すでに存在する VPC Id を指定します。	List	リージョンにある VpcId	•	•	•	○
5		VpcCidrBlock	VPC の IP レンジを指定 （例：vpc-xxxx (xx.xx.xx.xx/16)）	List	正規表現: (d{1,3}.){3}d{1,3}/d{1,2}	•	•	•	○
6		Public1	公開ネットワーク（AZ1）のサブネットを指定	List		•	•	•	○
7		Public2	公開ネットワーク（AZ2）のサブネットを指定（Public1 と異なるもの）	List		•	•	•	○
8		Private1	非公開ネットワーク（AZ1）のサブネットを指定	List		•	•	•	○
9		Private2	非公開ネットワーク（AZ2）のサブネットを指定（Private1 と異なるもの）	List		•	•	•	○
10		InternalDomainName	内部 DNS 名（例：mijin.internal）を指定	String		•	•	mijin.internal	mijin.internal
11	Security Group Configuration	PublicLocationIP	mijin エンドポイントへの接続許可 IP レンジ	String	正規表現: (d{1,3}.){3}d{1,3}/d{1,2}	○			
12	Node Configuration	DefaultUnixUser	EC2 リモートログインユーザー名	String	"catapult" 以外	○	ubuntu	ubuntu	ubuntu
13		KeyName	EC2 で使用する SSH 鍵	List		○	リージョンに依存	リージョンに依存	リージョンに依存
14	API Node Configuration	ApiPlacementNetwork	API ノード配置（Public または Private）	List	Public, Private	•	•	Public	Public
15		ApiInstanceType	API インスタンスのスペック	List	複数	△	t3.large	t3.large	t3.large
16		ApiRootVolumeSize	ルートボリュームの容量（GB）	List	30, 100	•	30	30	30
17		ApiBlockVolumeSize	ブロックデータ用のディスク容量（GB）	List	50, 300, 500, 800, 1000	•	50	500	500

次のページに続く

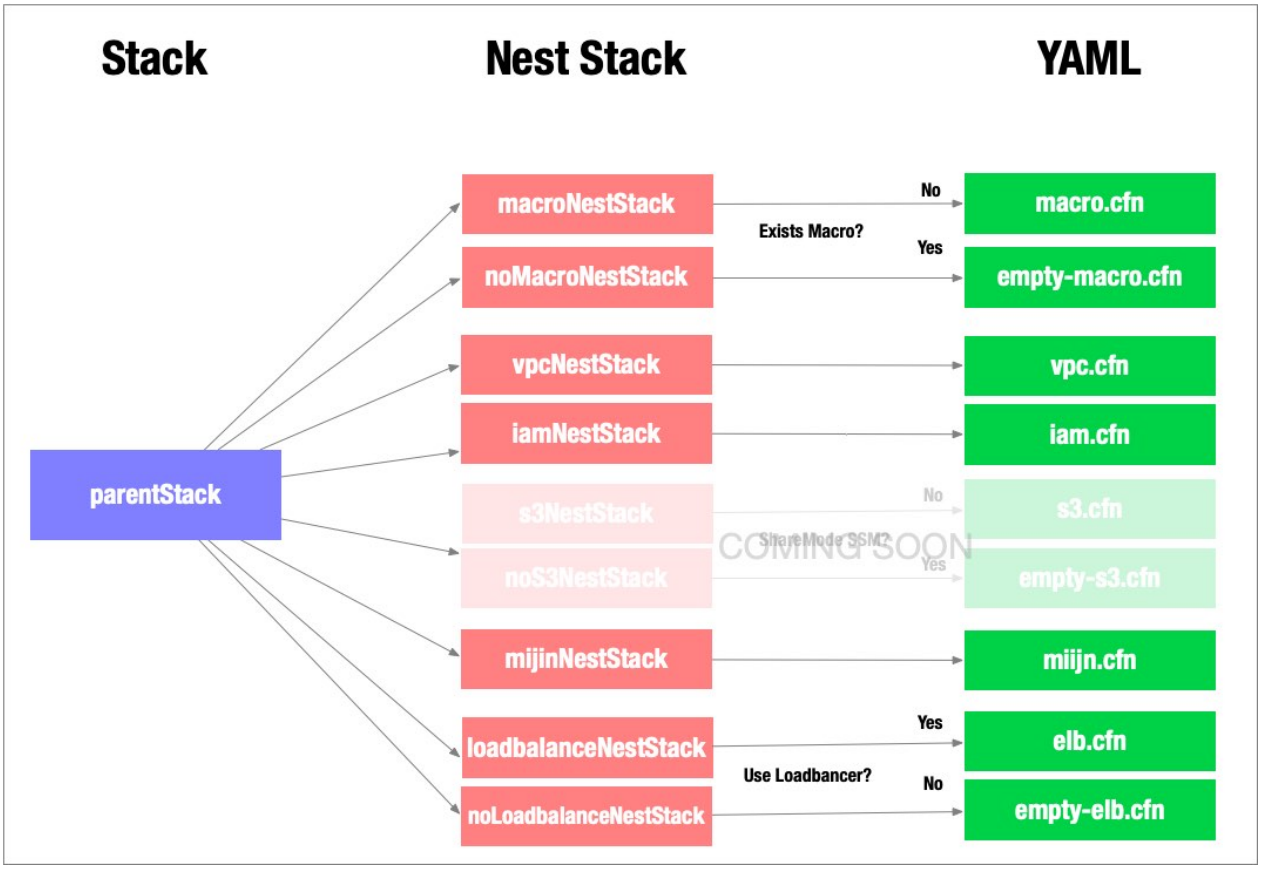
表 11 – 前のページからの続き

No	カテゴリー	設定名	説明	設定値	制限値	Trial 設定可否	Trial デフォルト値	Product NewVPC デフォルト値	Product ExistsVPC デフォルト値
18		ApiBlockVolumeIops	ブロックデータ用 IOPS（gp3）	List	3000, 5000, 10000	•	100	3000	3000
19		ApiMongoVolumeSize	MongoDB 用のディスク容量（GB）	List	50, 300, 500, 800, 1000	•	50	300	300
20		ApiMongoVolumeIops	MongoDB 用 IOPS（gp3）	List	3000, 5000, 10000	•	300	3000	3000
21	PEER Node Configuration	PeerNumberOfUnits	PEER ノード台数（固定）	Int	3–10	•	•	3	3
22		PeerInstanceType	PEER ノードのインスタンスタイプ	List		•	•	t3.large	t3.large
23		PeerRootVolumeSize	PEER ノードのルートディスク容量	List	30, 100	•	•	30	30
24		PeerBlockVolumeSize	PEER ノードのブロックデータ用容量	List	50, 300, 500, 800, 1000	•	•	500	500
25		PeerBlockVolumeIops	PEER ノードの IOPS（io1）	List	3000, 5000, 10000	•	•	3000	3000
26	mijin Configuration	CatapultVersion	起動時の Catapult バージョン	List	v10037,v10038	△	v10038	v10038	v10038
27		CatapultShareMode	初期データ（アドレス等）の保存場所（SSM 推奨）	List	ssm	△	ssm	ssm	ssm
28		mijinDataDiretoryName	Volume マウント先のパスを指定	String	絶対パス	•	/mnt/mijin	/mnt/mijin	/mnt/mijin
29		CatapultNetwork	mijin のネットワークタイプ	List	mijin, mijin-test	△	mijin-test	mijin	mijin
30		CatapultBlockGenerationTargetTime	ブロック生成間隔（目安）	List	5s, 15s, 30s, 60s	△	60s	15s	15s
31		CatapultEffectiveFee	トランザクション手数料有無	Boolean	Yes, No	•	Yes	No	No
32		MaxCosignedAccounts	最大連署アカウント数	List	25, 50, 100, 1000	•	25	25	25
33		FinalizationType	ファイナライズ方式		Deterministic, Probabilistic	•	Deterministic	Deterministic	Deterministic
34		MaxTransactionperBlock	ブロック内最大トランザクション数	List	6'000, 10'000, 20'000, 50'000, 100'000	•	6'000	6'000	6'000
35		RestThrottring	API 接続数（バースト時 +100）	List	30tps, 100tps, 200tps, 500tps, NoLimit	•	30tps	30tps	30tps
36		UnconfirmCacheSize	未承認トランザクションのキャッシュサイズ	List	Small, Medium, Large	•	Small	Small	Small
37	LoadBalancer Configuration	UseLoadBalancer	NLB を使用するか	Boolean	Yes, No	•	•	Yes	Yes
38		LoadBalancerType	NLB の配置場所	List	external, internal	•	•	external	external
39	Other	mijinStackAlreadyExists	既にスタックが存在するかの指定	Boolean	Yes, No	•	•	No	No

2.2.5.2 AWS MarketPlace Cloudformation 仕様

AWS MarketPlace にて提供している、mijin Catapult(v.2) は、オーケストラレーションツールである Cloudformation にてデプロイを行なっています。
本章では、Cloudformation を使って作成される AWS リソースの説明をします。

Cloudformation Template(CFT) の構成は複数のファイルから構成されており、親 Stack が子の Stack をそれぞれ呼び出しています。親 Stack のパラメータによって、呼び出される子の Stack が変わります。



macroNestStack

macroNestStack では、Cloudformation Macro を作成します。パラメータ mijinStackAlreadyExist を Yes の場合、Macro を作成しないスタックが (empty-macro) を呼び出します。これは、Cloudformation Macro がユニークな名前で作成するため、複数同一名で作成することができないためです。

Cloudformation Macro は Lambda(Node.js) で作成されており、この後に出てくる mijinNestStack の CFT を指定パラメータによって変換します。
この作成した Lambda から Amazon CloudWatch Logs への読み書き権限を許可をするため、以下の IAM ロール及びポリシーを割り当てます。

```
Resources:
  PeerUnitsExecutionRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - lambda.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      Path: /
      Policies:
        - PolicyName: root
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - 'logs:CreateLogGroup'
                  - 'logs:CreateLogStream'
                  - 'logs:PutLogEvents'
                Resource:
                  - Fn::Join:
                    - ':'
                    -
                      - 'arn:aws:logs'
                      - Ref: 'AWS::Region'
                      - Ref: 'AWS::AccountId'
                      - 'log-group:/aws/lambda/*:*:*'
```

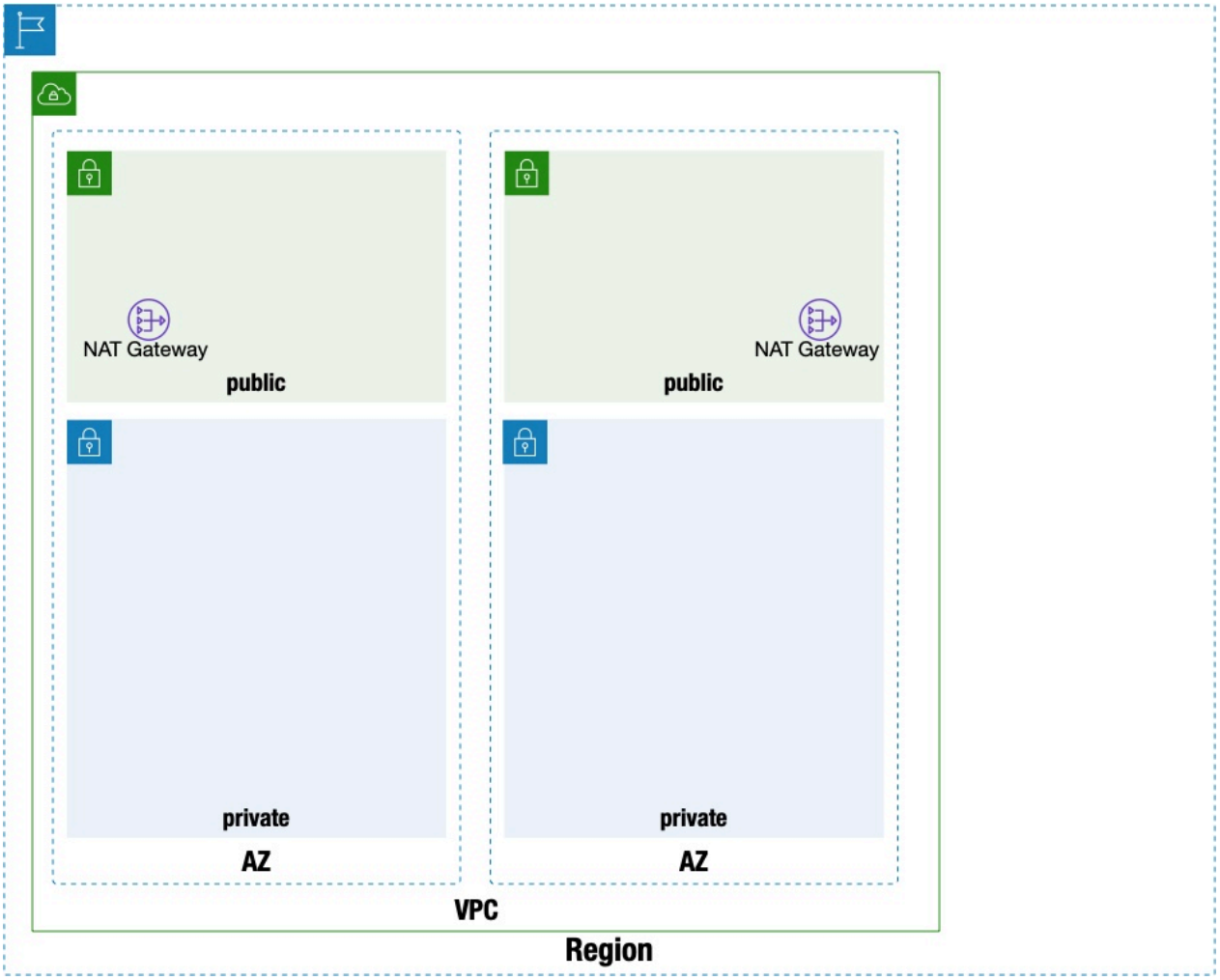
Macro では、パラメータ PeerNumberOfUnits の数に応じて、EC2 インスタンスのスタックを複製し、起動する EC 2インスタンスを動的に変更することができます。

警告:

同一リージョンに複数の mijin Catapult(v.2) を作成する場合は、mijinStackAlreadyExist を YES にする必要があります。

vpcNestStack

vpcNestStack では、新たに VPC を作成します。マルチ AZ 環境を構築し、各 AZ にパブリックサブネット・プライベートサブネットを配置します。また、プライベートネットワークのルーティングのデフォルトゲートウェイは同じ AZ にあるパブリックネットワークに配置した Nat Gateway を使用してインターネットに出ることができます。



注釈: 既存ネットワークに mijin を展開する場合は、このスタックは使いません。

iamNestStack

iamNestStack では、EC2 インスタンスが使用する IAM ロールおよびそれに紐づく IAM ポリシーを作成します。

以下は、API ノード、PEER ノードそれぞれに割り当てるロールです。

```
AWSApiAccessRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Principal:
            Service:
```

(continues on next page)

(continued from previous page)

```
            - ec2.amazonaws.com
          Action:
            - 'sts:AssumeRole'
    Path: /
AWSPeerAccessRole:
  Type: 'AWS::IAM::Role'
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - ec2.amazonaws.com
          Action:
            - 'sts:AssumeRole'
    Path: /
```

AWS Systems Manager Session Manager の権限を付与します。
AWS Systems Manager Session Manager では、EC2 インスタンスに IAM 権限でリモートログインできるように権限を付与し、IAM ロールに紐づけます。

```
AWSSSMRolePolicies:
  Type: 'AWS::IAM::Policy'
  Properties:
    PolicyName: AWSSSMAccessPolicy
    PolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Action:
            - 'ssm:DescribeAssociation'
            - 'ssm:GetDeployablePatchSnapshotForInstance'
            - 'ssm:GetDocument'
            - 'ssm:GetManifest'
            - 'ssm:GetParameters'
            - 'ssm:ListAssociations'
            - 'ssm:ListInstanceAssociations'
            - 'ssm:PutInventory'
            - 'ssm:PutComplianceItems'
            - 'ssm:PutConfigurePackageResult'
            - 'ssm:UpdateAssociationStatus'
            - 'ssm:UpdateInstanceAssociationStatus'
            - 'ssm:UpdateInstanceInformation'
          Resource: '*'
        - Effect: Allow
          Action:
            - 'ssmmessages:CreateControlChannel'
            - 'ssmmessages:CreateDataChannel'
            - 'ssmmessages:OpenControlChannel'
            - 'ssmmessages:OpenDataChannel'
          Resource: '*'
```

AWS Systems Manager パラメータストアへの読み書き権限を付与します。
AWS Systems Manager パラメータストアでは、mijin Catapult(v.2) の最初のノードの動的生成されたデータを保存し、その他の各ノードから参照できる権限を付与し、IAM ロールに紐づけます。

```
AWSPSAccessRolePolicies:
  Type: 'AWS::IAM::Policy'
  Properties:
    PolicyName: AWSPSAccessRole
    PolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Action:
            - 'ssm:PutParameter'
            - 'ssm:GetParameter'
            - 'ssm:GetParametersByPath'
```

同一アカウント同一サービス名の IAM ロールのリソースからの Security Token Service(STS) の付与し、IAM ロールを紐づけます。
STS が付与されていることで、AWSApiAccessRole 及び AWSPeerAccessRole の IAM ロールはポリシー指定した AWS サービスを操作できるようになります。

```
AWSAssumeAccessRolePolicies:
  Type: 'AWS::IAM::Policy'
  Properties:
    PolicyName: AWSAssumeAccessRole
    PolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Action:
            - 'sts:AssumeRole'
          Resource:
            - Fn::Join:
                    - ':'
                    -
                      - 'arn:aws:iam:'
                      - Ref: 'AWS::AccountId'
                      - !Sub "role/${ServiceName}*"
    Roles:
      - !Ref AWSApiAccessRole
      - !Ref AWSPeerAccessRole
```

s3NestStack

警告: この機能は現在無効化されており、パラメーターストアのみしか保存することができません。

s3NestStack では、mijin のデータを AWS Systems Manager パラメータストアに配置するか、S3 バケットに保存するかを選択し、S3 を選択した場合、S3 バケットを作成します。

mijinNestStack

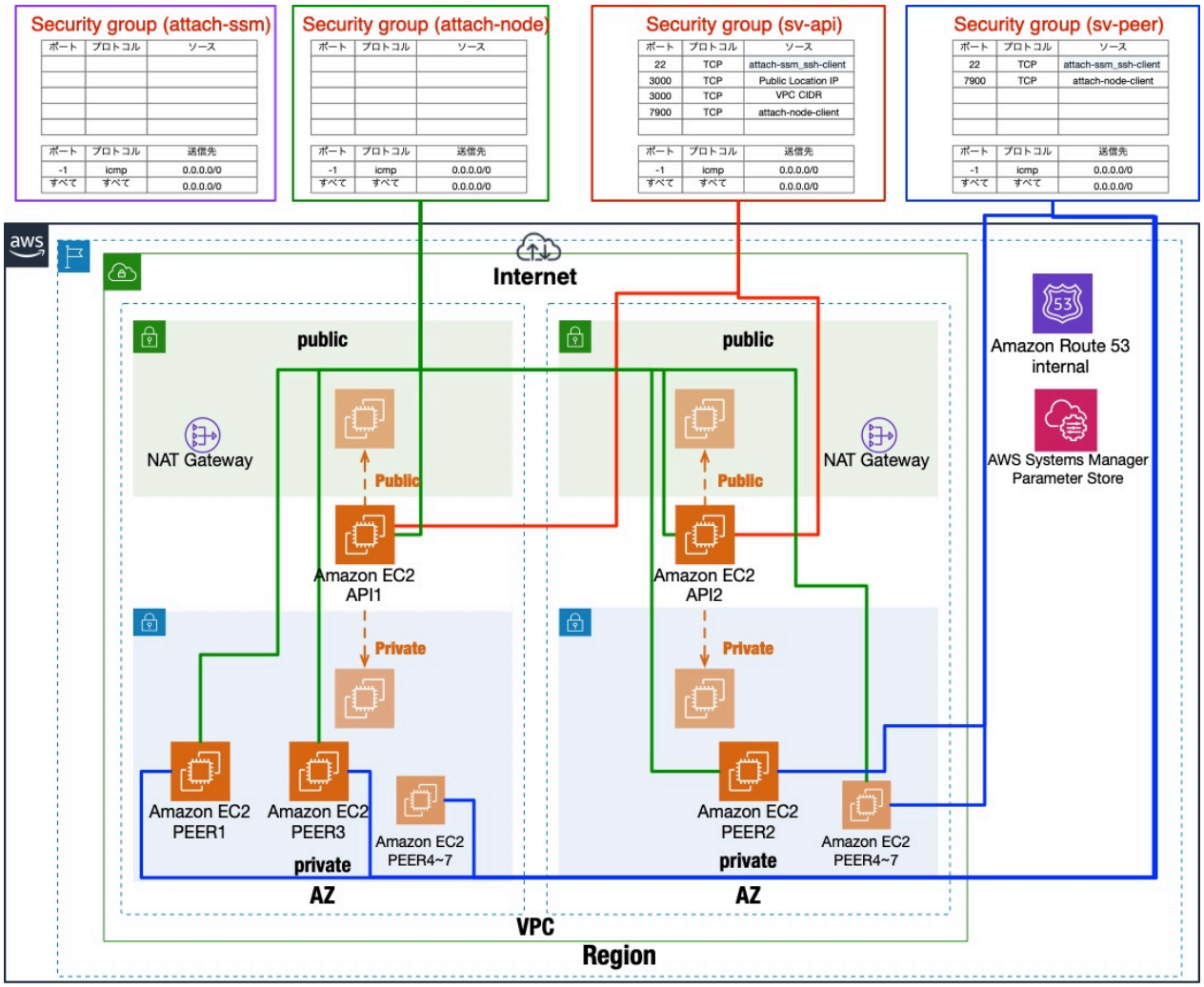
mijinNestStack では、Route53 によるプライベート DNS、セキュリティグループ、EBS、EC2 インスタンスを作成します。

mijin は各 EC2 インスタンス間を DNS 名で通信を行います。ドメインは mijin.internal で固定になり、各インスタンス名はそれぞれ以下のように、A レコード設定されます。

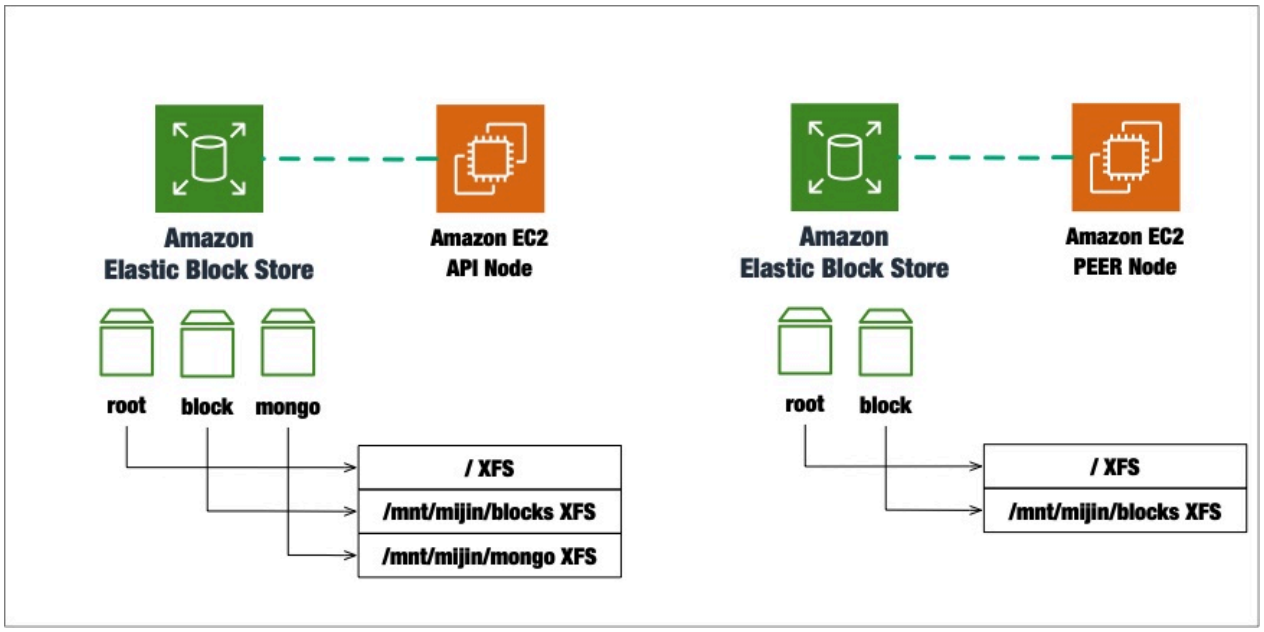
1. api1.mijin.internal
2. api2.mijin.internal
3. peer1.mijin.internal
4. peer2.mijin.internal
5. peer3.mijin.internal
6. peer4.mijin.internal
7. peer5.mijin.internal
8. peer6.mijin.internal
9. peer7.mijin.internal
10. peer8.mijin.internal
11. peer9.mijin.internal

セキュリティグループは以下、画像のように最低限の通信に使用するセキュリティグループが適用されます。

セキュリティグループ名	説明
attach-ssm_ssh-client	このセキュリティグループは踏み台など設定することで、SSH ログインが可能になります。新規 VPC では使用しません。(既存 VPC 用で作成した場合、既存の踏み台に割り当てるなど)
attach-node-client	ノード間通信用です。
sv-api	API ノード用です。 3000 ポート / REST アクセス用です 7900 / mijin ノード通信用です。
sv-peer	PEER ノード用です。 7900 ポート / mijin ノード通信用です。



EBS は mijin のデータ領域を保存するため、API ノードでは、root パーティション以外に 2 つの EBS、PEER ノードでは、一つの EBS をアタッチしています。
VolumeType は `GP3` 固定であり、ディスクの暗号化はしていません。
EC2 インスタンス初回構築時のみ、cloud-init を使って、EBS を XFS フォーマットし、ディスクマウントするように設計されています。
mijin データは `block` に保存されます。API ノードでは、block` と rest で使用する `mongo` のデータを保存しますが、`block にあるデータがあれば、mongo データを復元することができます。



EC2 インスタンスは、mijin パッケージがインストールされたカスタム AMI を使用し起動します。UserData にて、cloud-init を動作し、初期パッケージの設定を実行します。UserData の実行結果を受け取り、設定に失敗するとロールバックするように動作します。

1. パラメータで指定した Unix ユーザーの設定
2. ホスト名の設定
3. OS パッケージの更新
4. pip のインストール
5. cloudformation helper script のインストール
6. **mijin のセットアップ**
    1. EBS フォーマット
    2. mijin セットアップ (api1 はパラメータストアにデータをアップ)

また、インスタンスを作成する順序としては、以下のようになります。

1. ApiInstance1
2. ApiInstance2 PeerInstanceX 並行実行

ApiInstance1 にて、すべてのノードに使用する設定を作成し、AWS Systems Manager パラメータストアにデータを保存します。その他インスタンスはこのパラメータストアからデータを取得し、mijin を作成します。
パラメーターストアに保存される内容は、以下となります。

パラメータ名	説明
/デプロイ時に指定した冠名/shares/api_node.json	API ノードが使用する公開鍵
/デプロイ時に指定した冠名/shares/generation_hash.json	mijin Catapult(v.2) のブロックチェーンのジェネシスハッシュ (GenerationHash)
/デプロイ時に指定した冠名/shares/harvest_fee_sink_public_key.json	Harvest を受け取るアドレス (mijin では不要)
/デプロイ時に指定した冠名/shares/init_host_count.json	デプロイ時に作成したノード数
/デプロイ時に指定した冠名/shares/mosaic_rental_fee_sink_public_key.json	Mosaic レンタル費用を受け取るアドレス
/デプロイ時に指定した冠名/shares/namespace_rental_fee_sink_public_key.json	Namespace レンタル費用を受け取るアドレス
/デプロイ時に指定した冠名/shares/nemesis_addresses.json	とくに使用されてない空のアドレス (使用可能)
/デプロイ時に指定した冠名/shares/nemesis_addresses_harvesting.json	harvest などを受け取るアドレス
/デプロイ時に指定した冠名/shares/nemesis_addresses_harvesting_voting.json	ファイナライズの権限に使用するアドレス
/デプロイ時に指定した冠名/shares/nemesis_addresses_harvesting_vrf.json	セキュリティ強化用のアドレス (ブロック生成できる状態をわからなくする)
/デプロイ時に指定した冠名/shares/peer_node.json	PEER ノードが使用する公開鍵
/デプロイ時に指定した冠名/shares/rest_gateway_private_key.json	API ノードが使用する REST 用のアドレス
/デプロイ時に指定した冠名/shares/signer_private_key.json	Nemesis(Genesis) ブロックを署名するアドレス
/デプロイ時に指定した冠名/shares/new-cert/各ノード/CA/[*].pem	ノード間の通信を暗号化する SSL 証明書

注釈:

このパラメータストアの値はブロックチェーンの 1 ブロック目に作成する値として保存され、デプロイ以降、パラメータストアから呼び出すことはありません。
そのため、セキュリティとしてデータを削除しておきたいなどがある場合、このデータを削除しても問題ありません。
また、障害や新規に増設したいなどがある場合などは、このデータから復旧することができます。

loadbalanceNestStack

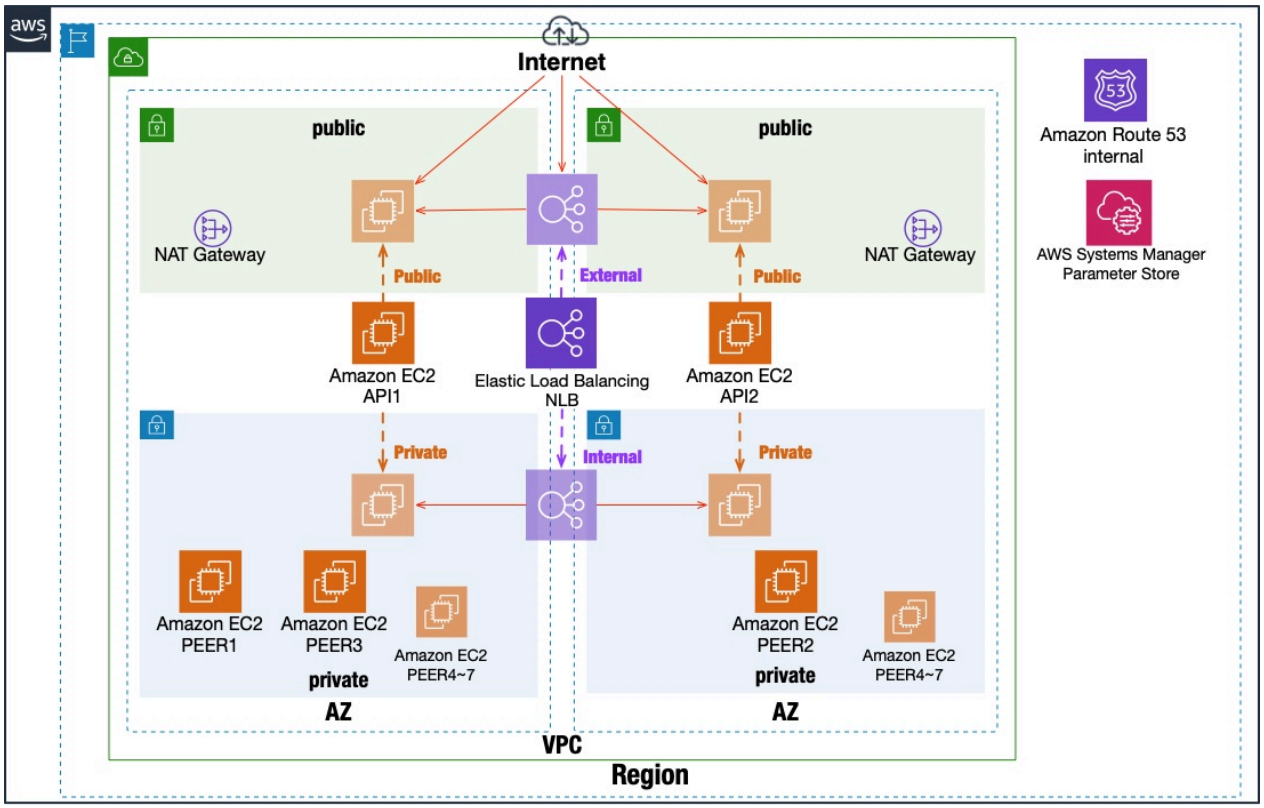
loadbalanceNestStack では、ELB(ロードバランサー) を作成します。
パラメータ UseLoadBalancer が No の場合は ELB を作成しないスタック (empty-elb) を呼び出します。
ELB は、API ノードの REST アクセスポート `3000` ポートへ分散します。
ELB の Type は NLB(ネットワークロードバランサー) を使用し、API ノードへの接続はスティッキーセッションにより同一セッションは一定期間、同じノードを使用する設定となっています。

注釈:

ELB は Network Load Balancer(NLB) のみで作成します。
NLB について理解したい場合は、以下を参照してください。
https://docs.aws.amazon.com/ja_jp/elasticloadbalancing/latest/network/introduction.html

```
TargetGroupAttributes:
  - Key: stickiness.enabled
    Value: 'true'
```

NLB は、プライベートのみで使用する内向き用の配置、インターネット経由で接続する外向き用の配置をパラメータで指定することができます。



API ノードの REST アクセスポート `3000`ポートへは以下のような条件でヘルスチェックをおこなっています。
ヘルスチェックは http://API ノード:3000/chain/info を死活監視しています。

```
Properties:
  HealthCheckIntervalSeconds: 10 # 10 秒間隔でチェックする Check at 10-second intervals.
  UnhealthyThresholdCount: 3 # 異常とみなす回数 Number of times considered abnormal
  HealthyThresholdCount: 3 # 正常とみなす回数 Number of times considered normal
  HealthCheckPath: /chain/info # ヘルスチェックをする URL URL for health check
  HealthCheckProtocol: HTTP
  Port: 3000 # ヘルスチェックポート health check port
```

2.2.5.3 AWS MarketPlace mijin Catapult(v.2) アーキテクチャパターンによるリカバリ戦略

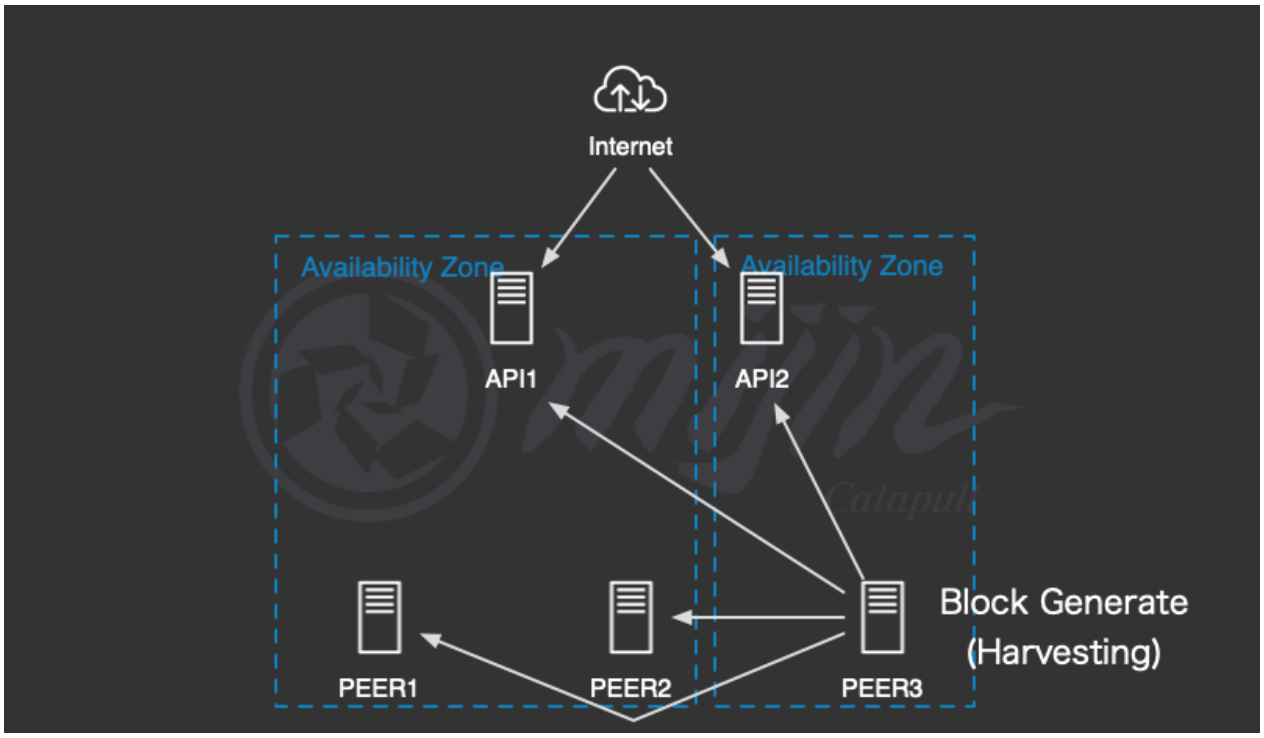
AWS MarketPlace で展開する mijin Catapult(v.2) は、デプロイ時のパラメーターによって様々なアーキテクチャパターンがあります。
本章では、アーキテクチャパターンを学び、ディザスタリカバリなどの BCP 対策によるリカバリ戦略を説明します。

mijin Catapult(v.2) の高可用性と耐障害性

mijin Catapult(v.2) は、全ノードに同じブロックチェーンデータを保持するため、最低でも PEER ノード１台あれば、ブロックチェーンの更新は継続します。(分散型フォールトトレランス)
ブロックチェーンは。RDB などレコード単位でデータ保存はせず、ブロック単位で纏めてデータ保存をする特徴があります。
そのため、障害があった状況によって、最新のブロックチェーンデータが必ずしも最新かつ正しいデータであるとは限りません。
理由としては、デプロイ時に設定したおよその秒数でブロックチェーンのブロックデータが生成され、ブロック生成するノード (ハーベスティングノード)、生成したブロックを受信するノードに分かれます。
ブロックチェーンデータを作成するノードはコンセンサスアルゴリズム (Proof-of-Stake Plus) によって、ブロックを生成するノード (ハーベスティングノード) が剪定されます。
コンセンサスアルゴリズムについて詳しく知りたい場合は、Symbol の Document を参照してください。

https://docs.symbol.dev/ja/concepts/consensus-algorithm.html

mijin にもブロック生成の仕組みによる報酬は存在しますが、プライベートブロックチェーンでは必要ないため、機能的に報酬プロセスが動いているだけの状態となります。
以下の図では、PEER3 でブロックチェーンを生成し、各ノードにブロックチェーンデータを送信する図です。

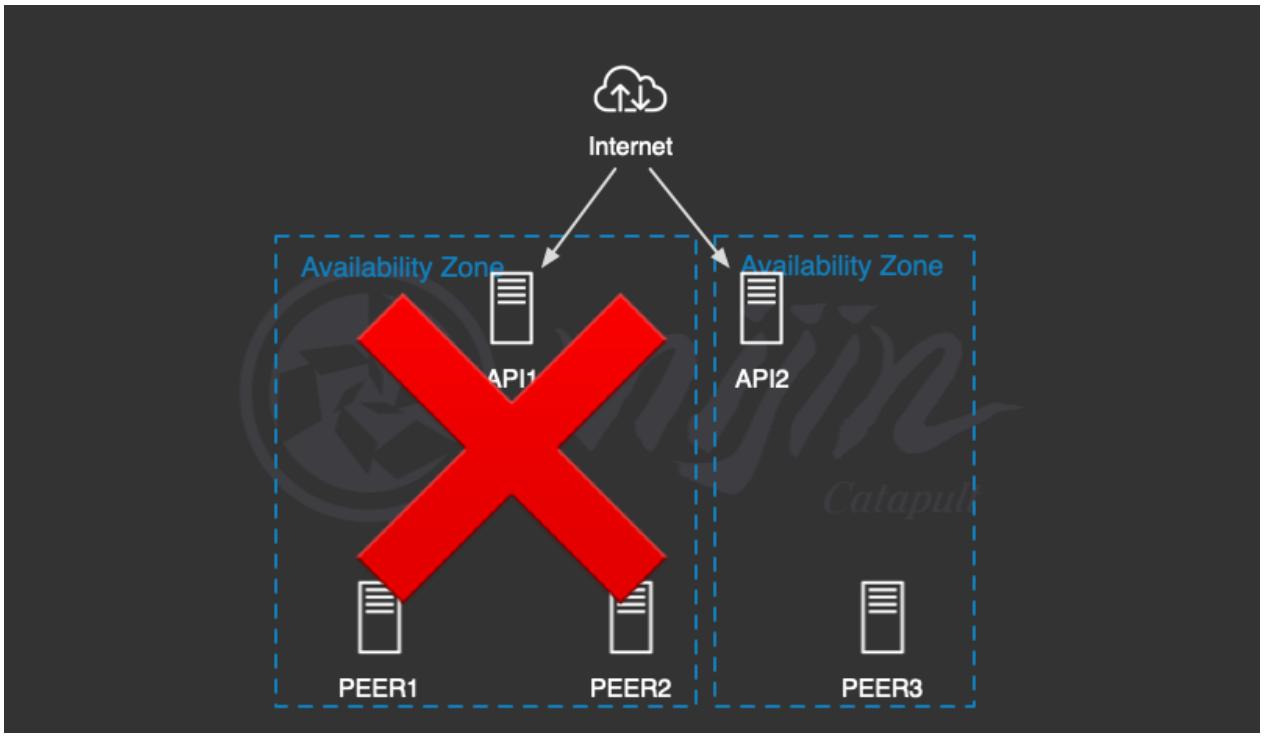


PEER3 から生成したブロックデータを受信していない状態で PEER2 のみが残った状況で障害になった場合、最低でも 1 ブロック分の差分が出る可能性があります。
その際、PEER2 は PEER1 と PEER3 を認識できないため、新たなブロックチェーンデータを生成し、独立したノードに変わります。(フォーク)
ただし、PEER1 と PEER3 がすぐに復旧した場合、正しいブロックデータを持っている PEER3 を正としてロールバックし、正常のブロックチェーンデータへとリカバリします。
API ノードが全てダウンした場合は、プログラムなどからアクセスはできませんが、PEER ノードがいる限りはブロック生成は進みます。

マルチ AZ によるリカバリ戦略

目標復旧時間 (RTO)	ほぼ 0
目標復旧時点 (RPO)	ほぼ 0

Marketplace に展開する mijin Catapult(v.2) は、マルチ AZ 環境にノードをそれぞれ配置します。
片方の AZ 側に障害が発生したとしても、分散型フォールトトレランスである mijin Catapult(v.2) はサービス継続することが可能です。
API ノードへの接続を継続したい場合は、Elastic Load Balance を有効にすることで、可用性が向上します。



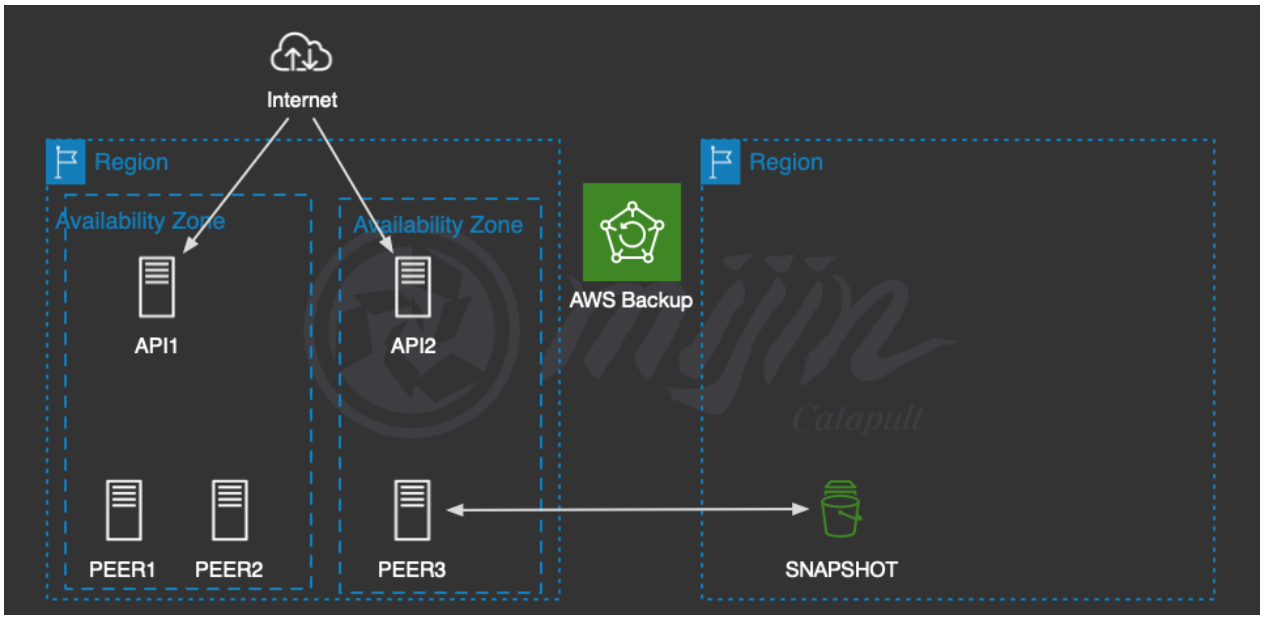
マルチリージョンにリカバリ戦略

リージョン間バックアップ

目標復旧時間 (RTO)	1 日
目標復旧時点 (RPO)	2 時間

Marketplace に展開する mijin Catapult(v.2) は、いずれかのノード一つからブロックチェーンデータをバックアップすることで全復旧することができます。
AWS Backup を使用することで、ノードにあるブロックチェーンデータを別リージョンに容易にバックアップすることが可能です。
バックアップ方法は、mijin Catapult(v.2) の定期的なノードのバックアップ を参照してください。
バックアップするまでの復元方法については、以下を参照してください。

https://docs.aws.amazon.com/ja_jp/aws-backup/latest/devguide/restore-resource.html



アクティブ/アクティブによるコンソーシアムチェーン

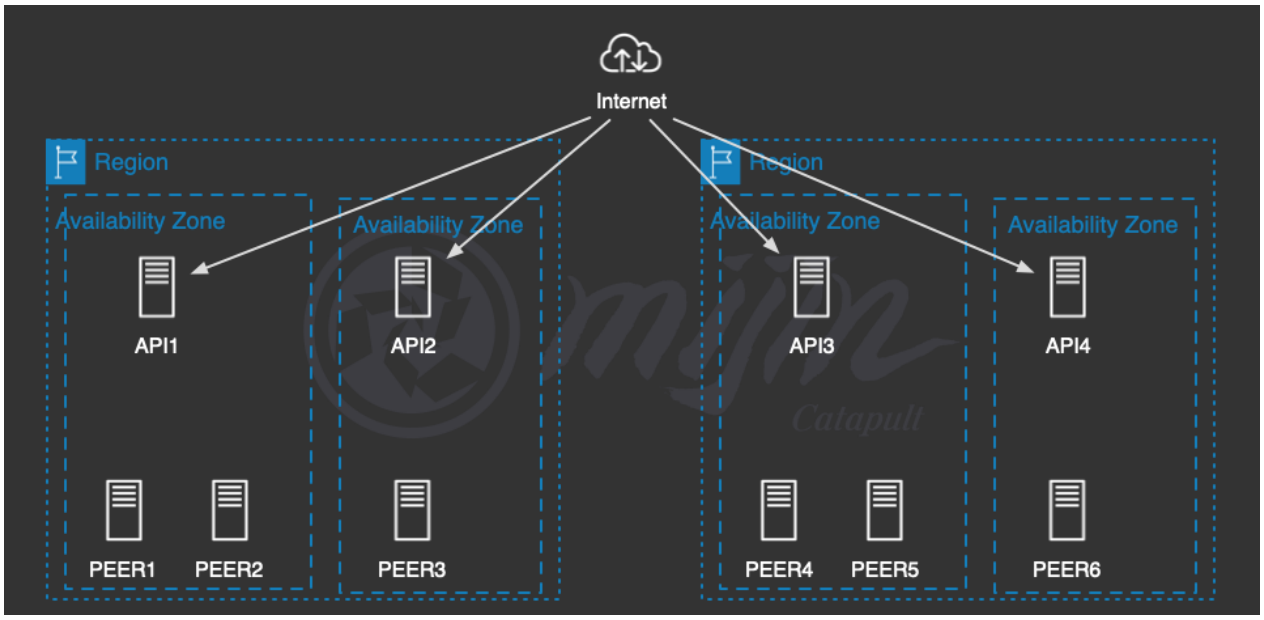
目標復旧時間 (RTO)	ほぼ 0
目標復旧時点 (RPO)	ほぼ 0

警告:

手動で、別リージョンに mijin を構築し、コンソーシアムチェーンを実現することで **0 ダウンタイム**を実現します。
以下図で例を示していますが、構築手順は環境によって異なるため、有償サポートによる支援も可能ですので、以下よりお問合せください。

https://mijin.io/aws_contact/

Marketplace に展開する mijin Catapult(v.2) は、世界 21 カ国のリージョンに設置することができますが、リージョン間でノードを構築する仕組みにはなっていません。
手動でメインリージョン以外に mijin ノードを構築することで、マルチリージョンによるディザスタリカバリ戦略を構築することができます。
API ノードも併せて設置することでコンソーシアムチェーンとなり、リージョン間でもアクティブ/アクティブ構成となります。



2.2.6 mijin Catapult(v.2) デプロイ後の AWS 設定

本章では、AWS MarketPlace にて提供している mijin Catapult(v.2) をデプロイ後、設定可能な方法を記載します。

2.2.6.1 mijin Catapult(v.2) EC2 インスタンスログイン方法

本章では、AWS 上の mijin Catapult(v.2) へのノードログイン方法を説明します。

mijin Catapult(v.2) は Linux サーバ上で動いているため、Linux ログイン方法の手順になりますが、AWS ではマネージメントコンソールから容易にリモートログインできる「Session Manager」があるため、デプロイ時に SessionManager でリモートログインできるように設定しています。
Session Manager について詳しく知りたい方は以下を参照してください。
https://docs.aws.amazon.com/ja_jp/systems-manager/latest/userguide/session-manager.html

注釈:

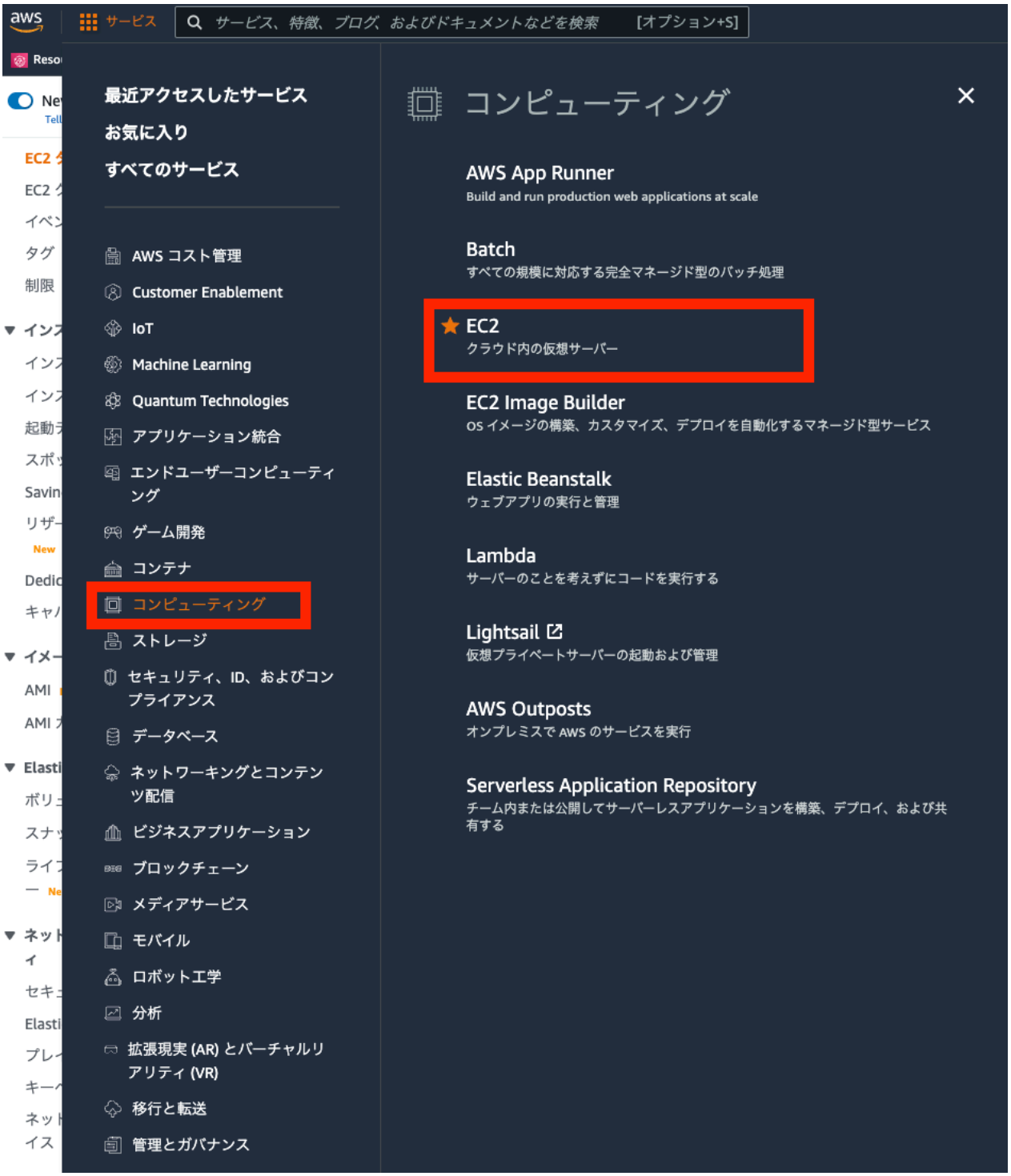
本章は、AWS MarketPlace にある mijin Catapult(v.2) をデプロイした場合の EC2 へのログイン手順の一例です。
接続方法は、SSH など従来のリモートログイン方法でもセキュリティグループなどを変更することで接続できます。

AWS マネージメントコンソールにログイン

AWS マネージドコンソールにて、ログインします。
https://aws.amazon.com/jp/console/

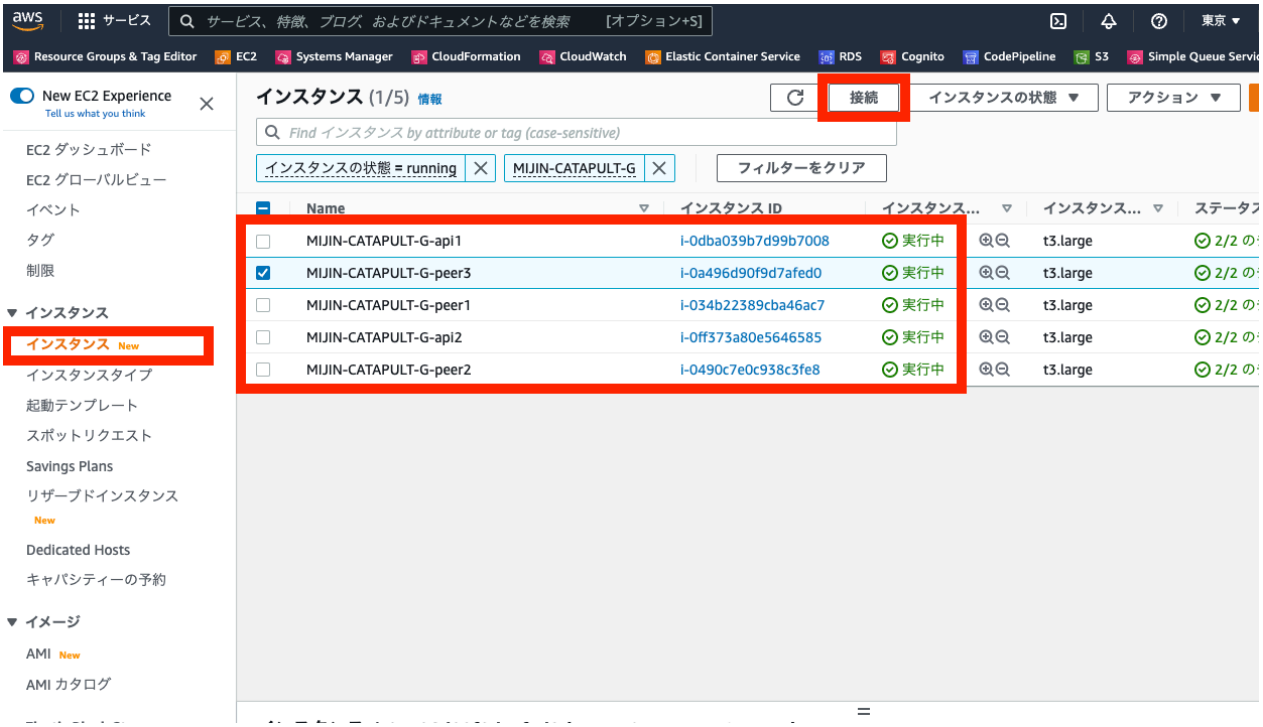
EC2 サービスに移動

1. 上部「サービス」をクリックします
2. 表示されたメニューから「コンピューティング」をクリックします
3. 「EC2」をクリックします。


aws
サービス
サービス、特徴、ブログ、およびドキュメントなどを検索
[オプション+S]
最近アクセスしたサービス
お気に入り
すべてのサービス
AWS コスト管理
Customer Enablement
IoT
Machine Learning
Quantum Technologies
アプリケーション統合
エンドユーザーコンピューティング
ゲーム開発
コンテナ
コンピューティング
ストレージ
セキュリティ、ID、およびコンプライアンス
データベース
ネットワーキングとコンテンツ配信
ビジネスアプリケーション
ブロックチェーン
メディアサービス
モバイル
ロボット工学
分析
拡張現実 (AR) とバーチャルリアリティ (VR)
移行と転送
管理とガバナンス
コンピューティング
AWS App Runner
Build and run production web applications at scale
Batch
すべての規模に対応する完全マネージド型のバッチ処理
EC2
クラウド内の仮想サーバー
EC2 Image Builder
OS イメージの構築、カスタマイズ、デプロイを自動化するマネージド型サービス
Elastic Beanstalk
ウェブアプリの実行と管理
Lambda
サーバーのことを考えずにコードを実行する
Lightsail
仮想プライベートサーバーの起動および管理
AWS Outposts
オンプレミスで AWS のサービスを実行
Serverless Application Repository
チーム内または公開してサーバーレスアプリケーションを構築、デプロイ、および共有する

ログインしたいインスタンスを選択し、接続

1. 左側メニューから「インスタンス」をクリックします
2. インスタンス一覧からログインしたいノードのチェックをクリックします。
3. 「接続」を押します。



セッションマネージャーを選択し、接続

1. 「セッションマネージャー」であることを確認します。(選択されていなければクリック)
2. 「接続」を押します。
3. コンソール画面のウィンドウが別で開きます。

コンソール画面の操作

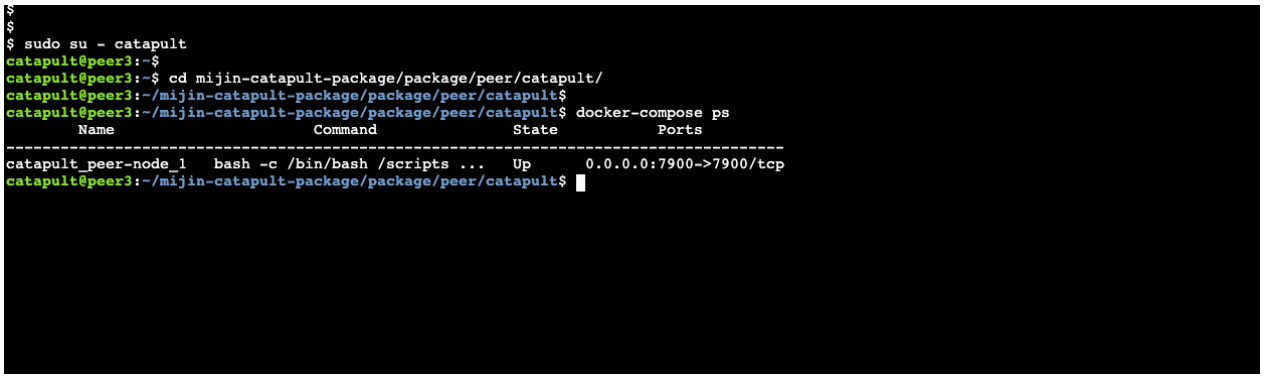
1. コンソール画面のウィンドウで　「$」が出ていることを確認します。
2. mijin Catapult(v.2) を動かしている UNIX ユーザー **catapult** にスイッチします。

```
sudo su - catapult
```

3. mijin Catapult(v.2) が動いているかを確認します。

```
# PEER ノードに接続した時 (本章では PEER ノード接続時) When connected to a PEER
node (in this chapter, when connected to a PEER node)
cd mijin-catapult-package/package/peer/catapult/
docker-compose ps

# API ノードに接続した時 When connected to an API node
cd mijin-catapult-package/package/api/catapult/
docker-compose ps
```



注釈:

mijin Catapult(v.2) は docker 上のコンテナの一つとして動いてます。
操作には docker の知識が必要になります。
Docker の知識は、以下のドキュメントを参照してください。
https://docs.docker.jp/
公式 (英語)
https://docs.docker.com/get-started/overview/

2.2.6.2 mijin Catapult(v.2) のノードストレージの暗号化

本章では、AWS 上の mijin Catapult(v.2) のノードで使用するストレージの暗号化を説明します。

デプロイ時点では、各ノードがマウントする EBS ボリュームは暗号化されていません。
より安全にするため、ブロックチェーンデータや Mongo のデータを暗号化することができます。
ここでは、PEER ノード 3 を例に、mijin Catapult(v.2) を格納している EBS ボリュームを暗号化する手順を説明します。

ノードのブロックチェーンデータを暗号化するまでの流れ

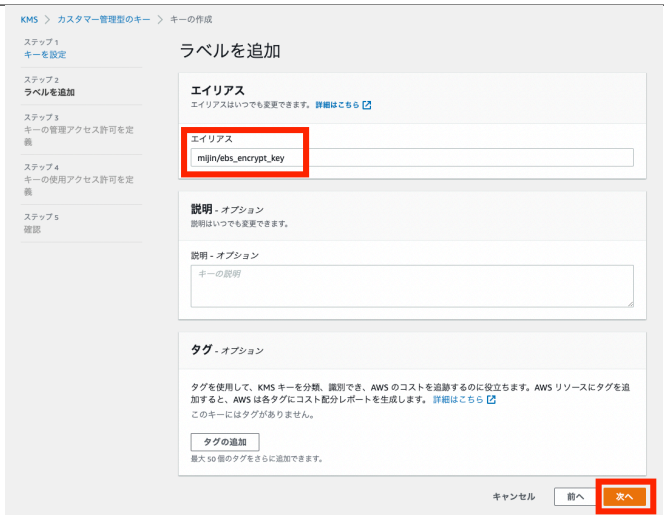
1. KMS Key を作成します
2. ノードの一台を停止する。(製品版では 1 台停止しても可用性は維持されます)
3. 停止したノードのスナップショットを取得する
4. 3 で作成したスナップショットをコピーして暗号化したスナップショットを作成する
5. 4 で作成した暗号化済みスナップショットから Volume を作成する。
6. 2 で停止した PEER ノードのブロックチェーンデータの Volume をデタッチする
7. 2 で停止した PEER ノードに 5 で作成した Volume をアタッチする
8. 2 で停止した PEER ノードを起動する。

注釈: API ノードでも同じ流れですが、API ノードの場合は、mongo データもマウントしているため、二つのボリュームを暗号化することができます。

KMS Key を作成

ストレージを暗号化するために、KMS を使用し、暗号鍵を作成します。
KMS による鍵作成について知りたい方は、以下を参照してください。
https://docs.aws.amazon.com/ja_jp/kms/latest/developerguide/create-keys.html

サービスから、セキュリティ、ID をクリックし、「Key Management Service」をクリックします。	
カスタマー管理型のキーをクリックし、「キーの作成」をクリックします。	

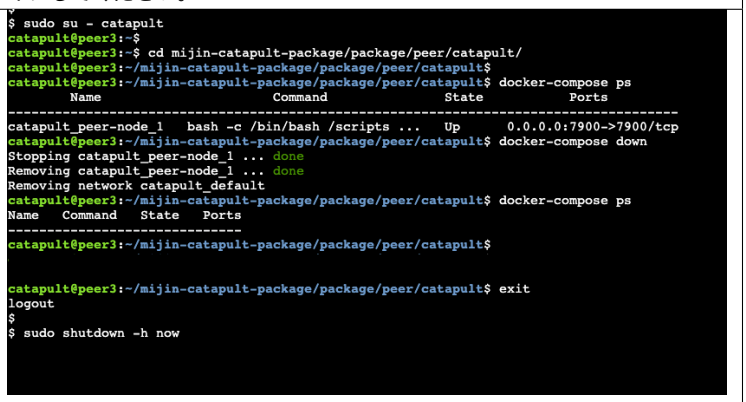
エイリアスに任意の名前を設定し、「次へ」をクリックします。	
キー名 (エイリアス) を指定し、「次へ」をクリックします。	
キー管理者を自分のアカウントを選択し、「次へ」をクリックします。(ここでは現在ログインしているアカウントを指定)	

キーを使用するアカウントを選択し、「次へ」をクリックします。(ここでは現在ログインしているアカウントを指定)	
値を確認し。「完了」をクリックします。	

ノードの一台を停止する

PEER ノード又は API ノードを一台停止しても、mijin Catapult(v.2) のブロックチェーンネットワークは停止しません。
ここでは、PEER ノード 3 を例に停止します。

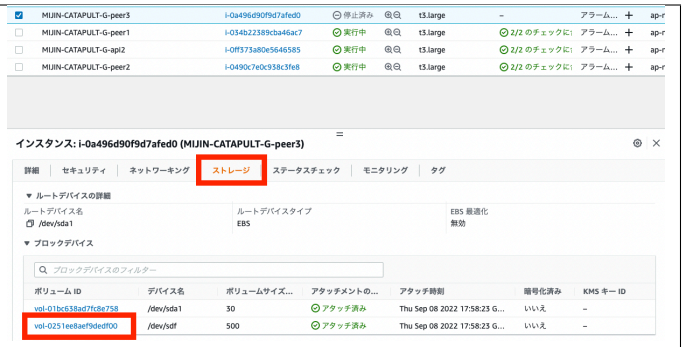
PEER ノード 3 にログインします。	mijin Catapult(v.2) EC2 インスタンスログイン方法 を参照し、ノードにログインしてください。

mijin Catapult(v.2) を停止し、EC2 インスタンスを停止します。
 	
`sudo su - catapult`
`cd mijin-catapult-package/`
`↪package/peer/catapult/`
`docker-compose down`
`docker-compose ps`
`exit`
`sudo shutdown -h now`
	

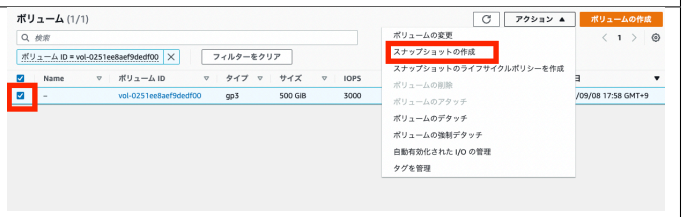
停止したノードのスナップショットを取得する

暗号化ボリュームを作成するには、まずは対象のボリュームのスナップショットを作成する必要があります。

停止した PEER ノード 3 を選択し、ストレージタブをクリックし、対象の VolumeId をクリックします。
ここで、VolumeId とデバイス名 (/dev/sdf) は控えておきます。



対象ボリュームをチェックをクリックし、「スナップショットの作成」をクリックします。



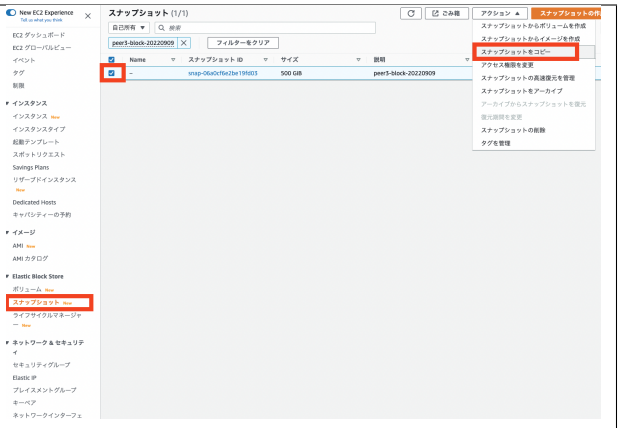
ここで、このボリュームがどのアベイラビリティゾーンに所属しているかを確認しておきます。

検索する時にわかりやすい名前を説明に記載し、「スナップショットの作成」をクリックします。

作成したスナップショットをコピーして暗号化したスナップショットを作成する

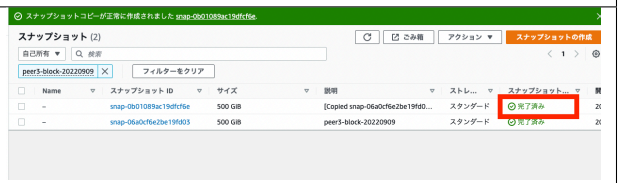
暗号化したスナップショットから暗号化ボリュームを作成するため、スナップショットをコピーする際に暗号化を行います。

メニューのスナップショットをクリックし、作成したスナップショットをチェックし、スナップショットをコピーします。



送信先リージョンはスナップショットを取得したノードと同じリージョンを選択します。
このスナップショットの暗号化をチェックし、作成した KMS を指定します。
「スナップショットをコピー」をクリックします。

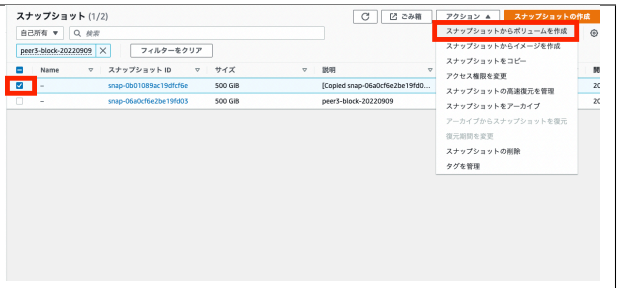
スナップショットが完了済みになっていることを確認してください。



暗号化済みスナップショットから Volume を作成する

暗号化したスナップショットから暗号化ボリュームを作成します。
この時点では、まだマウントされていないボリュームが作成されます。

暗号化したスナップショットをクリックし、「スナップショットからボリュームを作成」をクリックします。



アベイラビリティゾーンをスナップショットを取得したノードと同じアベイラビリティゾーンを選択します。
作成した KMS キーを選択します。
「ボリュームの作成」をクリックします。

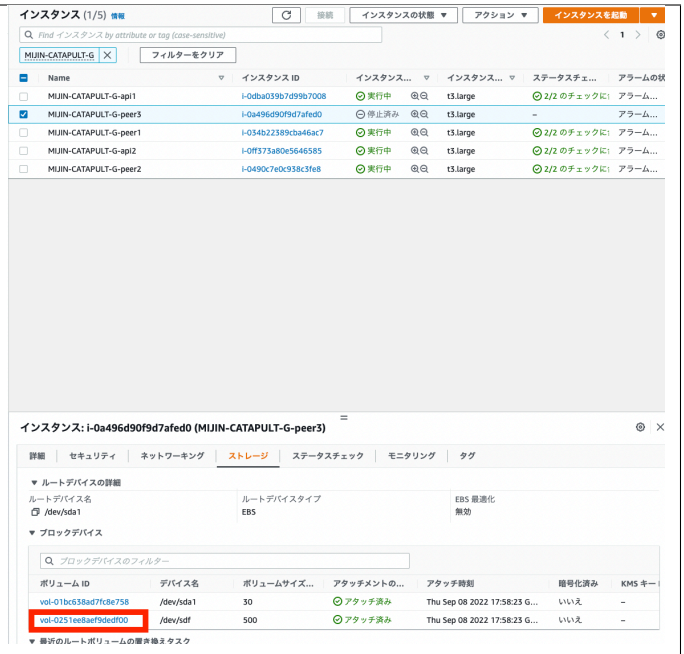
注釈:

ここで、ノードと違うアベイラビリティゾーンを選択すると、ノードにアタッチする際にボリュームが表示されないため、注意してください。
ノードのアベイラビリティゾーン確認は、停止したノードのスナップショットを取得する を確認してください。
アベイラビリティゾーンについて理解したい場合は、以下を参照してください。
https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/using-regions-availability-zones.html#concepts-availability-zones

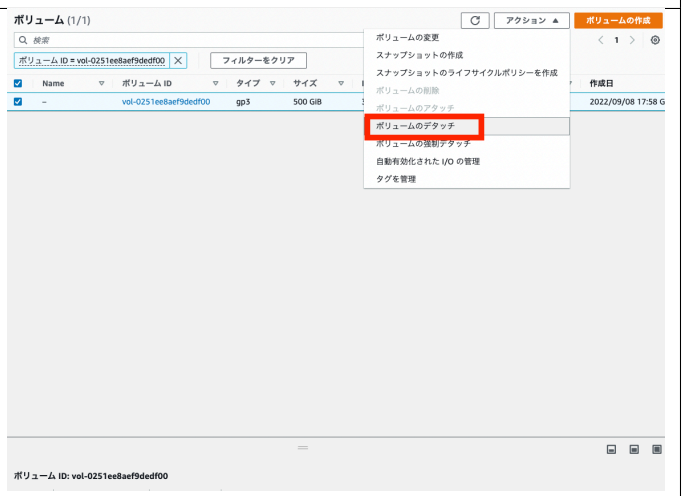
停止した PEER ノードのブロックチェーンデータの Volume をデタッチする

ノードにアタッチされている暗号化されていないボリュームを外します。

再度、PEER ノード 3 からボリュームを選択します。

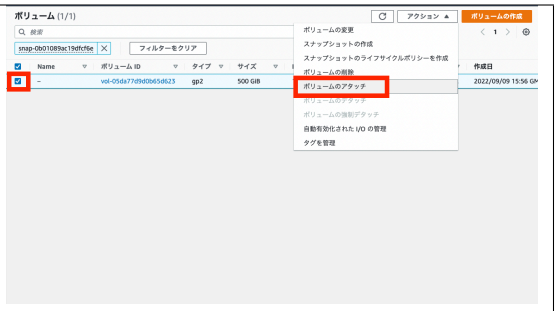
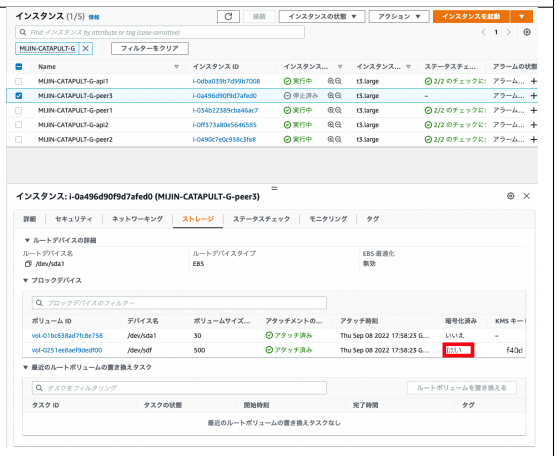


ボリュームから「ボリュームをデタッチ」をクリックし、ポップアップした画面で OK を押します。



PEER ノードに暗号化済み Volume をアタッチする

ノードに暗号化済みボリュームをアタッチします

暗号化済みボリュームを選択し、「ボリュームのアタッチ」をクリックします。	
インスタンスを PEER ノード 3 を選択し、控えておいたデバイス名を暗号化前のボリュームと同じパスを指定します。 「ボリュームのアタッチ」をクリックします。	
暗号化済みボリュームがアタッチされていることを確認します。	

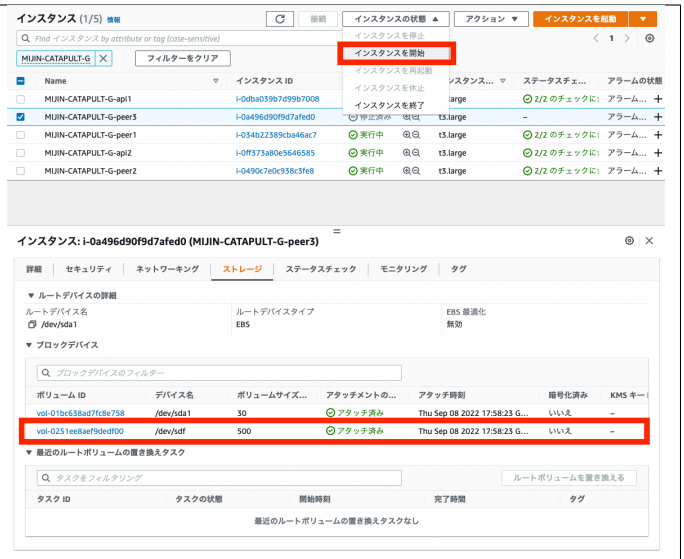
注釈: デバイス名は、停止したノードのスナップショットを取得する にて控えていたデバイス名を指定す

る必要があります。

停止した PEER ノードを起動する

停止していたノードを起動し、mijin Catapult(v.2) もあわせて起動します。

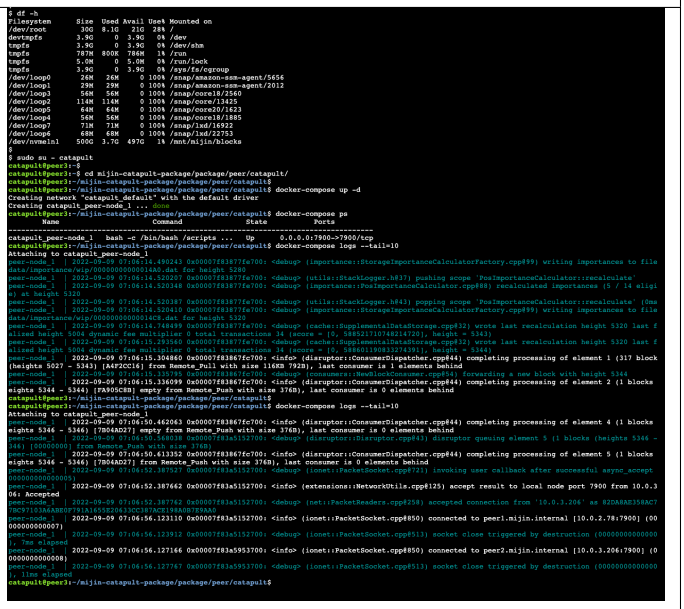
インスタンス一覧から、PEER ノード 3 をチェックし、「インスタンスの開始」をクリックします。



PEER ノード 3 にログインします。

mijin Catapult(v.2) EC2 インスタンスログイン方法 を参照し、ノードにログインしてください。

ディスクがマウントされていることを確認し、mijin Catapult(v.2) を起動します。




```
df -h
sudo su - catapult
cd mijin-catapult-package/package/
↪peer/catapult/
docker-compose up -d
docker-compose ps
```

2.2.6.3 mijin Catapult(v.2) の定期的なノードのバックアップ

本章では、AWS 上の mijin Catapult(v.2) のノードのデータをバックアップする手順を説明します。
ノードをバックアップすることで、万が一リージョン障害などがあった場合など、mijin Catapult(v.2) のブロックチェーンデータから復旧することができます。

AWS Backup サービスについて

EC2 インスタンス上で動いてる mijin Catapult(v.2) を容易にバックアップする「AWS Backup」を使用し、バックアップを定期的に実行します。
AWS Backup について知りたい方は、以下を参照してください。
https://docs.aws.amazon.com/ja_jp/aws-backup/latest/devguide/whatisbackup.html

バックアッププランの作成

説明	図
上タブにあるサービスをクリックします。 ストレージをクリックします。 「AWS Backup」サービスをクリックします。	
「バックアッププランの作成」をクリックします。	

[起動オプション] 新しいプランを立てるをチェックします。 バックアッププラン名を記載します。 **[バックアップルールの設定]** バックアップルール名を記載します。 バックアップウィンドウをカスタマイズをチェックします。 バックアップの開始時間を設定します（UTC 時間）。 コピー先にコピーで任意のリージョンを指定します（※1）。 「プランの作成」をクリックします。	
[全般] リソース割り当て名を記載します。 **[リソース選択]** 「特定のリソースタイプを含める」をチェックします。 リソースタイプ：EC2 インスタンス ID：ここでは PEER3 ノードを選択します。 「リソース選択」をクリックします。	

注釈:

バックアップするノード (EC2 インスタンス) を複数指定してバックアップすることはできますが、mijin Catapult(v.2) は１つのノードにあるブロックチェーンデータから全てを復旧できるため、定期的なバックアップは一つのノードだけバックアップすれば問題なく、コストを抑えることができます。

※1 コピー先のリージョンを同一リージョンにすると、バックアップしたスナップショットからリストアする にて復旧することができます。
別リージョンにすることで、ディザスタリカバリなどの対災害対策とすることもできます。

2.2.6.4 mijin Catapult(v.2) 手数料ありモード有効時の残高アカウントの残高移動

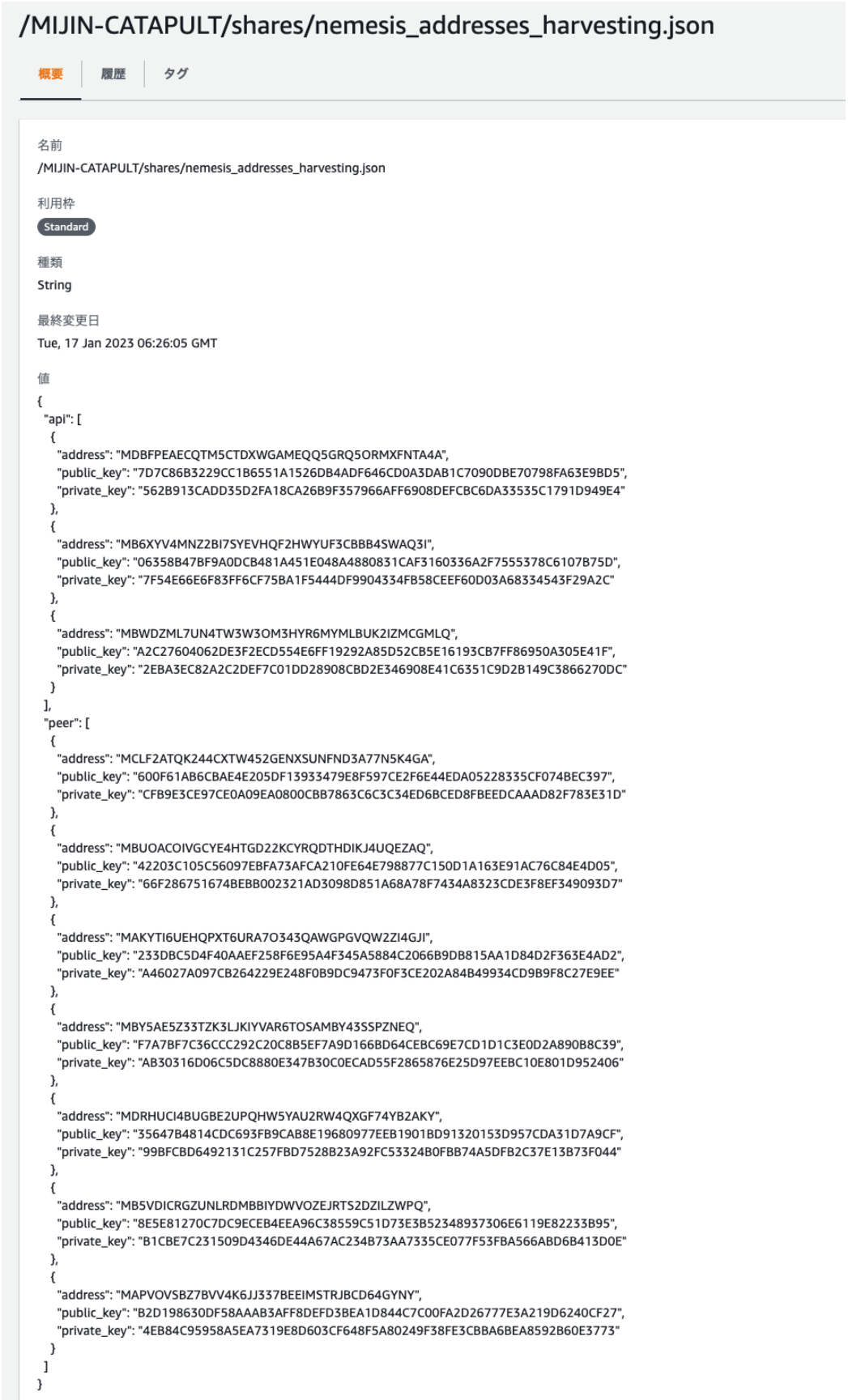
本章では、AWS 上の mijin Catapult(v.2) ノードの残高を持つアカウントから別のアカウントに残高を移動する方法を説明します。
移動したアカウントを mijin Catapult(v.2) の操作を行うアカウントとして推奨します。

注釈:

ブロックチェーンにおいて、基軸通貨は必ずあり、ブロックチェーンを操作する上でトランザクション手数料を支払う必要があります。
mijin Catapult(v.2) でも手数料ありモードを有効にすると、トランザクション作成に手数料が必要となり、トランザクション手数料、Mosaic レンタル手数料など全てに基軸通貨の残高を持っているアカウントで操作する必要があります。
逆に、手数料なしモードでは、残高を持ってないアカウントを使ってトランザクションを作成することができます。

残高あるアカウントの確認

残高あるアカウントは、AWS Systems Manager パラメータストアにある、
nemesis_addresses_harvesting.json に保存されます。
このファイルのアカウント一覧は、各ノードにすべて紐づいています。



/MIJIN-CATAPULT/shares/nemesis_addresses_harvesting.json

概要 | 履歴 | タグ

名前

/MIJIN-CATAPULT/shares/nemesis_addresses_harvesting.json

利用枠

Standard

種類

String

最終変更日

Tue, 17 Jan 2023 06:26:05 GMT

値


```
{
  "api": [
    {
      "address": "MDBFPEAECQTM5CTDXWGAMEQQ5GRQ5ORMXFNTA4A",
      "public_key": "7D7C86B3229CC1B6551A1526DB4ADF646CD0A3DAB1C7090DBE70798FA63E9BD5",
      "private_key": "562B913CADD35D2FA18CA26B9F357966AFF6908DEFCBC6DA33535C1791D949E4"
    },
    {
      "address": "MB6XYV4MNZ2BI7SYEVHQF2HWYUF3CBBB4SWAQ3I",
      "public_key": "06358B47BF9A0DCB481A451E048A4880831CAF3160336A2F7555378C6107B75D",
      "private_key": "7F54E66E6F83FF6CF75BA1F5444DF9904334FB58CEEF60D03A68334543F29A2C"
    },
    {
      "address": "MBWDZML7UN4TW3W3OM3HYR6MYMLBUK2IZMCGMLQ",
      "public_key": "A2C27604062DE3F2ECD554E6FF19292A85D52CB5E16193CB7FF86950A305E41F",
      "private_key": "2EBA3EC82A2C2DEF7C01DD28908CBD2E346908E41C6351C9D2B149C3866270DC"
    }
  ],
  "peer": [
    {
      "address": "MCLF2ATQK244CXTW452GENXSUNFND3A77N5K4GA",
      "public_key": "600F61AB6CBAE4E205DF13933479E8F597CE2F6E44EDA05228335CF074BEC397",
      "private_key": "CFB9E3CE97CE0A09EA0800CBB7863C6C3C34ED6BCED8FBEEDCAAAD82F783E31D"
    },
    {
      "address": "MBUOACOIVGCYE4HTGD22KCYRQDTHDIKJ4UQEZAQ",
      "public_key": "42203C105C56097EBFA73AFCA210FE64E798877C150D1A163E91AC76C84E4D05",
      "private_key": "66F286751674BEBB002321AD3098D851A68A78F7434A8323CDE3F8EF349093D7"
    },
    {
      "address": "MAKYTI6UEHQPXT6URA7O343QAWGPGVQW2ZI4GJI",
      "public_key": "233DBC5D4F40AAEF258F6E95A4F345A5884C2066B9DB815AA1D84D2F363E4AD2",
      "private_key": "A46027A097CB264229E248F0B9DC9473F0F3CE202A84B49934CD9B9F8C27E9EE"
    },
    {
      "address": "MBY5AE5Z33TZK3LJKIYVAR6TOSAMBY43SSPZNEQ",
      "public_key": "F7A7BF7C36CCC292C20C8B5EF7A9D166BD64CEBC69E7CD1D1C3E0D2A890B8C39",
      "private_key": "AB30316D06C5DC8880E347B30C0ECAD55F2865876E25D97EEBC10E801D952406"
    },
    {
      "address": "MDRHUCI4BUGBE2UPQHW5YAU2RW4QXGF74YB2AKY",
      "public_key": "35647B4814CDC693FB9CAB8E19680977EEB1901BD91320153D957CDA31D7A9CF",
      "private_key": "99BFCBD6492131C257FBD7528B23A92FC53324B0FBB74A5DFB2C37E13B73F044"
    },
    {
      "address": "MB5VDICRGZUNLRDMBBIYDWVOZEJRTS2DZILZWPQ",
      "public_key": "8E5E81270C7DC9ECEB4EEA96C38559C51D73E3B52348937306E6119E82233B95",
      "private_key": "B1CBE7C231509D4346DE44A67AC234B73AA7335CE077F53FBA566ABD6B413D0E"
    },
    {
      "address": "MAPVOVSBZ7BVV4K6JJ337BEEIMSTRJBCD64GYNY",
      "public_key": "B2D198630DF58AAAB3AFF8DEFD3BEA1D844C7C00FA2D26777E3A219D6240CF27",
      "private_key": "4EB84C95958A5EA7319E8D603CF648F5A80249F38FE3CBBA6BEA8592B60E3773"
    }
  ]
}
```

すべてのアカウントに残高はありますが、ここでは一番上の以下のデータを mijin-catapult-cli を使用して確認します。

```
{
  "address": "MDBFPEAECQTM5CTDXWGAMEQQ5GRQ5ORMXFNTA4A",
  "public_key": "7D7C86B3229CC1B6551A1526DB4ADF646CD0A3DAB1C7090DBE70798FA63E9BD5",
  "private_key": "562B913CADD35D2FA18CA26B9F357966AFF6908DEFCBC6DA33535C1791D949E4"
},
```

準備 nodejs 及び yarn をインストール

mijin-catapult-tools を使用するため nodejs をインストールします。nodejs は NodeSource を利用してインストールします。

```
$ curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash - && sudo apt-get
install -y nodejs
$ node --version
$ sudo npm install -g yarn
```

mijin-catapult-tools のインストール

yarn を使用し、mijin-catapult-tools をインストールします。

```
$ yarn global add  @tech-bureau/mijin-catapult-tools
$ echo 'export PATH="$HOME/.yarn/bin:$PATH"' >> ~/.bashrc && source ~/.bashrc
```

残高アカウントのインポート

残高アカウントをインポートします。
-u では、CloudFormation の出力にある mijinLBENdpoint 又は mijinEndpoint の URL を指定します。
-p では残高アカウントの private_key を指定します。

```
$ mijin-catapult-tools account generate -w aws.json -u http://xxxxxxxxxxxxxxxxxxxx.
↪elb.ap-northeast-1.amazonaws.com:3000 -s -p
562B913CADD35D2FA18CA26B9F357966AFF6908DEFCBC6DA33535C1791D949E4
2023-01-17T07:35:12.266Z [info] : mijin URL: http://xxxxxxxxxxxxxxxxxxxx.elb.ap-
↪northeast-1.amazonaws.com:3000
2023-01-17T07:35:12.268Z [info] : Network: 96
2023-01-17T07:35:12.268Z [info] : Mosaic Currency Id: 769E11974E2CAD76
2023-01-17T07:35:12.269Z [info] : Mosaic Harvest Id: 792022E7945425E4
2023-01-17T07:35:12.269Z [info] : Start Account Generate...
2023-01-17T07:35:12.343Z [info] : Write Config File: aws.json
2023-01-17T07:35:12.345Z [info] : New Account: {
  "url": "http://xxxxxxxxxxxxxxxxxxxx.elb.ap-northeast-1.amazonaws.com:3000",
  "workAccount": {
    "publicKey": "E28BF2A27FE64DF392CBF6D6883BE4858CF26790B4EABC3BCA8E08854BC6A9BF",
    "privateKey": "C3BE65EB9055405ED8CCB7AD568D1368067174F969E9321C0AB4379A7565C9C2",
    "address": "MB2ZQXQQQHOVYU4GX2TKNNJK6XLZPIZV6LK62MY"
```

(continues on next page)

(continued from previous page)


```
  },
  "balanceAccount": {
    "publicKey": "7D7C86B3229CC1B6551A1526DB4ADF646CD0A3DAB1C7090DBE70798FA63E9BD5",
    "privateKey": "562B913CADD35D2FA18CA26B9F357966AFF6908DEFCBC6DA33535C1791D949E4",
    "address": "MDBFPEAECQTM5CTDXWGAMEQQ5GRQ5ORMXFNTA4A"
  },
  "mainAccount": {
    "publicKey": "DAF95081E2D816062108424CF2404B9C3B7C4C7CD1DF6E1446158CC2A2D9B29B",
    "privateKey": "ACF704D53457DF418E1784BACB6D0977B2626BC2847506C7D91B39CDD0515F0F",
    "address": "MD4J2ZVP2AW3BCH6UKZOJNNBGS35DBKTNS4KGVI"
  },
  "keylink": {
    "vrf": {
      "publicKey": "E3D05474D23B57EEFCF953EB7A1AF7A44F9BA338C83900AB0A72927D933CD56A",
      "privateKey": "D0F48B1926ECA6C32C1D3A61AE25A7A483EF47B6DE96B9B4E4970773F904CB73",
      "address": "MBR5NJKKF66GVIOFQKCJOAOTR5KWFE5AKWTVLOQ"
    },
    "voting": {
      "publicKey": "8A615FD6E66CBCB6361FAE0156CB6E22E0932F7157F04849551FCAE9CC6E494F",
      "privateKey": "1615F3484A2123131A624FA065E4B17A0CE916FF6D5A772501FBC83876B3B912",
      "address": "MDWPSE2HLW4HFFTLSECCIMWQPF6IKFPQAI76V3Y"
    }
  },
  "test1Account": {
    "publicKey": "889E1705185A2138F4408D70C28A015536F05A69185392F8C683BB39A0BDB951",
    "privateKey": "8636D903270275A3A6459B41E73E8E7365A6A55ED2437AB25F3321230FD64C35",
    "address": "MBRU2UKYC5C7J6MNQU7F3KYXFVMSIUKUZTPWFHI"
  },
  "test2Account": {
    "publicKey": "24323D2D3594BF0A1993E018571EBD1175BD4B461EF3D15FE4EB09FADAB95834",
    "privateKey": "2A4F1B1E98BF4F91F6FE6D6F90844D9FCA78BCF198028AD8778CC85C409F6B5E",
    "address": "MAWOT4JSX3OYBMNGRM47ZQDDUHCJ3LLQKKO6RKA"
  }
}
```


アカウントの情報を確認します。
"currency": true となっている Mosaic Id **769E11974E2CAD76** が基軸通貨になります。
"harvest": true となっている通貨は、ブロック生成する権利があるノードでは所有が必要になります。

警告: 基軸通貨の Mosaic Id はデプロイ時に作成されるため、同じ ID でないことに注意してください。

```
$ mijin-catapult-tools account info -r aws.json -t balance
2023-01-17T07:36:16.432Z [info] : mijin URL: http://xxxxxxxxxxxxxxxxxxxx.elb.ap-
↪northeast-1.amazonaws.com:3000
2023-01-17T07:36:16.433Z [info] : Network: 96
2023-01-17T07:36:16.433Z [info] : Mosaic Currency Id: 769E11974E2CAD76
2023-01-17T07:36:16.434Z [info] : Mosaic Harvest Id: 792022E7945425E4
2023-01-17T07:36:16.434Z [info] : Start Account Info
2023-01-17T07:36:16.603Z [info] : balance Account: {
  "publicKey": "7D7C86B3229CC1B6551A1526DB4ADF646CD0A3DAB1C7090DBE70798FA63E9BD5",
```

(continues on next page)

(continued from previous page)


```
  "address": "MDBFPEAECQTM5CTDXWGAMEQQ5GRQ5ORMXFNTA4A",
  "mosaics": [
    {
      "id": "769E11974E2CAD76",
      "amount": "1799799999600000",
      "currency": true,
      "harvest": false
    },
    {
      "id": "792022E7945425E4",
      "amount": "3000000",
      "currency": false,
      "harvest": true
    }
  ],
  "keylink": {
    "vrf": {
      "publicKey": "9CD207F9A6DE6D485D350C29B749590251924A29C0EFD8E38DDE24866D71F160"
    },
    "voting": {
      "publicKey": "E3822AA0720F610847E4BE2B740F8FFF9130BEC4E9140BA845150DC2D591D86D",
      "startEpoch": 1,
      "endEpoch": 26280
    }
  }
}
```


新規アカウントの作成

残高がないアカウントを新規作成します。
アドレス **MDJMNVU47CWHTZBMX7B6M6WWT5NEEY4GTG66GLQ** に後ほど残高を移動します。

```
$ mijin-catapult-tools account generate -r aws.json
2023-01-17T07:38:41.738Z [info] : mijin URL: http://xxxxxxxxxxxxxxxxxxxx.elb.ap-
↪northeast-1.amazonaws.com:3000
2023-01-17T07:38:41.740Z [info] : Network: 96
2023-01-17T07:38:41.740Z [info] : Mosaic Currency Id: 769E11974E2CAD76
2023-01-17T07:38:41.740Z [info] : Mosaic Harvest Id: 792022E7945425E4
2023-01-17T07:38:41.740Z [info] : Start Account Generate...
2023-01-17T07:38:41.768Z [info] : New Account: {
  "publicKey": "7437EB45A39AF335F08CABD203503632115CA1793902F5106BC03963C96AEE4F",
  "privateKey": "708AB4973F37B89195340AEA7EBD733ED16AE51B99EB648E7A3885869CBAF3C9",
  "address": "MDJMNVU47CWHTZBMX7B6M6WWT5NEEY4GTG66GLQ"
```

アカウントの情報をノードに問い合わせます。
このアカウントは、ノードに残高の記録がないため、存在しないエラーになっていることを確認してください。

```
$ mijin-catapult-tools account info -r aws.json -t other -a
MDJMNVU47CWHTZBMX7B6M6WWT5NEEY4GTG66GLQ
2023-01-17T07:39:44.832Z [info] : mijin URL: http://xxxxxxxxxxxxxxxxxxxx.elb.ap-
↪northeast-1.amazonaws.com:3000
2023-01-17T07:39:44.834Z [info] : Network: 96
2023-01-17T07:39:44.834Z [info] : Mosaic Currency Id: 769E11974E2CAD76
2023-01-17T07:39:44.834Z [info] : Mosaic Harvest Id: 792022E7945425E4
2023-01-17T07:39:44.834Z [info] : Start Account Info
2023-01-17T07:39:45.061Z [error] : Address Not Found
```

残高の移動

残高がある balance アカウントから新規アカウント
(MDJMNVU47CWHTZBMX7B6M6WWT5NEEY4GTG66GLQ) に残高を転送します。
ここでは 10 万 cat.currency 転送してみます。

残高アカウント (mijin-have-currency-account) から転送トランザクションをアナウンスします。

```
$ mijin-catapult-tools transaction transfer -r aws.json -f balance -d
MDJMNVU47CWHTZBMX7B6M6WWT5NEEY4GTG66GLQ -a 100000
2023-01-17T07:41:40.559Z [info] : mijin URL: http://xxxxxxxxxxxxxxxxxxxx.elb.ap-
↪northeast-1.amazonaws.com:3000
2023-01-17T07:41:40.562Z [info] : Network: 96
2023-01-17T07:41:40.562Z [info] : Start Transfer Account...
2023-01-17T07:41:40.585Z [info] : From Account Address:
MDBFPEAECQTM5CTDXWGAMEQQ5GRQ5ORMXFNTA4A
2023-01-17T07:41:40.585Z [info] : Dest Account Address:
MDJMNVU47CWHTZBMX7B6M6WWT5NEEY4GTG66GLQ
2023-01-17T07:41:40.585Z [info] : Currecny Amount: 100000000000
2023-01-17T07:41:40.594Z [info] : Start Transfer Transaction...
2023-01-17T07:41:55.775Z [info] : End Transfer Transaction
2023-01-17T07:41:55.775Z [info] : http://xxxxxxxxxxxxxxxxxxxx.elb.ap-northeast-1.
↪amazonaws.com:3000/transactionStatus/
↪EC5FE12DBEFD1DF7DDE2D49287EC4DA1649546BB1EC43DE75641D5D4A7BEE770
2023-01-17T07:41:55.775Z [info] : http://xxxxxxxxxxxxxxxxxxxx.elb.ap-northeast-1.
↪amazonaws.com:3000/transactions/confirmed/
↪EC5FE12DBEFD1DF7DDE2D49287EC4DA1649546BB1EC43DE75641D5D4A7BEE770
```

新規アカウントに残高があることを確認します。
先ほどエラーになっていたアカウント情報が mijin Catapult(v.2) に認識され、残高を持っていることを確認できます。

```
$ mijin-catapult-tools account info -r aws.json -t other -a
MDJMNVU47CWHTZBMX7B6M6WWT5NEEY4GTG66GLQ
2023-01-17T07:42:26.802Z [info] : mijin URL: http://xxxxxxxxxxxxxxxxxxxx.elb.ap-
↪northeast-1.amazonaws.com:3000
2023-01-17T07:42:26.803Z [info] : Network: 96
2023-01-17T07:42:26.804Z [info] : Mosaic Currency Id: 769E11974E2CAD76
```

(continues on next page)

(continued from previous page)

```
2023-01-17T07:42:26.804Z [info] : Mosaic Harvest Id: 792022E7945425E4
2023-01-17T07:42:26.804Z [info] : Start Account Info
2023-01-17T07:42:26.927Z [info] : get Account: {
  "publicKey": "0000000000000000000000000000000000000000000000000000000000000000",
  "address": "MDJMNVU47CWHTZBMX7B6M6WWT5NEEY4GTG66GLQ",
  "mosaics": [
    {
      "id": "769E11974E2CAD76",
      "amount": "100000000000",
      "currency": true,
      "harvest": false
    }
  ],
  "keylink": {
    "vrf": {
      "publicKey": ""
    },
    "voting": {
      "publicKey": "",
      "startEpoch": "",
      "endEpoch": ""
    }
  }
}
```

2.2.6.5 mijin Catapult(v.2) ノードの投票権ファイルの更新方法

本章では、AWS 上の mijin Catapult(v.2) ノードにあるノード投票権の期限が切れた場合の対応方法を記載します。
ノードの投票権の期限は、ブロック生成間隔によって変動しますが、約 547 日 ~3285 日となっています。

警告:

投票権ファイルの期限が切れると、ファイナライズブロックが停止します。
そのため、ファイナライズブロックを使用している場合は必ず更新する必要があります。

投票権ノードの有効期限計算方法

一つの投票権ファイルの期限は以下の計算式で求めることができます。

警告: blockGenerationTargetTime(ブロック生成間隔) は例えば 15s を設定しても一定時間でブロック生成するわけではないため、目安として値となることに注意してください。

```
(VotingSetGroup * maxVotingKeyLifetime) / ( 60 / blockGenerationTargetTime * 60 * 24)
```

VotingSetGroup 及び maxVotingKeyLifetime は固定値となるため、blockGenerationTargetTime の値によって
有効期限が変動します。

```
# blockGenerationTargetTime 10s
(180 * 26280) / (60 /10 * 60 * 24) = 547 days
# blockGenerationTargetTime 15s
(180 * 26280) / (60 /15 * 60 * 24) = 821 days
# blockGenerationTargetTime 60s
(180 * 26280) / (60 /60 * 60 * 24) = 3285 days
```

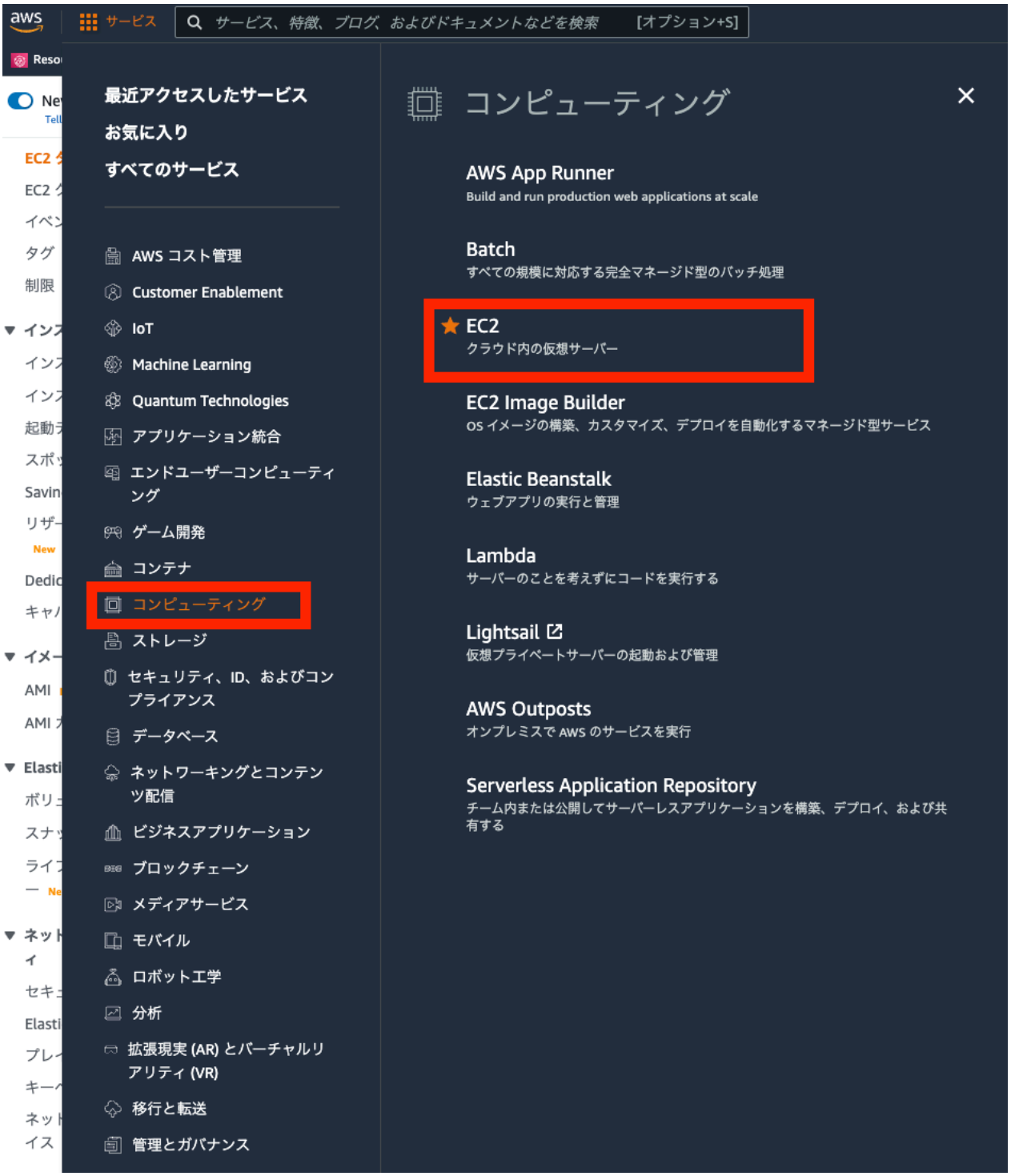
AWS マネージメントコンソールにログイン

AWS マネージドコンソールにて、ログインします。

https://aws.amazon.com/jp/console/

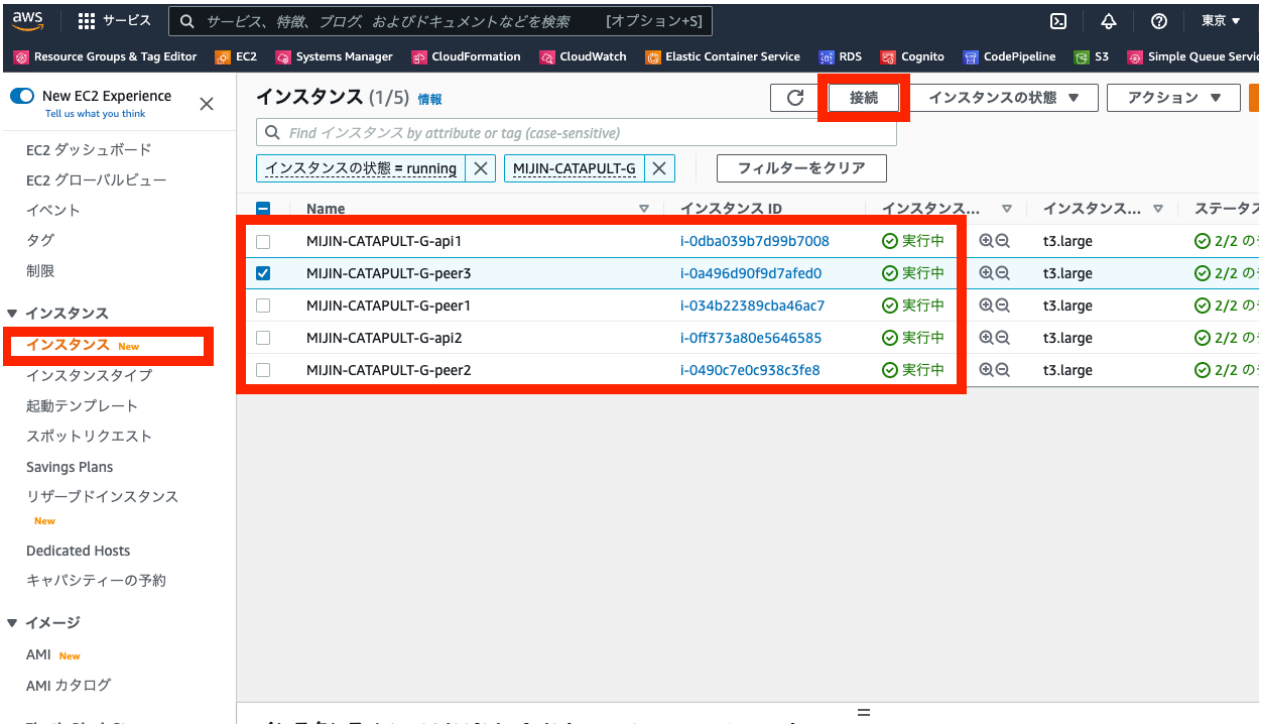
EC2 サービスに移動

1. 上部「サービス」をクリックします
2. 表示されたメニューから「コンピューティング」をクリックします
3.「EC2」をクリックします。


aws
サービス
サービス、特徴、ブログ、およびドキュメントなどを検索
[オプション+S]
最近アクセスしたサービス
お気に入り
すべてのサービス
AWS コスト管理
Customer Enablement
IoT
Machine Learning
Quantum Technologies
アプリケーション統合
エンドユーザーコンピューティング
ゲーム開発
コンテナ
コンピューティング
ストレージ
セキュリティ、ID、およびコンプライアンス
データベース
ネットワーキングとコンテンツ配信
ビジネスアプリケーション
ブロックチェーン
メディアサービス
モバイル
ロボット工学
分析
拡張現実 (AR) とバーチャルリアリティ (VR)
移行と転送
管理とガバナンス
コンピューティング
AWS App Runner
Build and run production web applications at scale
Batch
すべての規模に対応する完全マネージド型のバッチ処理
EC2
クラウド内の仮想サーバー
EC2 Image Builder
OS イメージの構築、カスタマイズ、デプロイを自動化するマネージド型サービス
Elastic Beanstalk
ウェブアプリの実行と管理
Lambda
サーバーのことを考えずにコードを実行する
Lightsail
仮想プライベートサーバーの起動および管理
AWS Outposts
オンプレミスで AWS のサービスを実行
Serverless Application Repository
チーム内または公開してサーバーレスアプリケーションを構築、デプロイ、および共有する

ログインしたいインスタンスを選択し、接続

1. 左側メニューから「インスタンス」をクリックします
2. インスタンス一覧からログインしたいノードのチェックをクリックします。
3.「接続」を押します。



セッションマネージャーを選択し、接続

1.「セッションマネージャー」であることを確認します。(選択されていなければクリック)
2.「接続」を押します。
3. コンソール画面のウィンドウが別で開きます。

コンソール画面の操作

1. コンソール画面のウィンドウで　「$」が出ていることを確認します。

```
$
```

mijin-catapult-tools のインストール

1. nodejs がインストールされてない場合、インストールします。

```
$ curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash - && sudo
apt-get install -y nodejs
```

2. catapult ユーザーにスイッチし、mijin-catapult-tools をインストールします。

```
$ yarn global add  @tech-bureau/mijin-catapult-tools
$ echo 'export PATH="$HOME/.yarn/bin:$PATH"' >> ~/.bashrc && source ~/.
↪bashrc
```

現在の投票権ファイルの状況確認

公開鍵を控えます。
ここでは、**402B6ECE0D1CF99A7F07B832477048C56F213A4F54ED4AEB35AE829507FBC4A6** です。
この投票権ファイルは 1 から 26280 までの有効期限となっていることが確認できます。

```
$ mijin-catapult-tools votingkey info -u http://localhost:3000 -d mijin-
↪catapult-package/package/api/catapult/userconfig/resources/votingkey
2023-01-26T05:22:29.833Z [info] : Start Voting Key Check Dir... mijin-
↪catapult-package/package/api/catapult/userconfig/resources/votingkey
2023-01-26T05:22:29.836Z [info] : votingfile: index0:  {"publicKey":
↪"402B6ECE0D1CF99A7F07B832477048C56F213A4F54ED4AEB35AE829507FBC4A6",
↪"startEpoch":1,"endEpoch":26280,"filename":"private_key_tree1.dat"}
```

注釈:

finalizationEpoch が 26280 になる前に更新する必要があるということに注意してください。
$ curl -Ss http://localhost:3000/chain/info | jq -r
{
"scoreHigh": "0",
"scoreLow": "16875391960469924",
"height": "310",
"latestFinalizedBlock": {
"finalizationEpoch": 3,
"finalizationPoint": 7,
"height": "296",
"hash": "AC19CA6C89F87F70470BD84649A31E4FA0E0C5DD71A55E9ADAE25C1AED47882F"
}
}

投票権ファイルの更新

上記で取得した公開鍵から秘密鍵を検索し、投票権ファイルを作成します。
連番ファイル private_key_tree2.dat が作成されます。

```
$ mijin-catapult-tools votingkey update \
    -u http://localhost:3000 \
    -d mijin-catapult-package/package/api/catapult/userconfig/resources/
↪votingkey \
    -p $(cat /mnt/mijin/shares/nemesis_addresses_harvesting_voting.json | jq -
↪r '.[]|.[]|select(.public_key ==
"402B6ECE0D1CF99A7F07B832477048C56F213A4F54ED4AEB35AE829507FBC4A6")|.private_
↪key')

2023-01-26T06:06:16.472Z [info] : Start Voting Key Update...
2023-01-26T06:06:16.476Z [info] : votingSetGroup: 160
2023-01-26T06:06:16.476Z [info] : votingMaxEpoch: 26280
2023-01-26T06:06:16.477Z [info] : votingStartEpoch: 26281
```

(continues on next page)

(continued from previous page)

```
2023-01-26T06:06:16.477Z [info] : votingEndEpoch: 52560
2023-01-26T06:06:16.477Z [info] : blockGenerationTargetTime: 15
2023-01-26T06:06:42.716Z [info] : Voting Key file Create: SUCCESS mijin-
↪catapult-package/package/api/catapult/userconfig/resources/votingkey/
↪private_key_tree2.dat
```

再度、投票権ファイルの状況確認

同じ公開鍵で作成された投票権ファイル (private_key_tree2.dat) が確認できていれば更新ファイルの作成は完了です。
次回は finalizationEpoch が 52560 になる前に、更新する必要があります。

```
$ mijin-catapult-tools votingkey info -u http://localhost:3000 -d mijin-
↪catapult-package/package/api/catapult/userconfig/resources/votingkey
2023-01-26T06:10:00.625Z [info] : Start Voting Key Check Dir... mijin-
↪catapult-package/package/api/catapult/userconfig/resources/votingkey
2023-01-26T06:10:00.630Z [info] : votingfile: index0:  {"publicKey":
↪"402B6ECE0D1CF99A7F07B832477048C56F213A4F54ED4AEB35AE829507FBC4A6",
↪"startEpoch":1,"endEpoch":26280,"filename":"private_key_tree1.dat"}
2023-01-26T06:10:00.630Z [info] : votingfile: index1:  {"publicKey":
↪"402B6ECE0D1CF99A7F07B832477048C56F213A4F54ED4AEB35AE829507FBC4A6",
↪"startEpoch":26281,"endEpoch":52560,"filename":"private_key_tree2.dat"}
```

2.2.6.6 [アーカイブ] mijin Catapult(v.2) 手数料ありモード有効時の残高アカウントの残高移動

本章では、AWS 上の mijin Catapult(v.2) ノードの残高を持つアカウントから別のアカウントに残高を移動する方法を説明します。
移動したアカウントを mijin Catapult(v.2) の操作を行うアカウントとして推奨します。

警告:

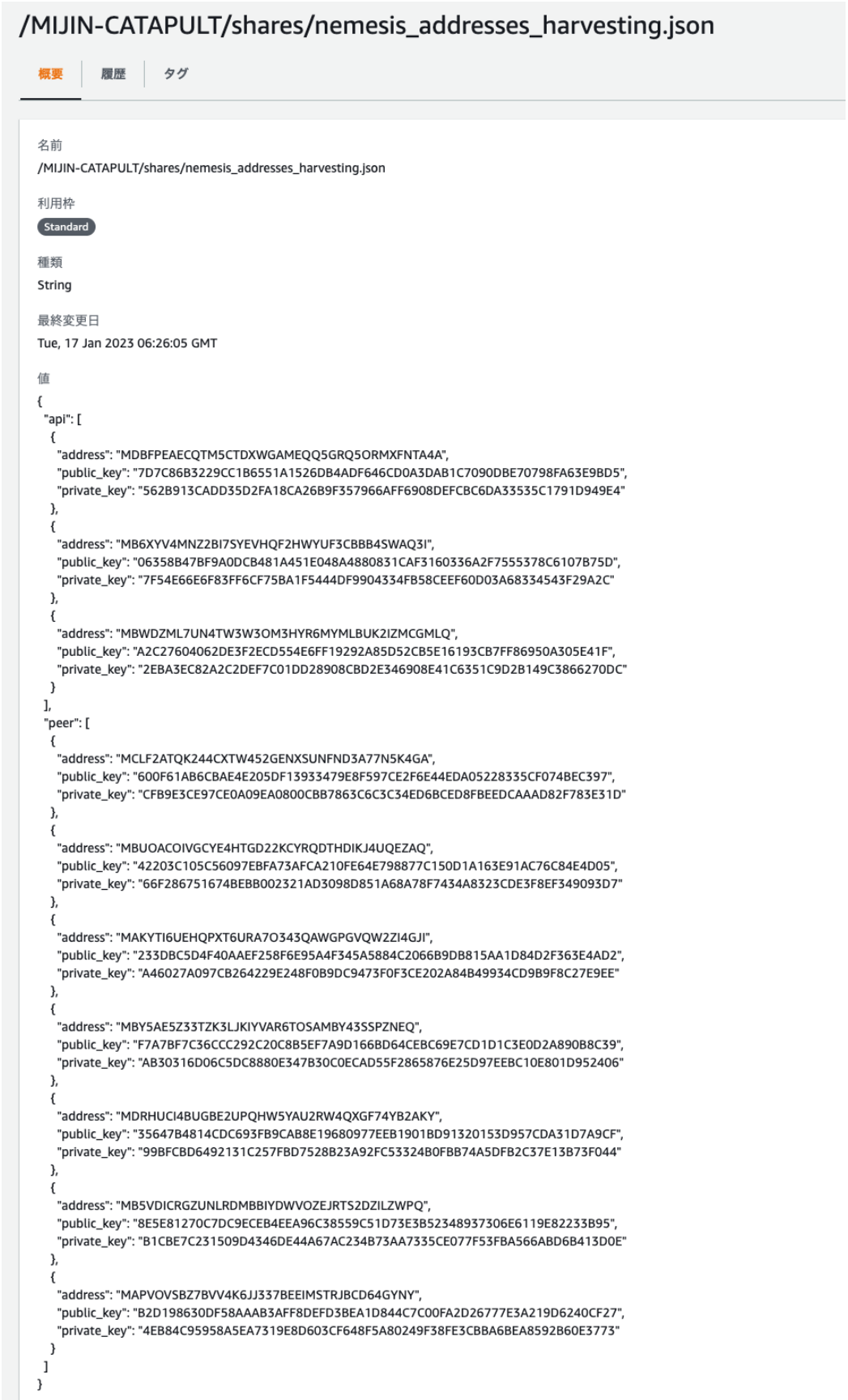
symbol-cli は アーカイブ化 されたため、symbol-cli は使用できない可能性があります。
1.0.3.4 以降は、mijin Catapult(v.2) 手数料ありモード有効時の残高アカウントの残高移動 を参照してください。

注釈:

ブロックチェーンにおいて、基軸通貨は必ずあり、ブロックチェーンを操作する上でトランザクション手数料を支払う必要があります。
mijin Catapult(v.2) でも手数料ありモードを有効にすると、トランザクション作成に手数料が必要となり、トランザクション手数料、Mosaic レンタル手数料など全てに基軸通貨の残高を持っているアカウントで操作する必要があります。
逆に、手数料なしモードでは、残高を持ってないアカウントを使ってトランザクションを作成することができます。

残高あるアカウントの確認

残高あるアカウントは、AWS Systems Manager パラメータストアにある、
nemesis_addresses_harvesting.json に保存されます。
このファイルのアカウント一覧は、各ノードにすべて紐づいています。



/MIJIN-CATAPULT/shares/nemesis_addresses_harvesting.json

概要 | 履歴 | タグ

名前

/MIJIN-CATAPULT/shares/nemesis_addresses_harvesting.json

利用枠

Standard

種類

String

最終変更日

Tue, 17 Jan 2023 06:26:05 GMT

値


```
{
 "api": [
  {
   "address": "MDBFPEAECQTM5CTDXWGAMEQQ5GRQ5ORMXFNTA4A",
   "public_key": "7D7C86B3229CC1B6551A1526DB4ADF646CD0A3DAB1C7090DBE70798FA63E9BD5",
   "private_key": "562B913CADD35D2FA18CA26B9F357966AFF6908DEFCBC6DA33535C1791D949E4"
  },
  {
   "address": "MB6XYV4MNZ2BI7SYEVHQF2HWYUF3CBBB4SWAQ3I",
   "public_key": "06358B47BF9A0DCB481A451E048A4880831CAF3160336A2F7555378C6107B75D",
   "private_key": "7F54E66E6F83FF6CF75BA1F5444DF9904334FB58CEEF60D03A68334543F29A2C"
  },
  {
   "address": "MBWDZML7UN4TW3W3OM3HYR6MYMLBUK2IZMCGMLQ",
   "public_key": "A2C27604062DE3F2ECD554E6FF19292A85D52CB5E16193CB7FF86950A305E41F",
   "private_key": "2EBA3EC82A2C2DEF7C01DD28908CBD2E346908E41C6351C9D2B149C3866270DC"
  }
 ],
 "peer": [
  {
   "address": "MCLF2ATQK244CXTW452GENXSUNFND3A77N5K4GA",
   "public_key": "600F61AB6CBAE4E205DF13933479E8F597CE2F6E44EDA05228335CF074BEC397",
   "private_key": "CFB9E3CE97CE0A09EA0800CBB7863C6C3C34ED6BCED8FBEEDCAAAD82F783E31D"
  },
  {
   "address": "MBUOACOIVGCYE4HTGD22KCYRQDTHDIKJ4UQEZAQ",
   "public_key": "42203C105C56097EBFA73AFCA210FE64E798877C150D1A163E91AC76C84E4D05",
   "private_key": "66F286751674BEBB002321AD3098D851A68A78F7434A8323CDE3F8EF349093D7"
  },
  {
   "address": "MAKYTI6UEHQPXT6URA7O343QAWGPGVQW2ZI4GJI",
   "public_key": "233DBC5D4F40AAEF258F6E95A4F345A5884C2066B9DB815AA1D84D2F363E4AD2",
   "private_key": "A46027A097CB264229E248F0B9DC9473F0F3CE202A84B49934CD9B9F8C27E9EE"
  },
  {
   "address": "MBY5AE5Z33TZK3LJKIYVAR6TOSAMBY43SSPZNEQ",
   "public_key": "F7A7BF7C36CCC292C20C8B5EF7A9D166BD64CEBC69E7CD1D1C3E0D2A890B8C39",
   "private_key": "AB30316D06C5DC8880E347B30C0ECAD55F2865876E25D97EEBC10E801D952406"
  },
  {
   "address": "MDRHUCI4BUGBE2UPQHW5YAU2RW4QXGF74YB2AKY",
   "public_key": "35647B4814CDC693FB9CAB8E19680977EEB1901BD91320153D957CDA31D7A9CF",
   "private_key": "99BFCBD6492131C257FBD7528B23A92FC53324B0FBB74A5DFB2C37E13B73F044"
  },
  {
   "address": "MB5VDICRGZUNLRDMBBIYDWVOZEJRTS2DZILZWPQ",
   "public_key": "8E5E81270C7DC9ECEB4EEA96C38559C51D73E3B52348937306E6119E82233B95",
   "private_key": "B1CBE7C231509D4346DE44A67AC234B73AA7335CE077F53FBA566ABD6B413D0E"
  },
  {
   "address": "MAPVOVSBZ7BVV4K6JJ337BEEIMSTRJBCD64GYNY",
   "public_key": "B2D198630DF58AAAB3AFF8DEFD3BEA1D844C7C00FA2D26777E3A219D6240CF27",
   "private_key": "4EB84C95958A5EA7319E8D603CF648F5A80249F38FE3CBBA6BEA8592B60E3773"
  }
 ]
}
```

すべてのアカウントに残高はありますが、ここでは一番上の以下のデータを symbol-cli を使用して確認します。

```
{
  "address": "MAL4SPKWUI3WGSNOWSDA3KKIBJG7QHMCXD7GZVA",
  "public_key": "FDA90ACB0B4DA564FBA3D9D3A3E67A7146A77D2F5C246BC67AC044AAD578E161",
  "private_key": "F36139408F597D2F0DA0C5E3CB1162E3D80EFEF188E21089284F57723676CC5C"
},
```

symbol-cli のインストール

npm より symbol-cli をインストールします。

```
$ sudo npm i -g symbol-cli@1.0.0
/usr/local/bin/symbol-cli -> /usr/local/lib/node_modules/symbol-cli/bin/symbol-cli
+ symbol-cli@1.0.0
updated 1 package in 8.724s
```

残高アカウントのインポート

残高アカウントをインポートします。

項目	説明	値
Select the network type	ネットワークを指定して下さい。 構築時に指定した CatapultNetwork の値	MIJIN/MIJIN_TEST
Enter the Symbol node URL.	Cloudformation Stack 内にある Outouts タブの mijinLBENdpoint または mijinEndpoint の URL を指定します。	<http://xxxxxxx:300>
Enter a profile name	アカウントを呼び出すプロファイル名を指定します。	任意
Enter your wallet password	アカウントのパスワードを指定します	任意
Do you want to set the account as the default profile?	このアカウントを Default で使うかどうかを指定します。	任意
Select an import type	再度保存するためのインポート方法を指定します。	PrivateKey
Enter your account private key	**nemesis_addresses_harvesting.json** の private_key を指定します。	任意 **ここでは、F3613 で始まる値**

```
$ symbol-cli profile import
✓ Select the network type: › MIJIN
✓ Enter the Symbol node URL. (Example: http://localhost:3000): ⋯ http://
↪xxxxxxxxxxxxxxxxxxxx.elb.ap-northeast-1.amazonaws.com:3000
✓ Enter a profile name: ⋯ mijin-have-currency-account
✓ Enter your wallet password: ⋯ ********
✓ Do you want to set the account as the default profile? ⋯ no
✓ Select an import type: › PrivateKey
```

(continues on next page)

(continued from previous page)

```
✓ Enter your account private key: ···
**************************************************************

Account
┌──────────────┬──────────────────────────────────────────────────────────────────┐
│ Property     │ Value                                                            │
├──────────────┼──────────────────────────────────────────────────────────────────┤
│ Address      │ MAL4SP-KWUI3W-GSNOWS-DA3KKI-BJG7QH-MCXD7G-ZVA                    │
├──────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key   │ FDA90ACB0B4DA564FBA3D9D3A3E67A7146A77D2F5C246BC67AC044AAD578E161  │
├──────────────┼──────────────────────────────────────────────────────────────────┤
│ Private Key  │ F36139408F597D2F0DA0C5E3CB1162E3D80EFEF188E21089284F57723676CC5C  │
├──────────────┼──────────────────────────────────────────────────────────────────┤
│ Password     │ Test1234                                                         │
└──────────────┴──────────────────────────────────────────────────────────────────┘

SUCCESS Stored mijin-have-currency-account profile
```

アカウントの情報を確認します。
Balance Information にある Mosaic Id の Amount の数値が高い **1D8350FA8D4830FA** が基軸通貨になります。

警告: 基軸通貨の Mosaic Id はデプロイ時に作成されるため、同じ ID でないことに注意してください。

```
$ symbol-cli account info --profile mijin-have-currency-account
⠸ Processing
Account Information
┌──────────────────┬──────────────────────────────────────────────────────────────────┐
│ Property         │ Value                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address          │ MAL4SP-KWUI3W-GSNOWS-DA3KKI-BJG7QH-MCXD7G-ZVA                    │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address Height   │ 1                                                                │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key       │ FDA90ACB0B4DA564FBA3D9D3A3E67A7146A77D2F5C246BC67AC044AAD578E161  │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key Height │ 1                                                               │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance       │ 2850000                                                          │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance Height │ 2600                                                            │
└──────────────────┴──────────────────────────────────────────────────────────────────┘
Balance Information
┌──────────────────┬──────────────────┬──────────────────┬──────────────────┐
│ Mosaic Id        │ Relative Amount  │ Absolute Amount  │ Expiration Height │
├──────────────────┼──────────────────┼──────────────────┼──────────────────┤
│ 1D8350FA8D4830FA │ 1,799,799,999.6  │ 1799799999600000 │ Never            │
├──────────────────┼──────────────────┼──────────────────┼──────────────────┤
│ 01964E14621F06F6 │ 3,000            │ 3000000          │ Never            │
└──────────────────┴──────────────────┴──────────────────┴──────────────────┘
```

新規アカウントの作成

残高がないアカウントを新規作成します。
アドレス **MADIFG-N27CKA-6DY42J-UMEFJA-7OKXLO-NXLAEQ-XII** に後ほど残高を移動します。

項目	説明	値
Select the network type	ネットワークを指定して下さい。 構築時に指定した CatapultNetwork の値	MIJIN/MIJIN_TEST
Do you want to save the account?	このアカウントを保存します。	yes
Select an import type	再度保存するためのインポート方法を指定します。	PrivateKey
Enter the Symbol node URL.	Cloudformation Stack 内にある Outouts タブの mijinLBENdpoint または mijinEndpoint の URL を指定します。	<http://xxxxxxx:300>
Enter a profile name	アカウントを呼び出すプロファイル名を指定します。	任意
Enter your wallet password	アカウントのパスワードを指定します	任意

```
$ symbol-cli account generate
✓ Select the network type: › MIJIN
✓ Do you want to save the account? ⋯ yes
✓ Select an import type: › PrivateKey
✓ Enter the Symbol node URL. (Example: http://localhost:3000): ⋯ http://
↪xxxxxxxxxxxxxxxxxxxx.elb.ap-northeast-1.amazonaws.com:3000
✓ Enter a profile name: ⋯ mijin-no-currency-account
✓ Enter your wallet password: ⋯ ********
✓ Do you want to set the account as the default profile? ⋯ no

Account
┌──────────────┬──────────────────────────────────────────────────────────────────┐
│ Property     │ Value                                                            │
├──────────────┼──────────────────────────────────────────────────────────────────┤
│ Address      │ MADIFG-N27CKA-6DY42J-UMEFJA-7OKXLO-NXLAEQ-XII                    │
├──────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key   │ B86CDD63C3BA820C4659CF7FC3D53DA035CF8370AC3E0DBF025BEE691AED7DFA  │
├──────────────┼──────────────────────────────────────────────────────────────────┤
│ Private Key  │ E911E779671BD33B26A9D424DB331A36BDD497BA62D91B27ADAA4B1350A52D43  │
├──────────────┼──────────────────────────────────────────────────────────────────┤
│ Password     │ Test1234                                                         │
└──────────────┴──────────────────────────────────────────────────────────────────┘

SUCCESS Stored mijin-no-currency-account profile
```

アカウントの情報をノードに問い合わせます。
このアカウントは、ノードに残高の記録がないため、エラーになっていることを確認してください。

```
$ symbol-cli account info --profile mijin-no-currency-account

ERR {"statusCode":404,"statusMessage":"Not Found","body":"{\"code\":\"ResourceNotFound\
↪",\"message\":\"no resource exists with id 'MADIFGN27CKA6DY42JUMEFJA7OKXLONXLAEQXII'\
↪"}"}

TIP The account has to receive at least one transaction to be recorded on the network
```

残高の移動

残高アカウント (mijin-have-currency-account) から　新規アカウント (mijin-no-currency-account) に残高を転送します。
ここでは 10 万 cat.currency 転送してみます。

残高アカウント (mijin-have-currency-account) から転送トランザクションをアナウンスします。

項目	説明	値
Enter your wallet password	設定したパスワードを指定してください	任意
Mosaics to transfer in the format (mosaicId(hex)|@aliasName)::absoluteAmount	基軸通貨 (cat.currency) を 10 万送ります。基軸通貨は可分性が 6 のため 6 桁小数点があるため、0 を 6 足します。	**3BF3AF8B22CB53D8::1**
Enter the recipient address or @alias	新規アカウントの転送先アドレスを指定します	MADIFGN27CKA6DY42JUMEFJA7OKXLONXLAEQXII
Enter a message	転送トランザクションにメッセージを追加できます	任意
Enter the maximum fee (absolute amount)	トランザクション手数料を指定します。 これは手数料モードで変わります。 手数料あり 20000 程度 (0.2cat.currency) 手数料なし 0	0
Select the transaction announce mode	トランザクションをアナウンスする方法を指定します。	normal

```
$ symbol-cli transaction transfer --profile mijin-have-currency-account
✓ Enter your wallet password: ⋯ ********
✓ Mosaics to transfer in the format (mosaicId(hex)|@aliasName)::absoluteAmount, (Ex:
sending 1 symbol.xym, @symbol.xym::1000000). Add multiple mosaics separated by commas:
⋯ @cat.currency::100000000000
✓ Enter the recipient address or @alias: ⋯ MADIFGN27CKA6DY42JUMEFJA7OKXLONXLAEQXII
✓ Enter a message: ⋯ test
✓ Enter the maximum fee (absolute amount): ⋯ 200000
✓ Select the transaction announce mode: › normal
┌──────────────────────────────────────────────────────────────────────────────┐
│                                   TRANSFER                                   │
├──────────────┬───────────────────────────────────────────────────────────────┤
│ Max fee:     │ 200,000                                                       │
├──────────────┼───────────────────────────────────────────────────────────────┤
│ Network type: │ MIJIN                                                        │
├──────────────┼───────────────────────────────────────────────────────────────┤
```

(continues on next page)

(continued from previous page)

```
│ Deadline:      │ 2022-10-01 23:57:37.489                                      │
├────────────────┼──────────────────────────────────────────────────────────────┤
│ Recipient:     │ MADIFG-N27CKA-6DY42J-UMEFJA-7OKXLO-NXLAEQ-XII                │
├────────────────┼──────────────────────────────────────────────────────────────┤
│ Message:       │ test                                                         │
├────────────────┼──────────────────────────────────────────────────────────────┤
│ Mosaic (1/1):  │ 100,000,000,000 cat.currency (85BBEA6CC462B244)              │
├────────────────┴──────────────────────────────────────────────────────────────┤
│                              Signature details                                │
├────────────────┬──────────────────────────────────────────────────────────────┤
│ Payload:       │ B500000000000000496383B0C2AF6B3295D615336F48B2C299AAF38619399C40 │
│                │ 85AC7F6CF58092EAE743D5C754DC3C149E4E5EFA8E6038519F8BAAFDDF3B05BC │
│                │ 41B355638528AE03FDA90ACB0B4DA564FBA3D9D3A3E67A7146A77D2F5C246BC6 │
│                │ 7AC044AAD578E161000000000001605441400D030000000000517F484B18000000 │
│                │ 60068299BAF8940F0F1CD268C21520FB9575B9B758090BA10500010000000000 │
│                │ 44B262C46CEABB8500E87648170000000074657374                   │
├────────────────┼──────────────────────────────────────────────────────────────┤
│ Hash:          │ 24FACA961CB1DF4D3F76DFFBE302D2CAA512F7D1BD424CC4E0D14ACFA7221FA4 │
├────────────────┼──────────────────────────────────────────────────────────────┤
│ Signer:        │ FDA90ACB0B4DA564FBA3D9D3A3E67A7146A77D2F5C246BC67AC044AAD578E161 │
└────────────────┴──────────────────────────────────────────────────────────────┘
✓ Do you want to announce this transaction? … yes

SUCCESS Transaction announced correctly

TIP To check if the network confirms or rejects the transaction, run the command
↪'symbol-cli transaction status'
```

新規アカウントに残高があることを確認します。
先ほどエラーになっていたアカウント情報が mijin Catapult(v.2) に認識され、残高を持っていることを確認できます。

```
$ symbol-cli account info --profile mijin-no-currency-account
⠏ Processing
Account Information
┌────────────────────┬──────────────────────────────────────────────────────────────────┐
│ Property           │ Value                                                            │
├────────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address            │ MADIFG-N27CKA-6DY42J-UMEFJA-7OKXLO-NXLAEQ-XII                    │
├────────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address Height     │ 2706                                                             │
├────────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key         │ 0000000000000000000000000000000000000000000000000000000000000000 │
├────────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key Height  │ 0                                                                │
├────────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance         │ 0                                                                │
├────────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance Height  │ 0                                                                │
└────────────────────┴──────────────────────────────────────────────────────────────────┘
Balance Information
┌──────────────────┬─────────────────┬─────────────────┬───────────────────┐
│ Mosaic Id        │ Relative Amount │ Absolute Amount │ Expiration Height │
```

(continues on next page)

(continued from previous page)

```
├──────────────────┼──────────────┼──────────────┼──────────────────┤
│ 1D8350FA8D4830FA │ 100,000      │ 100000000000 │ Never            │
└──────────────────┴──────────────┴──────────────┴──────────────────┘
```

2.2.7 AWS のトラブルシューティング

AWS Marketplace で起動した mijin Catapult(v.2) のトラブルシューティングを纏めます。

2.2.7.1 バックアップしたスナップショットからリストアする

本章では、AWS 上の mijin Catapult(v.2) のノードのデータを AWS Backup で取得した Snapshot からバックアップからリストアする手順を説明します。

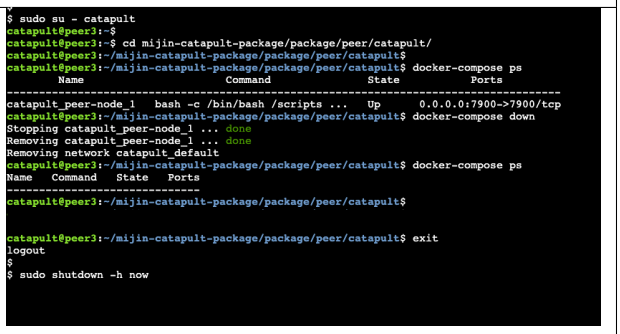
注釈:

Snaoshot からではなく、ブロックチェーンデータを消して、他のノードからリストアしたい場合は、mijin Catapult(v.2) のノード再同期 を参照してください。

ノードを完全に停止する

ブロックチェーンデータを
ここでは、PEER ノード 3 を例に停止します。

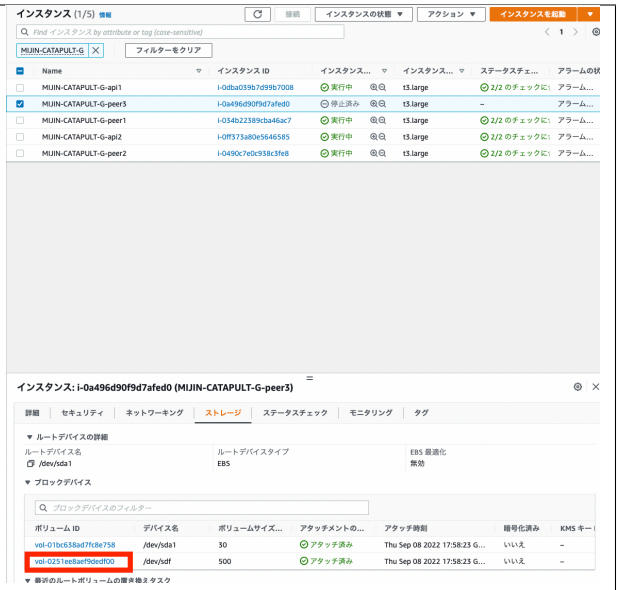
PEER ノード 3 にログインします。	mijin Catapult(v.2) EC2 インスタンスログイン方法 を参照し、ノードにログインしてください。
Volume が設置しているアベイラビリティゾーンをメモしてください。 GP3 であることを確認してください	

mijin Catapult(v.2) を停止し、EC2 インスタンスを停止します。
 	
`sudo su - catapult`
`cd mijin-catapult-package/package/peer/catapult/`
`docker-compose down`
`docker-compose ps`
`exit`
`sudo shutdown -h now`
	

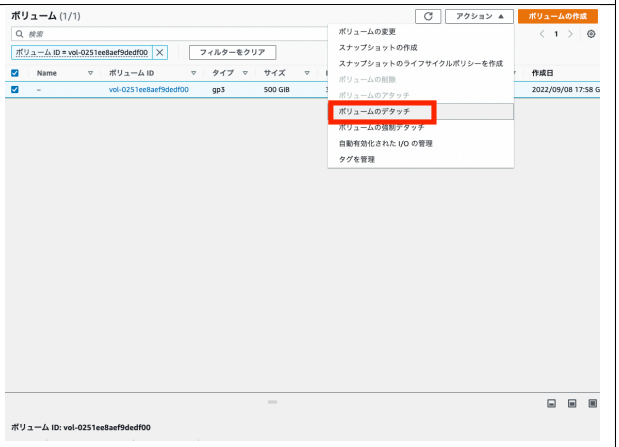
停止した PEER ノードの Volume をデタッチする

ノードにアタッチされているボリュームを外します。

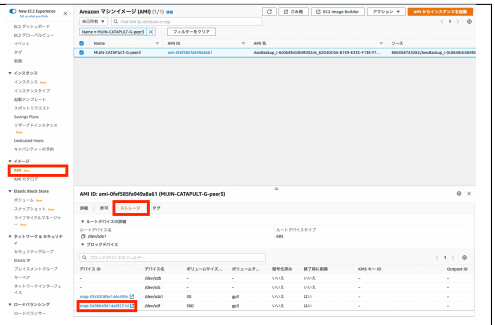
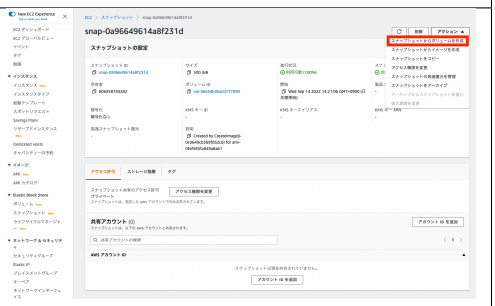
再度、PEER ノード 3 からボリュームを選択します。



ボリュームから「ボリュームをデタッチ」をクリックし、ポップアップした画面で OK を押します。

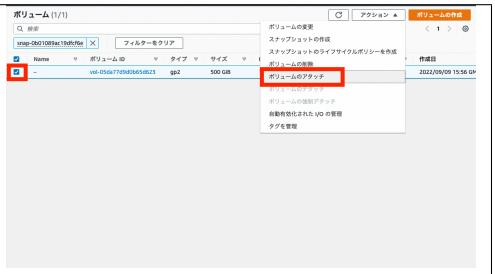
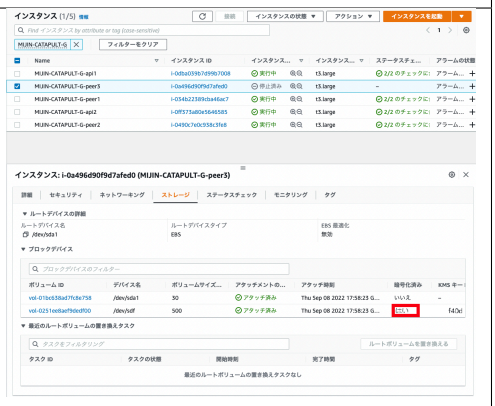


AWS Backup した Snapshot からボリュームを作成する

AWS Backup の場合、AMI から mijin のデータの Snapshot を選択します。 手動で取得した Snapshot を選択する場合は、この手順はスキップします。	
「スナップショットからボリュームを作成」をクリックします。	
ボリュームタイプ「gp3」を選択 アベイラビリティーゾーンを選択ノードを完全に停止する を参照	

PEER ノードにリストアした Volume をアタッチする

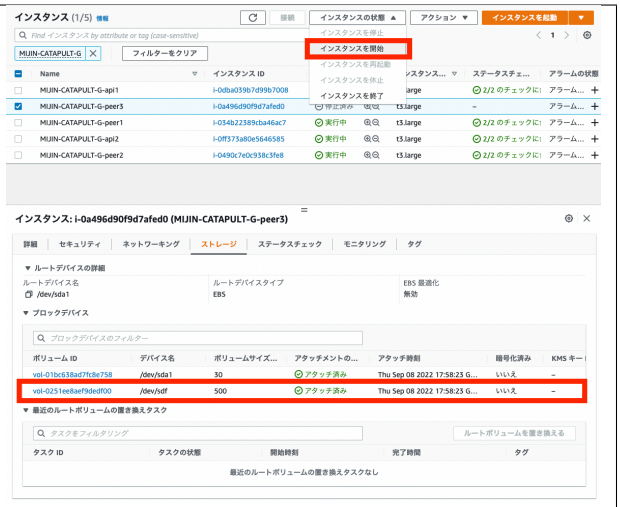
ノードに暗号化済みボリュームをアタッチします

暗号化済みボリュームを選択し、「ボリュームのアタッチ」をクリックします。	
インスタンスを PEER ノード 3 を選択し、デタッチしたボリュームと同じパスを指定します。 「ボリュームのアタッチ」をクリックします。	
リストアしたボリュームがアタッチされていることを確認します。	

停止した PEER ノードを起動する

停止していたノードを起動し、mijin Catapult(v.2) もあわせて起動します。

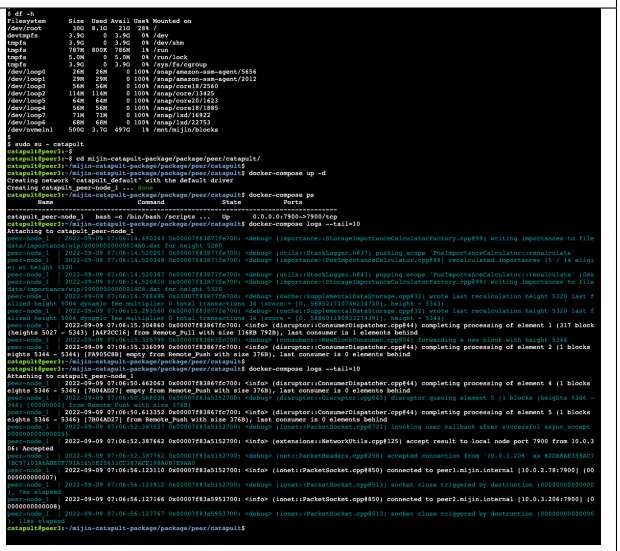
インスタンス一覧から、PEER ノード 3 をチェックし、「インスタンスの開始」をクリックします。



PEER ノード 3 にログインします。

mijin Catapult(v.2) EC2 インスタンスログイン方法 を参照し、ノードにログインしてください。

ディスクがマウントされていることを確認し、mijin Catapult(v.2) を起動します。



```
df -h
sudo su - catapult
cd mijin-catapult-package/package/peer/
↪catapult/
rm -rf /mnt/mijin/blocks/data/*.lock
docker-compose up -d
docker-compose ps
```

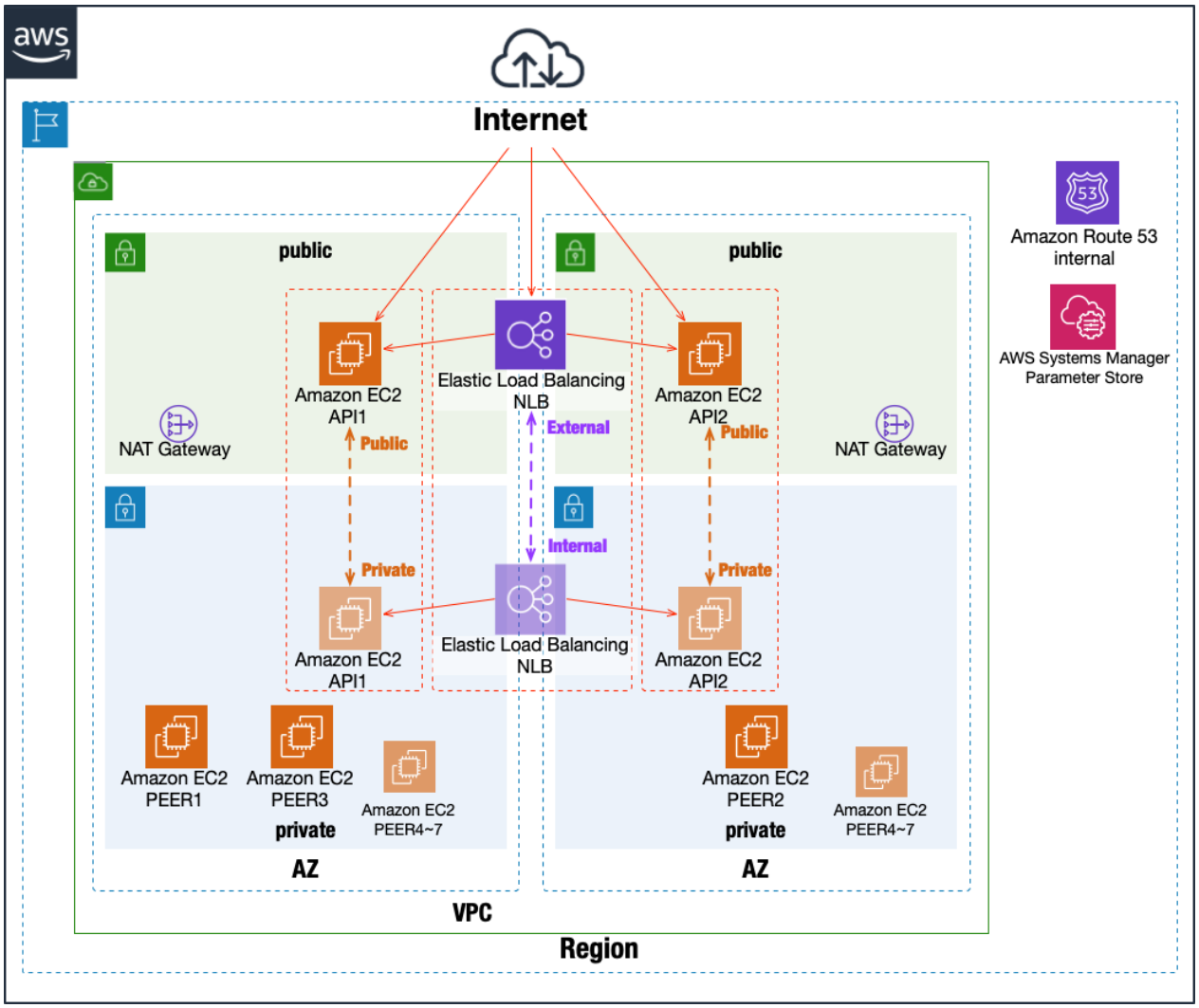
2.2.7.2 アベイラビリティゾーン (AZ) 障害時の対応方法

本章では、AWS 上の mijin Catapult(v.2) の VPC のアベイラビリティゾーン (AZ) に障害あった場合、どのように対応していくかを説明します。

mijin Catapult(v.2) 製品版の構成

mijin Catapult(v.2) 製品版では、１つのリージョンに VPC を配置し、二つのアベイラビリティゾーン (AZ) にノードを分散しています。
ELB 設置をデプロイ時のパラメーターで有効にすることで、片方の AZ に障害があっても、サービスを継続することができます。
mijin Catapult(v.2) は一台の PEER ノードがあれば、ブロックチェーンデータを更新することができます。
プログラムなどからアクセスしたい場合は、API ノードが一台あれば、アクセスを継続することができます。

例えば、以下の図にあるように左側 AZ に囲まれたサービス全体が停止したとしても、mijin Catapult(v.2) は停止することはありません。



AZ 復旧後の対応方法

AZ 復旧後の対応は簡単です。
mijin Catapult(v.2) は、復旧後 (EC2 インスタンス起動後)、自動で動いているブロックチェーンデータがあるノードに接続し、データの同期を開始します。
そのため、とくに復旧に特別な作業を必要としません。
ノードが復旧しているかを確認する場合は、以下のようにコマンド又はブラウザーから確認することで確認することができます。

http://mijin エンドポイント:3000/node/peers

```
$ curl http://mijin エンドポイント:3000/node/peers
[
    {
        "version": 0,
        "publicKey": "9073B0A623934996A9AAAC85C6DEC8540AE17258D6997E42E00100CCFE6848EF",
        "networkGenerationHashSeed":
```

(continues on next page)

(continued from previous page)

```
↪"B319300B02B12264B7DF867F0EFD583CC3C6E65ED2732E3FD77BBC1DE8E00E85",
        "roles": 70,
        "port": 7900,
        "networkIdentifier": 96,
        "host": "api1.mijin.internal",
        "friendlyName": "api1.mijin.internal"
    },
    {
        "version": 0,
       "publicKey": "82DA8AE358AC7DF7BC97103A6ABE0F791A1655E20633CC387ACE198A0B7E9AA0",
        "networkGenerationHashSeed":
↪"B319300B02B12264B7DF867F0EFD583CC3C6E65ED2732E3FD77BBC1DE8E00E85",
        "roles": 69,
        "port": 7900,
        "networkIdentifier": 96,
        "host": "peer2.mijin.internal",
        "friendlyName": "peer2.mijin.internal"
    },
    {
        "version": 0,
       "publicKey": "4EE257A9DD6D3F19331A467C6C76BA86B50B1297181E32C7A83C1184B666996C",
        "networkGenerationHashSeed":
↪"B319300B02B12264B7DF867F0EFD583CC3C6E65ED2732E3FD77BBC1DE8E00E85",
        "roles": 69,
        "port": 7900,
        "networkIdentifier": 96,
        "host": "peer1.mijin.internal",
        "friendlyName": "peer1.mijin.internal"
    },
    {
        "version": 0,
       "publicKey": "C158D513266B2C04216CDC03AD99036757A41AD2AFDF59D2A67F6D2D4F8CC84F",
        "networkGenerationHashSeed":
↪"B319300B02B12264B7DF867F0EFD583CC3C6E65ED2732E3FD77BBC1DE8E00E85",
        "roles": 69,
        "port": 7900,
        "networkIdentifier": 96,
        "host": "peer3.mijin.internal",
        "friendlyName": "peer3.mijin.internal"
    }
]
```

上記でノードが確認できない場合、急なノードによって、ブロックチェーンデータが壊れてしまう可能性があります。
その場合は、mijin Catapult(v.2) のノード再同期 やバックアップしたスナップショットからリストアするを参照し、ノードを復旧してください。

2.2.8 AWS MarketPlace mijin Catapult(v.2) FAQ 一覧

AWS MarketPlace で展開する mijin Catapult(v.2) における FAQ 一覧です。

2.2.8.1 製品版 FAQ 一覧

Q.Free Trial 版からデータを引き継ぐことは可能ですか？
A. データを引き継ぐことはできません。

Q. 商用での利用は可能でしょうか？
A. 可能です

Q.mijin ライセンス費用はどこに含まれているでしょうか？
A.mijin のライセンス費用は、AWS より請求されます。
EC2 インスタンスの費用同様に 1 時間単位の利用料金として請求が追加されます。

Q. 起動するのに初期費用はかかりますか？
A. 初期費用はかかりません。ただし、従量課金で mijin ライセンス費用と AWS 使用料が発生します。

Q. 製品版のフリートライアル期間はありますか？
A. 製品版のフリートライアル期間はございません。
フリートライアル版を別で用意していますので、AWS Marketplace を使ったデプロイ準備 のトライアル板を参照してください。

Q. サポートを受けたいのですが、どこに問い合わせればよいでしょうか？
A. ケースによって問い合わせ先が異なりますので、以下をご参照下さい。

■ 構築時の問題または mijin に関する技術問い合わせ (有償サポート)

※ mijin に関する技術問い合わせは、有償サポートチケットの購入が必要となりますので以下からお問い合わせください。

https://mijin.io/en/aws_contact

■ AWS 上のトラブル

https://aws.amazon.com/jp/premiumsupport/tech-support-guidelines/

Q.mijin Catapult (v.2) 製品版のバージョンを教えてください
A.

catapult-server: 1.0.3.8
catapult-rest: 2.5.0

(2025/6/10 時点)

Q. バージョンアップは可能でしょうか？
A. バージョンアップすることは可能です。
ただし、バージョン差が大きい場合、バージョンアップができない可能性もあります。
有償サポートを購入することで、バージョンアップのサポートやアナウンスを受け取ることができます。

https://mijin.io/en/aws_contact

Q. 同じリージョンに二つ目の mijin Catapult(v.2) を立ち上げようとして失敗しましたが、なぜでしょうか。
A. 二つ目以降の mijin Catapult(v.2) を立ち上げる場合はデプロイ時のパラメーター
「**mijinStackAlreadyExist** 」の値を **YES** に指定してください。

Q. 災害対策としてディザスタリカバリ環境を構築したいです。
A. 製品版は標準でマルチ AZ 環境にノードを分散して配置します。
詳しくは、AWS MarketPlace mijin Catapult(v.2) アーキテクチャパターンによるリカバリ戦略 を参照してください。

Q.AWS Systems Manager Parameter Store にあるデータをローテーションしたいです。
A.AWS Systems Manager Parameter Store にあるデータは、初期デプロイ時にのみ使用され、
バックアップとして保存されています。
Parameter Store にはノードの暗号化通信に使用する証明書データがあるため、更新したい場合は、
ノード間の暗号化通信の更新 を参照してください。

Q.mijin は arm(Graviton) のインスタンスでも動きますか？
A.AWS Marketplace には x86_64、arm それぞれのバージョンにて提供しています。
AWS MarketPlace Enterprise x86_64 Version
AWS MarketPlace Enterprise arm64 Version

Q.arm(Graviton) 版と x86_64 版の差はありますか？
A.mijin としては、それぞれの CPU アーキテクチャに最適化されていますが速度は同等であり、ライセンス料金も同一です。
arm で立ち上げた方が、AWS 利用料ランニングコストをを抑えることができます。
価格差については、それぞれの最低要件のインスタンスにおけるシミュレーションデータを参考にしてください。
(※ シミュレーションデータに mijin のライセンス費用は含まれません)

x86_64 構成 https://calculator.aws/#/estimate?id=3df2b4611ffde3cc598ffc6fec9aff49b8a986b2
arm64 構成
https://calculator.aws/#/estimate?id=c3bdc61df9a07f9760fdb790680cec8d3807b0dc

---

2.2.8.2 フリートライアル版 FAQ 一覧

Q. 機能制限はありますか？
A.mijin としての機能制限はございませんが、以下の制約を行っております。
1. 基軸通貨を「2,000cat.currency」と少額に制限しております。
2.Mosaic、Namespace、トランザクション発行手数料が必要になります。

Q. 利用期間の制限はありますか？
A. システム上での利用期間の設定はございませんが、あくまでテストライセンスのため、商用利用不可、サポート問い合わせできないなどございます。

Q. 商用利用可能な製品版はありますか？
A. 製品版は商用利用可能です。
詳しくは、AWS Marketplace を使ったデプロイ準備 を確認してください。

Q.mijin の機能についてのサポートはどこまでしていただけますか？
A. フリートライアル版は、テストライセンスのため、mijin についての機能や開発のサポートは行っておりません。
サポートが必要な場合は、製品版をお使いいただき、サポート契約していただくことで可能です。

Q.mijin Catapult (v.2) フリートライアル版のバージョンを教えてください
A.
catapult-server: 1.0.3.6
catapult-rest: 2.4.3

2.2.9 AWS MarketPlace mijin Catapult(v.2) 利用料金比較表

AWS MarketPlace で展開する mijin Catapult(v.2) では、AWS の EC2 インスタンスを自由に選択することができます。
ライセンス費用以外で発生するサーバ利用料金の比較表です。

プロセッサ	EC2	EBS	VPC	リージョン	インスタンス支払い方法	12 か月 ($)	月額 ($)	3 ヶ月概算 ($)	月額 (円)	3 ヶ月概算 (円)	見積
x86	t3.large * 5	GP2 30GB	新規	us-east-1	オンデマンド	4914.00	409.50	1227	¥57,330	¥171,780	見積
				us-east-1	リザーブド 1 年	3399.84	283.32	849	¥39,665	¥118,860	見積
				ap-north-east1	オンデマンド	6385.32	532.11	1596	¥74,495	¥223,440	見積
				ap-north-east1	リザーブド 1 年	4419.88	368.32	1104	¥51,565	¥154,560	見積
			既存	us-east-1	オンデマンド	4124.64	343.72	1029	¥48,121	¥144,060	見積

次のページに続く

表 12 – 前のページからの続き

プロセッサ	EC2	EBS	VPC	リージョン	インスタンス支払い方法	12 か月 ($)	月額 ($)	3 ヶ月概算 ($)	月額 (円)	3 ヶ月概算 (円)	見積
				us-east-1	リザーブド 1 年	2610.48	217.54	651	¥30,456	¥91,140	見積
				ap-north-east1	オンデマンド	5297.64	441.47	1323	¥61,806	¥185,220	見積
				ap-north-east1	リザーブド 1 年	3332.20	277.68	831	¥38,876	¥116,340	見積
	API c5n.2xlage * 2 PEER c5n.xlarge * 3	IO1 IOPS100 130GB * 2 IO1 IOPS100 80GB * 3	新規	us-east-1	オンデマンド	15474.96	1289.58	3867	¥180,541	¥541,380	見積
				us-east-1	リザーブド 1 年	10018.84	834.90	2502	¥116,886	¥350,280	見積
				ap-north-east1	オンデマンド	19378.92	1614.91	4842	¥226,087	¥677,880	見積
				ap-north-east1	リザーブド 1 年	12506.88	1042.24	3126	¥145,914	¥437,640	見積
			既存	us-east-1	オンデマンド	14685.60	1223.80	3669	¥171,332	¥513,660	見積
				us-east-1	リザーブド 1 年	9229.48	769.12	2307	¥107,677	¥322,980	見積
				ap-north-east1	オンデマンド	18291.24	1524.27	4572	¥213,398	¥640,080	見積
				ap-north-east1	リザーブド 1 年	11419.20	951.60	2853	¥133,224	¥399,420	見積
Arm	t4g.large	GP2 30GB	新規	us-east-1	オンデマンド	4213.20	351.10	1053	¥49,154	¥147,420	見積
				us-east-1	リザーブド 1 年	2994.84	249.57	747	¥34,940	¥104,580	見積
				ap-north-east1	オンデマンド	5404.20	450.35	1350	¥63,049	¥189,000	見積
				ap-north-east1	リザーブド 1 年	3844.88	320.41	960	¥44,857	¥134,400	見積
			既存	us-east-1	オンデマンド	3423.84	285.32	855	¥39,945	¥119,700	見積
				us-east-1	リザーブド 1 年	2205.48	183.79	549	¥25,731	¥76,860	見積
				ap-north-east1	オンデマンド	4316.52	359.71	1077	¥50,359	¥150,780	見積
				ap-north-east1	リザーブド 1 年	2987.20	248.93	744	¥34,851	¥104,160	見積

次のページに続く

表 12 – 前のページからの続き

プロセッサ	EC2	EBS	VPC	リージョン	インスタンス支払い方法	12 か月 ($)	月額 ($)	3 ヶ月概算 ($)	月額 (円)	3 ヶ月概算 (円)	見積
	API c6g.2xlage * 2 PEER c6g.xlarge * 3	IO1 IOPS100 130GB * 2 IO1 IOPS100 80GB * 3	新規	us-east-1	オンデマンド	10569.36	880.78	2640	¥123,309	¥369,600	見積
				us-east-1	リザーブド 1 年	7134.84	594.57	1782	¥83,240	¥249,480	見積
				ap-north-east1	オンデマンド	13197.86	1099.82	3297	¥153,975	¥461,580	見積
				ap-north-east1	リザーブド 1 年	8873.88	739.49	2217	¥103,529	¥310,380	見積
			既存	us-east-1	オンデマンド	9780.00	815.00	2445	¥114,100	¥342,300	見積
				us-east-1	リザーブド 1 年	6345.48	528.79	1584	¥74,031	¥221,760	見積
				ap-north-east1	オンデマンド	12110.18	1009.18	3027	¥141,285	¥423,780	見積
				ap-north-east1	リザーブド 1 年	7786.20	648.85	1944	¥90,839	¥272,160	見積

Technical

3.1 mijin Catapult(v.2) の基礎知識

3.1.1 mijin Catapult(v.2) のアクセス方法

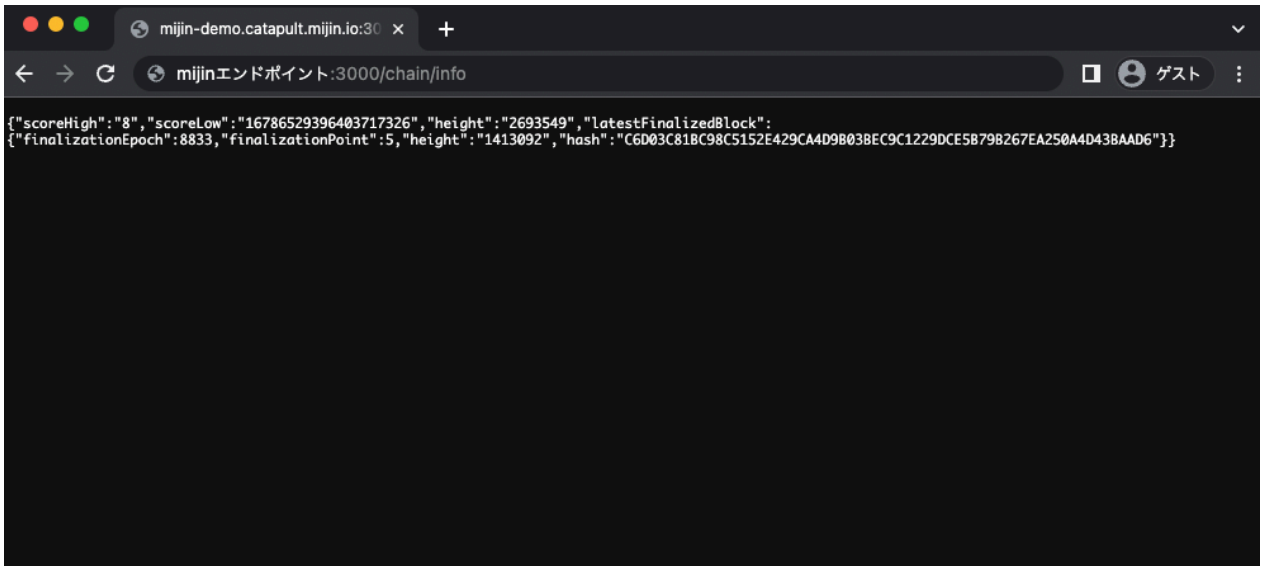
mijin Catapult(v.2) のアクセスは API ノードの API Rest エンドポイントにアクセスすることでブロックチェーンの操作が可能です。

以下では、コマンドでブロックチェーンの現在のブロック数を確認することができます。

```
$ curl -Ss http://mijin エンドポイント:3000/chain/info | jq -r
{
  "scoreHigh": "8",
  "scoreLow": "16778237146341708801",
  "height": "2693365",
  "latestFinalizedBlock": {
    "finalizationEpoch": 8833,
    "finalizationPoint": 5,
    "height": "1413092",
    "hash": "C6D03C81BC98C5152E429CA4D9B03BEC9C1229DCE5B79B267EA250A4D43BAAD6"
  }
}
```

項目	説明
height	ブロックチェーンの現在のブロック高
latestFinalizedBlock.height	確定したブロックチェーンのブロック高

Chrome などのブラウザでも確認することができます。



その他、sdk や cli でもアクセスすることができます。

cli のアクセス方法はmijin Catapult(v.2) を操作する を参照してください。

3.1.2 mijin Catapult(v.2) ステータス確認方法

mijin ノードの各状態は、REST から取得できます。
ブラウザや curl で直接確認することができます。
また、sdk でも同様に確認することができます。

REST のレスポンスに関しては、Symbol from Nem と共通となります。

https://symbol.github.io/symbol-openapi/v1.0.0/

3.1.2.1 ブロック高を確認する

/chain/info

```
$ curl -Ss http://mijin-catapult-1-nlb-rest-eef0ebffe49c4aa3.elb.ap-northeast-1.
↪amazonaws.com:3000/chain/info | jq -r
{
  "scoreHigh": "0",
  "scoreLow": "1400415221207545868",
  "height": "12318",
  "latestFinalizedBlock": {
    "finalizationEpoch": 78,
    "finalizationPoint": 8,
    "height": "12300",
    "hash": "7DE8B8052D35E29D0020EE7DB65BE075F0B6CEC69F17018447470E205B68175D"
  }
}
```

項目	説明
height	現在のブロック高
latestFinalizedBlock	ファイナライズブロック
finalizationEpoch	•
finalizationPoint	•
height	•

3.1.2.2 REST のバージョンを確認する

/node/server

```
$ curl -Ss http://mijin-catapult-1-nlb-rest-eef0ebffe49c4aa3.elb.ap-northeast-1.
↪amazonaws.com:3000/node/server | jq -r
{
  "serverInfo": {
    "restVersion": "2.3.5",
    "sdkVersion": "2.3.5"
  }
}
```

項目	説明
restVersion	rest の version
sdkVersion	rest が使用する catapult-sdk の version

3.1.2.3 ノード情報を確認する

/node/info

```
$ curl -Ss http://mijin-catapult-1-nlb-rest-eef0ebffe49c4aa3.elb.ap-northeast-1.
↪amazonaws.com:3000/node/info | jq -r
{
  "version": 16777216,
  "publicKey": "03ECD9C1929E26ED53BEBCCCF17E6F32F37ED9C6474397F592C883F771AB6A05",
  "networkGenerationHashSeed":
↪"2DE20B93EBE048A3BA132CC9874BCABBC21C87E18FE9836B8D5D002E57640D4B",
  "roles": 70,
  "port": 7900,
  "networkIdentifier": 96,
  "host": "api2.mijin.internal",
  "friendlyName": "api2.mijin.internal",
  "nodePublicKey": "5958AE940208CF8FD0D7FF2A584F8B234A3814AFC4D93F304A5CEA926EF6A747"
}
```

項目	説明
publicKey	ハーベストで使われる公開鍵
networkGenerationHashSeed	作成したブロックチェーン固有の GenerationHash 設定
roles	ノードのロール (api/peer/dual/voting)
port	ノード間の通信ポート
networkIdentifier	ネットワークタイプ
host	実行しているノードのホスト名
friendlyName	ノードのフレンドリー名
nodePublicKey	ノード用の公開鍵

3.1.2.4 接続しているノードを確認する

/node/peers

```
$ curl -Ss http://mijin-catapult-1-nlb-rest-eef0ebffe49c4aa3.elb.ap-northeast-1.
↪amazonaws.com:3000/node/peers | jq -r
[
  {
    "version": 0,
    "publicKey": "DB8D9DD59D78AE62E157824305DE31B9D415AA217EFE1DF14A7361E9D20E7456",
    "networkGenerationHashSeed":
↪"2DE20B93EBE048A3BA132CC9874BCABBC21C87E18FE9836B8D5D002E57640D4B",
    "roles": 69,
    "port": 7900,
    "networkIdentifier": 96,
    "host": "peer1.mijin.internal",
    "friendlyName": "peer1.mijin.internal"
  },
  {
    "version": 0,
    "publicKey": "22722F1534AE77DA44A065C0E2ACB125CB66FB45E80403A183EFEBE222BF3D90",
    "networkGenerationHashSeed":
↪"2DE20B93EBE048A3BA132CC9874BCABBC21C87E18FE9836B8D5D002E57640D4B",
    "roles": 69,
    "port": 7900,
    "networkIdentifier": 96,
    "host": "peer2.mijin.internal",
    "friendlyName": "peer2.mijin.internal"
  },
  {
    "version": 0,
    "publicKey": "239CC13A2B3D112C4146415EE532146D5338614BBBAD1A1E2E8E4690638F07D9",
    "networkGenerationHashSeed":
↪"2DE20B93EBE048A3BA132CC9874BCABBC21C87E18FE9836B8D5D002E57640D4B",
    "roles": 69,
    "port": 7900,
    "networkIdentifier": 96,
    "host": "peer3.mijin.internal",
    "friendlyName": "peer3.mijin.internal"
  },
  {
    "version": 0,
    "publicKey": "E4BF3706483B4D42243F3DCB2625021C3E3AE7C253CC466154EEDF9775012C20",
    "networkGenerationHashSeed":
↪"2DE20B93EBE048A3BA132CC9874BCABBC21C87E18FE9836B8D5D002E57640D4B",
```

(continues on next page)

(continued from previous page)

```
    "roles": 70,
    "port": 7900,
    "networkIdentifier": 96,
    "host": "api1.mijin.internal",
    "friendlyName": "api1.mijin.internal"
  }
]
```

項目	説明
publicKey	ハーベストで使われる公開鍵
networkGenerationHashSeed	作成したブロックチェーン固有の GenerationHash 設定
roles	ノードのロール (api/peer/dual/voting)
port	ノード間の通信ポート
networkIdentifier	ネットワークタイプ
host	実行しているノードのホスト名
friendlyName	ノードのフレンドリー名
nodePublicKey	ノード用の公開鍵

3.1.2.5 総トランザクション数、総アカウント数を確認する

/node/storage

```
$ curl -Ss http://mijin-catapult-1-nlb-rest-eef0ebffe49c4aa3.elb.ap-northeast-1.
↪amazonaws.com:3000/node/storage | jq -r
{
  "numBlocks": 12322,
  "numTransactions": 34,
  "numAccounts": 14
}
```

項目	説明
numBlocks	現在のブロック高
numTransactions	過去発行されたトランザクション数の総数
numAccounts	過去使用されたアカウントの総数

3.1.2.6 ネットワークタイプを確認する

` /network `

```
$ curl -Ss http://mijin-catapult-1-nlb-rest-eef0ebffe49c4aa3.elb.ap-northeast-1.
↪amazonaws.com:3000/network | jq -r
{
  "name": "mijin",
  "description": "mijin network"
}
```

項目	説明
name	使われているネットワーク名 mijin or mijin-test
description	ネットワークの説明

3.1.2.7 ノードのコンテナの状況を確認する

/node/health

```
$ curl -Ss http://mijin-catapult-1-nlb-rest-eef0ebffe49c4aa3.elb.ap-northeast-1.
↪amazonaws.com:3000/node/health | jq -r
{
  "status": {
    "apiNode": "up",
    "db": "up"
  }
}
```

項目	説明
apiNode	api-node コンテナのステータス up or down
db	db コンテナのステータス up or down

3.1.2.8 ブロックチェーン全体の設定を確認する

/network/properties

```
$ curl -Ss http://mijin-catapult-1-nlb-rest-eef0ebffe49c4aa3.elb.ap-northeast-1.
↪amazonaws.com:3000/network/properties | jq -r
{
  "network": {
    "identifier": "mijin",
    "nemesisSignerPublicKey":
↪"12086D4CB80CB6461887427BD49ED22D3914117526F573CC6F9937FC19DB2F73",
    "nodeEqualityStrategy": "host",
    "generationHashSeed":
↪"2DE20B93EBE048A3BA132CC9874BCABBC21C87E18FE9836B8D5D002E57640D4B",
    "epochAdjustment": "1560294000s"
  },
  "chain": {
    "enableVerifiableState": true,
    "enableVerifiableReceipts": true,
    "currencyMosaicId": "0x61D0'A72B'3C62'5448",
    "harvestingMosaicId": "0x1248'680A'CB99'E205",
    "blockGenerationTargetTime": "15s",
    "blockTimeSmoothingFactor": "3000",
    "importanceGrouping": "40",
    "importanceActivityPercentage": "5",
    "maxRollbackBlocks": "0",
    "maxDifficultyBlocks": "60",
    "defaultDynamicFeeMultiplier": "0",
    "maxTransactionLifetime": "24h",
    "maxBlockFutureTime": "500ms",
    "initialCurrencyAtomicUnits": "8'998'999'998'000'000",
    "maxMosaicAtomicUnits": "9'000'000'000'000'000",
    "totalChainImportance": "15'000'000",
    "minHarvesterBalance": "1'000'000",
    "maxHarvesterBalance": "15'000'000",
    "minVoterBalance": "1'000'000",
    "votingSetGrouping": "160",
    "maxVotingKeysPerAccount": "3",
```

(continues on next page)

(continued from previous page)

```
    "minVotingKeyLifetime": "72",
    "maxVotingKeyLifetime": "26280",
    "harvestBeneficiaryPercentage": "10",
    "harvestNetworkPercentage": "5",
    "harvestNetworkFeeSinkAddress": "MBVF6QLFNKAXDBZLJYBPBT2YYKMJW7UE7GH7RTY",
    "maxTransactionsPerBlock": "6'000"
  },
  "plugins": {
    "accountlink": {
      "dummy": "to trigger plugin load"
    },
    "aggregate": {
      "maxTransactionsPerAggregate": "1'000",
      "maxCosignaturesPerAggregate": "25",
      "enableStrictCosignatureCheck": false,
      "enableBondedAggregateSupport": true,
      "maxBondedTransactionLifetime": "48h"
    },
    "lockhash": {
      "lockedFundsPerAggregate": "0",
      "maxHashLockDuration": "2d"
    },
    "locksecret": {
      "maxSecretLockDuration": "30d",
      "minProofSize": "1",
      "maxProofSize": "1000"
    },
    "metadata": {
      "maxValueSize": "1024"
    },
    "mosaic": {
      "maxMosaicsPerAccount": "1'000",
      "maxMosaicDuration": "3650d",
      "maxMosaicDivisibility": "6",
      "mosaicRentalFeeSinkAddress": "MBKRTIOKHE34GF7J5WZDW6VLXEDYFRFFURN2EZA",
      "mosaicRentalFee": "0"
    },
    "multisig": {
      "maxMultisigDepth": "3",
      "maxCosignatoriesPerAccount": "25",
      "maxCosignedAccountsPerAccount": "25"
    },
    "namespace": {
      "maxNameSize": "64",
      "maxChildNamespaces": "256",
      "maxNamespaceDepth": "3",
      "minNamespaceDuration": "1m",
      "maxNamespaceDuration": "3650d",
      "namespaceGracePeriodDuration": "30d",
      "reservedRootNamespaceNames": "xem, nem, user, account, org, com, biz, net, edu,
mil, gov, info",
      "namespaceRentalFeeSinkAddress": "MBWRFMKEJRDUZC5WEW2PFYG374AI444HL2WQX6A",
      "rootNamespaceRentalFeePerBlock": "1",
      "childNamespaceRentalFee": "0"
    },
    "restrictionaccount": {
      "maxAccountRestrictionValues": "512"
```

(continues on next page)

(continued from previous page)

```
    },
    "restrictionmosaic": {
      "maxMosaicRestrictionValues": "20"
    },
    "transfer": {
      "maxMessageSize": "1024"
    }
  }
}
```

この設定値については、

3.1.2.9 トランザクション手数料を確認する

/network/fees/transaction

```
$ curl -Ss http://mijin-catapult-1-nlb-rest-eef0ebffe49c4aa3.elb.ap-northeast-1.
↪amazonaws.com:3000/network/fees/transaction | jq -r
{
  "averageFeeMultiplier": 0,
  "medianFeeMultiplier": 0,
  "highestFeeMultiplier": 0,
  "lowestFeeMultiplier": 0,
  "minFeeMultiplier": 0
}
```

項目	説明
averageFeeMultiplier	平均の乗数値 (自動)
medianFeeMultiplier	中央値の乗数値 (自動)
highestFeeMultiplier	使われた最大の乗数値
lowestFeeMultiplier	使われた最小の乗数値
minFeeMultiplier	ノードに設定した最小で必要な乗数値。0 であれば、手数料なしモード

3.2 mijin Catapult(v.2) を操作する

Symbol という公開ブロックチェーンと mijin は元々同じものであり、デュアルライセンスのもと OSS 化しています。そのため、Symbol のツールを mijin でも使用することが可能です。ここでは、mijin Catapult(v.2) にて送金するテストを操作します。

警告:

symbol-cli は Public Archive となっており、正常に動作しない可能性があります。
symbol-sdk は、mijin 用にカスタムした sdk を fork しています。

3.2.1 mijin アカウント作成

本章では、mijin を操作するために、必ず必要となるアカウント操作について説明します。
操作は Linux(Ubuntu20.04) にて行いますので、最低限の Linux 操作を理解している前提になります。

注釈: mijin Catapult(v.2) を操作するにはアカウントを作成し、そのアカウントを使ってトランザクションを送信します。また、デプロイ時の指定で手数料ありモードの場合は、アカウントに基軸通貨 (cat.currency) の残高を持つ必要があることに注意してください。

3.2.1.1 nodejs 及び yarn をインストール

mijin-catapult-tools を使用するため nodejs をインストールします。nodejs は NodeSource を利用してインストールします。

```
$ curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash - && sudo apt-get
install -y nodejs
$ node --version
$ sudo npm install -g yarn
```

3.2.1.2 mijin-catapult-tools のインストール

yarn を使用し、mijin-catapult-tools をインストールします。

```
$ yarn global add  @tech-bureau/mijin-catapult-tools
$ echo 'export PATH="$HOME/.yarn/bin:$PATH"' >> ~/.bashrc && source ~/.bashrc
```

mijin-catapult-tools が使えることを確認します。

```
$ mijin-catapult-tools
```

3.2.1.3 アカウントの作成

まずは、テストで使用するアカウントを作成してみましょう。ここでは、**test1Account test2Account** を使用するため mijin.json に保存します。

アカウント作成

```
$ mijin-catapult-tools account generate -u http://localhost:3000 -w mijin.json -s
2023-01-17T06:34:59.412Z [info] : mijin URL: http://localhost:3000
2023-01-17T06:34:59.413Z [info] : Network: 96
2023-01-17T06:34:59.413Z [info] : Mosaic Currency Id: 268CF9B2D33FBD22
2023-01-17T06:34:59.413Z [info] : Mosaic Harvest Id: 4C39D26C386E3182
2023-01-17T06:34:59.413Z [info] : Start Account Generate...
2023-01-17T06:34:59.475Z [info] : Write Config File: mijin.json
2023-01-17T06:34:59.476Z [info] : New Account: {
  "url": "http://localhost:3000",
  "workAccount": {
```

(continues on next page)

(continued from previous page)


```
    "publicKey": "425F06A8870381A00BD83E2D1083BB690F9FCB815F0919DFFA1A53A68E144D06",
    "privateKey": "9E9C660164AB344030DF5E77539952D9F5B380311C978369BA08923F577D8DAB",
    "address": "MC5VTRZO7IGJUG2VPQXBD4GDO4A74YLFUCVZZGI"
  },
  "balanceAccount": {
    "publicKey": "",
    "privateKey": "",
    "address": ""
  },
  "mainAccount": {
    "publicKey": "F2985009341A526B17E954EB6EA3EC77E1A0B33AE31EB29F86A69D9BB283AF67",
    "privateKey": "FF9A552ED35D930378F1E6B349A677771F19A06BCA0C4D5DCA621F354F5C6956",
    "address": "MCV66SABR6MBWT2I56YI3ULCPYWREJFX5DHHGDQ"
  },
  "keylink": {
    "vrf": {
      "publicKey": "C074A57A0EDF633414DDD764C6771926E91596481E0C2CAD4D9C3EFAAA432BCE",
      "privateKey": "C07D9A63B8DD8B0E639AAAE6059D7FA6C554082677926B7CACF0514E3940692C",
      "address": "MBLYBGODVGLT3HED6EXAQAXJE4CHHVRWGIAS45Y"
    },
    "voting": {
      "publicKey": "F6571CBC420A4EBA09E027AA53E5DD9486642911CDC11E0D3A6D1B2E4BC228D1",
      "privateKey": "FD5744779348F77F6EA6288232D0C8944E2F3C9E24D2E0FB1E93A8C4F9DD9BA5",
      "address": "MD2M6SDIE64O6ZWF2IJ7R4R5RDKOT2FNJDUIYOQ"
    }
  },
  "test1Account": {
    "publicKey": "CB51613497A40D9A256B17932579BC64D5037A04B29737B944965C1ADADD6E04",
    "privateKey": "8D5969EF1796F5F90256C92B5017396E40786ED87995169D4E26C9E5E01D7F8C",
    "address": "MAS36UGDCOGG6GYCBDPX3ROISABSNPZ6JQXMJSA"
  },
  "test2Account": {
    "publicKey": "5CC14799D5B2643914C8E574C8D073A7EE9AE2A405F1339A53612B566498AB1D",
    "privateKey": "FDE625C1D53AF04533FEB06A3556679FC02C4C6246952D3A534EB3E19CF83C56",
    "address": "MC5AH4UGBPPHNCMPTVNSY6LRDPIMEDODS2O373A"
  }
}
```


この時点で test1Account、test2Account を確認しても、存在しないことになっています。
ブロックチェーン上で、このアドレスのやり取りがないため、存在を確認できないからです。

```
$ catapult@catapult:~$ mijin-catapult-tools account info -r mijin.json -t test1
2023-01-17T06:36:26.600Z [info] : mijin URL: http://localhost:3000
2023-01-17T06:36:26.601Z [info] : Network: 96
2023-01-17T06:36:26.601Z [info] : Mosaic Currency Id: 268CF9B2D33FBD22
2023-01-17T06:36:26.601Z [info] : Mosaic Harvest Id: 4C39D26C386E3182
2023-01-17T06:36:26.601Z [info] : Start Account Info
2023-01-17T06:36:26.620Z [error] : Address Not Found
```

```
$ mijin-catapult-tools account info -r mijin.json -t test2
2023-01-17T06:36:43.387Z [info] : mijin URL: http://localhost:3000
2023-01-17T06:36:43.387Z [info] : Network: 96
2023-01-17T06:36:43.387Z [info] : Mosaic Currency Id: 268CF9B2D33FBD22
```

(continues on next page)

(continued from previous page)

```
2023-01-17T06:36:43.387Z [info] : Mosaic Harvest Id: 4C39D26C386E3182
2023-01-17T06:36:43.387Z [info] : Start Account Info
2023-01-17T06:36:43.405Z [error] : Address Not Found
```

手数料について

手数料とは、以下の項目で発行するトランザクションによって手数料が発生し、基軸通貨 (cat.currency) にて支払う必要があります。

- トランザクション手数料
- Mosaic レンタル手数料
- Namespace レンタル手数料
- ロック保証金

マーケットプレイスで提供する mijin では手数料モードは標準で **なし**に設定されています。
手数料なしでは、アカウントに基軸通貨の残高を持つ必要がありませんので特に意識することはありませんが、sdk などで手数料設定を都度 **0** を設定する必要があります。
手数料ありで mijin を作成した場合は、(番外) 手数料ありモード時の、基軸通貨の移動方法 の項目を参照し、**test1Account** のアカウントに基軸通貨を送信してください。

3.2.1.4 Mosaic を作成し転送

まずは基本となる Mosaic を作成し、転送してみましょう。

Mosaic の作成

test1 アカウントをオーナーとした 1 つの発行数の Mosaic(Token) を発行します。

```
$ mijin-catapult-tools transaction mosaic create -r mijin.json -o test1 -s 1 -d 0
2023-01-17T06:42:51.942Z [info] : mijin URL: http://localhost:3000
2023-01-17T06:42:51.942Z [info] : Network: 96
2023-01-17T06:42:51.942Z [info] : Create Mosaic...
2023-01-17T06:42:51.955Z [info] : Mosaic Owener Account:
MAS36UGDCOGG6GYCBDPX3ROISABSNPZ6JQXMJSA
2023-01-17T06:42:51.955Z [info] : MosaicId: 3D86C9FE5D52DE6F
2023-01-17T06:42:51.955Z [info] : Mosaic Flags: supplymutable:true, transferable:true,
restrictable:true, revokable:false
2023-01-17T06:42:51.955Z [info] : Mosaic Supply: divisibility:0, supply:1
2023-01-17T06:42:51.964Z [info] : Transaction Fee: 0
2023-01-17T06:42:51.964Z [info] : Mosaic Rental Fee: 0
2023-01-17T06:42:51.964Z [info] : Start Aggregate Transaction...
2023-01-17T06:43:00.967Z [info] : End Aggregate Transaction
2023-01-17T06:43:00.967Z [info] : http://localhost:3000/transactionStatus/
↪39BBE7E083460C3B85EB7D5FA2FB486F9786CA3FF1E0F976214F5753E527383A
2023-01-17T06:43:00.967Z [info] : http://localhost:3000/transactions/confirmed/
↪39BBE7E083460C3B85EB7D5FA2FB486F9786CA3FF1E0F976214F5753E527383A
```

A アカウントの情報を確認すると、MosaicId 3D86C9FE5D52DE6F を 1 だけ持っていることが確認できます。

```
$ mijin-catapult-tools account info -r mijin.json -t test1
2023-01-17T06:43:43.349Z [info] : mijin URL: http://localhost:3000
2023-01-17T06:43:43.350Z [info] : Network: 96
2023-01-17T06:43:43.350Z [info] : Mosaic Currency Id: 268CF9B2D33FBD22
2023-01-17T06:43:43.350Z [info] : Mosaic Harvest Id: 4C39D26C386E3182
2023-01-17T06:43:43.350Z [info] : Start Account Info
2023-01-17T06:43:43.365Z [info] : test1 Account: {
  "publicKey": "CB51613497A40D9A256B17932579BC64D5037A04B29737B944965C1ADADD6E04",
  "address": "MAS36UGDCOGG6GYCBDPX3ROISABSNPZ6JQXMJSA",
  "mosaics": [
    {
      "id": "3D86C9FE5D52DE6F",
      "amount": "1",
      "currency": false,
      "harvest": false
    }
  ],
  "keylink": {
    "vrf": {
      "publicKey": ""
    },
    "voting": {
      "publicKey": "",
      "startEpoch": "",
      "endEpoch": ""
    }
  }
}
```

Mosaic を転送する

先程作成した Mosaic **3D86C9FE5D52DE6F** を **test1** から **test2** に転送します。

```
$ mijin-catapult-tools transaction transfer -r mijin.json -f test1 -d test2 -m
3D86C9FE5D52DE6F -a 1
2023-01-17T06:44:46.983Z [info] : mijin URL: http://localhost:3000
2023-01-17T06:44:46.984Z [info] : Network: 96
2023-01-17T06:44:46.984Z [info] : Start Transfer Account...
2023-01-17T06:44:46.996Z [info] : From Account Address:
MAS36UGDCOGG6GYCBDPX3ROISABSNPZ6JQXMJSA
2023-01-17T06:44:46.996Z [info] : Dest Account Address:
MC5AH4UGBPPHNCMPTVNSY6LRDPIMEDODS2O373A
2023-01-17T06:44:47.005Z [info] : Start Transfer Transaction...
2023-01-17T06:44:52.197Z [info] : End Transfer Transaction
2023-01-17T06:44:52.197Z [info] : http://localhost:3000/transactionStatus/
↪454DFDC48F00852A3DE355D7D2AD4B581D718D999BA177E1BB392AECD1656C6A
2023-01-17T06:44:52.197Z [info] : http://localhost:3000/transactions/confirmed/
↪454DFDC48F00852A3DE355D7D2AD4B581D718D999BA177E1BB392AECD1656C6A
```

test1 アカウントの状態を確認します。
Mosaic **3D86C9FE5D52DE6F** を持っていないことを確認できます。

```
$ mijin-catapult-tools account info -r mijin.json -t test1
2023-01-17T06:45:51.931Z [info] : mijin URL: http://localhost:3000
2023-01-17T06:45:51.931Z [info] : Network: 96
2023-01-17T06:45:51.931Z [info] : Mosaic Currency Id: 268CF9B2D33FBD22
2023-01-17T06:45:51.931Z [info] : Mosaic Harvest Id: 4C39D26C386E3182
2023-01-17T06:45:51.931Z [info] : Start Account Info
2023-01-17T06:45:51.946Z [info] : test1 Account: {
  "publicKey": "CB51613497A40D9A256B17932579BC64D5037A04B29737B944965C1ADADD6E04",
  "address": "MAS36UGDCOGG6GYCBDPX3ROISABSNPZ6JQXMJSA",
  "mosaics": [],
  "keylink": {
    "vrf": {
      "publicKey": ""
    },
    "voting": {
      "publicKey": "",
      "startEpoch": "",
      "endEpoch": ""
    }
  }
}
```

続いて、**test2 アカウント**の状態を確認します。
Mosaic **3D86C9FE5D52DE6F** を所有していることが確認できます。

```
$ mijin-catapult-tools account info -r mijin.json -t test2
2023-01-17T06:46:41.737Z [info] : mijin URL: http://localhost:3000
2023-01-17T06:46:41.738Z [info] : Network: 96
2023-01-17T06:46:41.738Z [info] : Mosaic Currency Id: 268CF9B2D33FBD22
2023-01-17T06:46:41.738Z [info] : Mosaic Harvest Id: 4C39D26C386E3182
2023-01-17T06:46:41.738Z [info] : Start Account Info
2023-01-17T06:46:41.754Z [info] : test1 Account: {
  "publicKey": "0000000000000000000000000000000000000000000000000000000000000000",
  "address": "MC5AH4UGBPPHNCMPTVNSY6LRDPIMEDODS2O373A",
  "mosaics": [
    {
      "id": "3D86C9FE5D52DE6F",
      "amount": "1",
      "currency": false,
      "harvest": false
    }
  ],
  "keylink": {
    "vrf": {
      "publicKey": ""
    },
    "voting": {
      "publicKey": "",
      "startEpoch": "",
      "endEpoch": ""
    }
  }
}
```

3.2.1.5 (番外) 手数料ありモード時の、基軸通貨の移動方法

手数料ありモードで mijin を作成した場合、トランザクション送信などに全て基軸通貨から手数料が取られるようになります。
そのため、アカウント作成時に基軸通貨の残高を送信する手順を説明します。

基軸通貨の保持アカウント登録

注釈:

AWS Marketplace の | mijin | を使用した場合、AWS Cloudformation Stack 内にある Outouts タブを選択します。
Key「mijinLBEndpint」又は「mijinbEndpoint」の URL を控えてください。
次に、Key「HarvestAddress」の Value の URL をクリックしてください。
api の中にある一番はじめの Vaule にある private_key の値を控えてください。(ここでは 055E で始まる値)
詳しくは、既存 VPC 上に、mijin をデプロイする 又は新規 VPC を作成し、mijin をデプロイする を参照してください。

---

mijin-catapult-cli にて上記の private_key を import します。すでに mijin.json ファイルがある場合は、balanceAccount のみ追記することができます。

```
$ mijin-catapult-tools account generate -r mijin.json -w mijin.json -s -p
90EEBCB77A767F8F5CCCE9D0F89A60CB2D7FCD5FD8F469E2F8BDFC0CDD8B8A2F
2023-01-17T06:55:52.167Z [info] : mijin URL: http://localhost:3000
2023-01-17T06:55:52.167Z [info] : Network: 96
2023-01-17T06:55:52.167Z [info] : Mosaic Currency Id: 268CF9B2D33FBD22
2023-01-17T06:55:52.167Z [info] : Mosaic Harvest Id: 4C39D26C386E3182
2023-01-17T06:55:52.167Z [info] : Start Account Generate...
2023-01-17T06:55:52.229Z [info] : Write Config File: mijin.json
2023-01-17T06:55:52.233Z [info] : New Account: {
  "url": "http://localhost:3000",
  "workAccount": {
    "publicKey": "425F06A8870381A00BD83E2D1083BB690F9FCB815F0919DFFA1A53A68E144D06",
    "privateKey": "9E9C660164AB344030DF5E77539952D9F5B380311C978369BA08923F577D8DAB",
    "address": "MC5VTRZO7IGJUG2VPQXBD4GDO4A74YLFUCVZZGI"
  },
  "balanceAccount": {
    "publicKey": "4FAAC9BF9881893CB31BC2065E8A8D0B12364423E2A08DAF4F77E1FEF5D5B2E8",
    "privateKey": "90EEBCB77A767F8F5CCCE9D0F89A60CB2D7FCD5FD8F469E2F8BDFC0CDD8B8A2F",
    "address": "MCPYNELDE5QS63QBRZ2L7OLNZ63YPQJSOCEWXQI"
  },
  "mainAccount": {
    "publicKey": "F2985009341A526B17E954EB6EA3EC77E1A0B33AE31EB29F86A69D9BB283AF67",
    "privateKey": "FF9A552ED35D930378F1E6B349A677771F19A06BCA0C4D5DCA621F354F5C6956",
    "address": "MCV66SABR6MBWT2I56YI3ULCPYWREJFX5DHHGDQ"
  },
  "keylink": {
    "vrf": {
      "publicKey": "C074A57A0EDF633414DDD764C6771926E91596481E0C2CAD4D9C3EFAAA432BCE",
      "privateKey": "9E9C660164AB344030DF5E77539952D9F5B380311C978369BA08923F577D8DAB",
      "address": "MBLYBGODVGLT3HED6EXAQAXJE4CHHVRWGIAS45Y"
    },
```

(continues on next page)

(continued from previous page)

```
    "voting": {
      "publicKey": "F6571CBC420A4EBA09E027AA53E5DD9486642911CDC11E0D3A6D1B2E4BC228D1",
      "privateKey": "FD5744779348F77F6EA6288232D0C8944E2F3C9E24D2E0FB1E93A8C4F9DD9BA5",
      "address": "MD2M6SDIE64O6ZWF2IJ7R4R5RDKOT2FNJDUIYOQ"
    }
  },
  "test1Account": {
    "publicKey": "CB51613497A40D9A256B17932579BC64D5037A04B29737B944965C1ADADD6E04",
    "privateKey": "8D5969EF1796F5F90256C92B5017396E40786ED87995169D4E26C9E5E01D7F8C",
    "address": "MAS36UGDCOGG6GYCBDPX3ROISABSNPZ6JQXMJSA"
  },
  "test2Account": {
    "publicKey": "5CC14799D5B2643914C8E574C8D073A7EE9AE2A405F1339A53612B566498AB1D",
    "privateKey": "FDE625C1D53AF04533FEB06A3556679FC02C4C6246952D3A534EB3E19CF83C56",
    "address": "MC5AH4UGBPPHNCMPTVNSY6LRDPIMEDODS2O373A"
  }
}
```

これで balanceAccount にインポートすることができました。
アカウントの情報を確認します。
このアカウントでは、初期時２つの Mosaic を持っています。
Balance Information の 268CF9B2D33FBD22 が基軸通貨 (cat.currency) となり、4C39D26C386E3182 は Harvest 有効用の Mosaic です。

警告: この ID は mijin を作成毎に違う値になり、基軸通貨は"currency": true と表示された ID となります。

```
$ mijin-catapult-tools account info -r mijin.json -t balance
2023-01-17T06:56:51.299Z [info] : mijin URL: http://localhost:3000
2023-01-17T06:56:51.299Z [info] : Network: 96
2023-01-17T06:56:51.299Z [info] : Mosaic Currency Id: 268CF9B2D33FBD22
2023-01-17T06:56:51.299Z [info] : Mosaic Harvest Id: 4C39D26C386E3182
2023-01-17T06:56:51.299Z [info] : Start Account Info
2023-01-17T06:56:51.315Z [info] : balance Account: {
  "publicKey": "4FAAC9BF9881893CB31BC2065E8A8D0B12364423E2A08DAF4F77E1FEF5D5B2E8",
  "address": "MCPYNELDE5QS63QBRZ2L7OLNZ63YPQJSOCEWXQI",
  "mosaics": [
    {
      "id": "268CF9B2D33FBD22",
      "amount": "8998977498000000",
      "currency": true,
      "harvest": false
    },
    {
      "id": "4C39D26C386E3182",
      "amount": "15000000",
      "currency": false,
      "harvest": true
    }
  ],
```

(continues on next page)

(continued from previous page)

```
  "keylink": {
    "vrf": {
      "publicKey": "2A88BA2689D584B03A3D4B829347F0A8B63AF55A8E9F176F1D2327F9E87E22D8"
    },
    "voting": {
      "publicKey": "22BB9DCA05D483E4D4DDE764E8742E741ADA676F461D5F3E6663840C5290320F",
      "startEpoch": 1,
      "endEpoch": 26280
    }
  }
}
```

3.2.2 mijin Catapult(v.2) のバージョンアップ

本章では、mijin Catapult(v.2) ノードのマイナーバージョンアップ方法を説明します。
mijin Catapult(v.2) のプログラムは、docker コンテナのバージョンを更新、必要であれば Config ファイルを更新することでバージョンアップすることが可能です。
バージョンアップのアナウンスは、有償サポートのお客様向けに発信しています。

警告:

メジャーバージョンアップや、バージョンの差異がある場合は、同期に失敗する可能性があります。
2022/9/20 時点バージョンアップはございません。

3.2.2.1 Step.1

ノードにリモートログインします。
AWS MarketPlace の mijin Catapult(v.2) にログインしたい場合は、mijin Catapult(v.2) EC2 インスタンスログイン方法 を参照してください。

3.2.2.2 Step.2

mijin を起動している「catapult」ユーザーにスイッチします。

```
$ sudo su - catapult
catapult@api1:~$
```

3.2.2.3 Step.3

mijin の起動ファイルがあるディレクトリに移動します。
API,PEER ノードでディレクトリが違うことに注意してください。

ノード	ディレクトリ
API/Dual	mijin-catapult-package/package/**api**/catapult/
PEER	mijin-catapult-package/package/**peer**/catapult/

API ノードの場合は、以下のように移動します。

```
catapult@api1:~$ cd mijin-catapult-package/package/api/catapult/
catapult@api1:~/mijin-catapult-package/package/api/catapult$
```

3.2.2.4 Step.4

mijin では docker による複数コンテナを起動しています。
複数のコンテナを確認します。

API ノードの場合

４つのコンテナが起動しています。全ての State が Up であることを確認してください。何かしら異常がある場合は、Up`ではなく `Exit になり、ダウンした状態になります。

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ docker-compose ps
          Name                        Command               State           Ports
-------------------------------------------------------------------------------------
 ↪-----
catapult_api-node-broker_1   bash -c /bin/bash /scripts ...   Up
catapult_api-node_1          bash -c perl /scripts/wait ...   Up      0.0.0.0:7900->
 ↪7900/tcp
catapult_db_1                docker-entrypoint.sh bash  ...   Up      27017/tcp
catapult_rest-gateway_1      docker-entrypoint.sh ash - ...   Up      0.0.0.0:3000->
 ↪3000/tcp
```

PEER ノードの場合

１つのコンテナが起動しています。
全ての State が Up であることを確認してください。
何かしら異常がある場合は、Up`ではなく`Exit になり、ダウンした状態になります。

```
catapult@peer1:~/mijin-catapult-package/package/peer/catapult$ docker-compose ps
        Name                       Command               State           Ports
-------------------------------------------------------------------------------------
catapult_peer-node_1   bash -c /bin/bash /scripts ...   Up      0.0.0.0:7900->7900/tcp
```

3.2.2.5 Step.5

docker-compose を使って、mijin コンテナすべてを停止します。

コンテナ全停止

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ docker-compose down
Stopping catapult_rest-gateway_1 ... done
Stopping catapult_api-node_1     ... done
Stopping catapult_db_1           ... done
Removing catapult_api-node-broker_1 ... done
Removing catapult_rest-gateway_1    ... done
Removing catapult_api-node_1        ... done
Removing catapult_db_1              ... done
Removing network catapult_default
```

停止後、ロックファイルを確認し、あれば削除する

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ ls -la /mnt/mijin/blocks/
↪data/*.lock
---------- 1 catapult catapult 0 Jul 14 02:17 /mnt/mijin/blocks/data/broker.lock
---------- 1 catapult catapult 0 Jul 14 02:17 /mnt/mijin/blocks/data/server.lock
```

上記のように停止後にロックファイルが存在する場合、異常停止したと考えられます。
そのため、ロックファイルを削除します。

```
$ rm -rf /mnt/mijin/blocks/data/broker.lock /mnt/mijin/blocks/data/server.lock
```

3.2.2.6 Step.6

docker-compose ファイルを修正します。
ここでは例として、**1.0.0.0** から **1.0.0.1** へ置き換えます。

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ sed -i -e s/gcc-1.0.0.0/
↪gcc-1.0.0.1/g docker-compose.yml
```

注釈:

Config ファイルの修正が必要な場合があります。
バージョンアップのアナウンス時に手順を公開します。(2022/9/20 時点バージョンなし)

3.2.2.7 Step.7

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ docker-compose up -d
Creating network "catapult_default" with the default driver
Creating catapult_db_1 ... done
Creating catapult_rest-gateway_1    ... done
Creating catapult_api-node-broker_1 ... done
Creating catapult_api-node_1        ... done
```

起動後、Step.4 の項目を実行し、すべてのコンテナが Up になっていることを確認してください。

3.2.3 [アーカイブ] mijin アカウント作成 (>=1.0.0.0)

本章では、mijin を操作するために、必ず必要となるアカウント操作について説明します。

警告:

symbol-cli は アーカイブ化 されたため、symbol-cli は使用できない可能性があります。
1.0.3.4 以降は、mijin アカウント作成 を参照してください。

注釈:

mijin Catapult(v.2) を操作するにはアカウントを作成し、そのアカウントを使ってトランザクションを送信します。
また、デプロイ時の指定で手数料ありモードの場合は、アカウントに基軸通貨 (cat.currency) の残高を持つ必要があることに注意してください。

3.2.3.1 symbol-cli のインストール

npm より symbol-cli をインストールします。

```
$ sudo npm i -g symbol-cli@1.0.0
/usr/local/bin/symbol-cli -> /usr/local/lib/node_modules/symbol-cli/bin/symbol-cli
+ symbol-cli@1.0.0
updated 1 package in 8.724s
```

3.2.3.2 アカウントの作成

まずは、テストで使用する２つのアカウントを作成してみましょう。

項目	説明	値
Select the network type	ネットワークを指定して下さい。 構築時に指定した CatapultNetwork の値	MIJIN/MIJIN_TEST
Do you want to save the account?	このアカウントを保存します。	yes
Select an import type	再度保存するためのインポート方法を指定します。	PrivateKey
Enter the Symbol node URL.	Cloudformation Stack 内にある Outouts タブの mijinLBENdpoint または mijinEndpoint の URL を指定します。	<http://xxxxxxx:300>
Enter a profile name	アカウントを呼び出すプロファイル名を指定します。	任意
Enter your wallet password	アカウントのパスワードを指定します	任意

一つ目のアカウント (Profile mijin-a) 作成

```
$ symbol-cli account generate

? Select the network type: › - Use arrow-keys. Return to submit.
✓ Select the network type: › MIJIN
✓ Do you want to save the account? ··· yes
✓ Select an import type: › PrivateKey
✓ Enter the Symbol node URL. (Example: http://localhost:3000): ··· http://
↪xxxxxxxxxxxxxxxxxxxx.elb.ap-northeast-1.amazonaws.com:3000
✓ Enter a profile name: ··· mijin-a
✓ Enter your wallet password: ··· ********
✓ Do you want to set the account as the default profile? ··· yes

Account
┌─────────────┬──────────────────────────────────────────────────────────────────┐
│ Property    │ Value                                                            │
├─────────────┼──────────────────────────────────────────────────────────────────┤
│ Address     │ MA36BR-7DCFZT-65BQZP-TM5QND-EZKSB7-HNE4DU-6TI                    │
├─────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key  │ 707902962A0A2E32226243D1E7B98D2DD40261E9D3649543E7C28A0F024D4A38 │
├─────────────┼──────────────────────────────────────────────────────────────────┤
│ Private Key │ 2515EDCAAA3985F30D6E758ED139823290DAB11034BF4113849FF5CB9355B9C9 │
├─────────────┼──────────────────────────────────────────────────────────────────┤
```

(continues on next page)

(continued from previous page)

```
| Password       | Test1234                                                                 |
└────────────────┴──────────────────────────────────────────────────────────────────────────┘

SUCCESS Stored mijin-a profile
```

この時点で mijin-a のアカウントを確認しても、存在しないことになっています。
ブロックチェーン上で、このアドレスのやり取りがないため、存在を確認できないからです。

```
$ symbol-cli account info --profile mijin-a
⠸ Processing(node:7718) [DEP0091] DeprecationWarning: crypto.DEFAULT_ENCODING is
deprecated.
(node:7718) [DEP0010] DeprecationWarning: crypto.createCredentials is deprecated. Use
tls.createSecureContext instead.
(node:7718) [DEP0011] DeprecationWarning: crypto.Credentials is deprecated. Use tls.
↪SecureContext instead.

ERR {"statusCode":404,"statusMessage":"Not Found","body":"{\"code\":\"ResourceNotFound\
↪",\"message\":\"no resource exists with id 'MA36BR7DCFZT65BQZPTM5QNDEZKSB7HNE4DU6TI'\
↪"}"}

TIP The account has to receive at least one transaction to be recorded on the network
```

2つ目のアカウント (Profile mijin-b) 作成

```
$ symbol-cli account generate

✓ Select the network type: › MIJIN
✓ Do you want to save the account? ⋯ yes
✓ Select an import type: › PrivateKey
✓ Enter the Symbol node URL. (Example: http://localhost:3000): ⋯ http://
↪xxxxxxxxxxxxxxxxxxxx.elb.ap-northeast-1.amazonaws.com:3000
✓ Enter a profile name: ⋯ mijin-b
✓ Enter your wallet password: ⋯ ********
✓ Do you want to set the account as the default profile? ⋯ no

Account
┌────────────────┬──────────────────────────────────────────────────────────────────────────┐
| Property       | Value                                                                    |
├────────────────┼──────────────────────────────────────────────────────────────────────────┤
| Address        | MCLO63-LBWG6V-PLJD4O-MADZ37-W6QXQE-DPC3H3-EGQ                            |
├────────────────┼──────────────────────────────────────────────────────────────────────────┤
| Public Key     | 2D2AC0FF30FABEFC12CB3FBB2323F8CD079ED1055FAAF2581CA29697130292FA         |
├────────────────┼──────────────────────────────────────────────────────────────────────────┤
| Private Key    | 654065E33D00446F1FAAF2CF7D72CC287BDD91E55E9489AEC42769EDDB7A9759         |
├────────────────┼──────────────────────────────────────────────────────────────────────────┤
| Password       | Test1234                                                                 |
└────────────────┴──────────────────────────────────────────────────────────────────────────┘

SUCCESS Stored mijin-b profile
```

1つ目と同じで、mijin-b アカウントは存在しません。

```
$ symbol-cli account info --profile mijin-b

⠸ Processing(node:53) [DEP0091] DeprecationWarning: crypto.DEFAULT_ENCODING is
deprecated.
(node:53) [DEP0010] DeprecationWarning: crypto.createCredentials is deprecated. Use
tls.createSecureContext instead.
(node:53) [DEP0011] DeprecationWarning: crypto.Credentials is deprecated. Use tls.
↪SecureContext instead.

ERR {"statusCode":404,"statusMessage":"Not Found","body":"{\"code\":\"ResourceNotFound\
↪",\"message\":\"no resource exists with id 'MCLO63LBWG6VPLJD4OMADZ37W6QXQEDPC3H3EGQ'\
↪"}"}

TIP The account has to receive at least one transaction to be recorded on the network
```

手数料について

手数料とは、以下の項目で発行するトランザクションによって手数料が発生し、基軸通貨 (cat.currency) にて支払う必要があります。

- トランザクション手数料
- Mosaic レンタル手数料
- Namespace レンタル手数料
- ロック保証金

マーケットプレイスで提供する mijin では手数料モードは標準で なしに設定されています。
手数料なしでは、アカウントに基軸通貨の残高を持つ必要がありませんので特に意識することはありませんが、symbol-cli や sdk にて手数料設定を都度 0 を設定する必要があります。
手数料ありで mijin を作成した場合は、「手数料ありモード時の、基軸通貨の移動方法」の項目を参照し、Profile mijin-a に基軸通貨を送信してください。

3.2.3.3 Mosaic を作成し転送

まずは基本となる Mosaic を作成し、転送してみましょう。

Mosaic の作成

Profile mijin-a のアカウントにて 1amount の Mosaic(Token) を発行します。

項目	説明	値
Enter your wallet password	設定したパスワードを指定してください	任意
Do you want a non-expiring mosaic	Mosaic の期限を指定するか、無期限かを指定します。Yes で無期限にします。	yes
Enter the mosaic divisibility	Mosaic の可分性を指定します。0 でなしにします。	0
Do you want this mosaic to have a mutable supply?	Mosaic の最大発行数を変更できるようにするかを指定します。なしにします。	no
Do you want this mosaic to be transferable?	Mosaic の転送を許可するか指定します。	yes
Do you want this mosaic to be restrictable?	Mosaic の制限を許可するか指定します。	yes
Amount of mosaics units to create	Mosaic の発行数を指定します。ここでは 1 で発行します、	1
Enter the maximum fee (absolute amount)	トランザクション手数料を指定します。 これは手数料モードで変わります。 手数料あり 20000 程度 (0.2cat.currency) 手数料なし 0	0
Select the transaction announce mode	トランザクションをアナウンスする方法を指定します。	normal

Listing 1: Symbol CLI によるモザイク定義

```
$ symbol-cli transaction mosaic --profile mijin-a

✓ Enter your wallet password: ⋯ ********
✓ Do you want a non-expiring mosaic? ⋯ yes
✓ Enter the mosaic divisibility: ⋯ 0
✓ Do you want this mosaic to have a mutable supply? ⋯ no
✓ Do you want this mosaic to be transferable? ⋯ yes
✓ Do you want this mosaic to be restrictable? ⋯ yes
✓ Amount of mosaics units to create:  ⋯ 1
✓ Enter the maximum fee (absolute amount): ⋯ 0
✓ Select the transaction announce mode: › normal
✓ Do you want to announce this transaction? ⋯ yes

SUCCESS Transaction announced correctly

TIP To check if the network confirms or rejects the transaction, run the command
↪'symbol-cli transaction status'
```

再度 `mijin-a`のアカウントを確認すると、Balance Information にて、MosaicId 3BF3AF8B22CB53D8 を 1 だけ持っていることが確認できます。

```
$ symbol-cli account info --profile mijin-a

⠏ Processing(node:7795) [DEP0091] DeprecationWarning: crypto.DEFAULT_ENCODING is
deprecated.
(node:7795) [DEP0010] DeprecationWarning: crypto.createCredentials is deprecated. Use
tls.createSecureContext instead.
(node:7795) [DEP0011] DeprecationWarning: crypto.Credentials is deprecated. Use tls.
↪SecureContext instead.
⠇ Processing
Account Information
```

(continues on next page)

(continued from previous page)

```
┌──────────────────┬──────────────────────────────────────────────────────────────────┐
│ Property         │ Value                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address          │ MA36BR-7DCFZT-65BQZP-TM5QND-EZKSB7-HNE4DU-6TI                    │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address Height   │ 959                                                              │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key       │ 707902962A0A2E32226243D1E7B98D2DD40261E9D3649543E7C28A0F024D4A38 │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key Height │ 959                                                             │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance       │ 0                                                                │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance Height │ 0                                                               │
└──────────────────┴──────────────────────────────────────────────────────────────────┘
Balance Information
┌──────────────────┬─────────────────┬─────────────────┬─────────────────┐
│ Mosaic Id        │ Relative Amount │ Absolute Amount │ Expiration Height │
├──────────────────┼─────────────────┼─────────────────┼─────────────────┤
│ 3BF3AF8B22CB53D8 │ 1               │ 1               │ Never           │
└──────────────────┴─────────────────┴─────────────────┴─────────────────┘
```

Mosaic を転送する

先程作成した Mosaic 3BF3AF8B22CB53D8 を Profile mijin-a から mijin-b に転送します。

項目	値
Mosaic	3BF3AF8B22CB53D8
転送 amount	1
転送先アドレス (Profile mijin-b のアドレス)	MCLO63-LBWG6V-PLJD4O-MADZ37-W6QXQE-DPC3H3-EGQ

項目	説明	値
Enter your wallet password	設定したパスワードを指定してください	任意
Mosaics to transfer in the format (mosaicId(hex)|@aliasName)::absoluteAmount	転送するモザイク ID（またはエイリアス）にコロン2つを追加した後、転送 amount を指定します。	**3BF3AF8B22CB53D8::1**
Enter the recipient address or @alias	転送先アドレスを指定します	MCLO63-LBWG6V-PLJD4O-MADZ37-W6QXQE-DPC3H3-EGQ
Enter a message	転送トランザクションにメッセージを追加できます	任意
Enter the maximum fee (absolute amount)	トランザクション手数料を指定します（モードにより異なります）。 - 手数料あり: 約``20000`` (= 0.2cat.currency) - 手数料なし: `0`	0
Select the transaction announce mode	トランザクションをアナウンスする方法を指定します。	normal

```
$ symbol-cli transaction transfer --profile mijin-a

? Enter your wallet password: › (node:97) [DEP0091] DeprecationWarning: crypto.DEFAULT_
```

(continues on next page)

(continued from previous page)

```
↪ENCODING is deprecated.
(node:97) [DEP0010] DeprecationWarning: crypto.createCredentials is deprecated. Use
tls.createSecureContext instead.
(node:97) [DEP0011] DeprecationWarning: crypto.Credentials is deprecated. Use tls.
↪SecureContext instead.
✓ Enter your wallet password: ⋯ ********
✓ Mosaics to transfer in the format (mosaicId(hex)|@aliasName)::absoluteAmount, (Ex:
sending 1 symbol.xym, @symbol.xym::1000000). Add multiple mosaics separated by commas:
⋯ 3BF3AF8B22CB53D8::1
✓ Enter the recipient address or @alias: ⋯ MCLO63-LBWG6V-PLJD4O-MADZ37-W6QXQE-DPC3H3-
↪EGQ
✓ Enter a message: ⋯
✓ Enter the maximum fee (absolute amount): ⋯ 0
✓ Select the transaction announce mode: › normal
```

TRANSFER	
Max fee:	0
Network type:	MIJIN
Deadline:	2021-05-17 16:40:45.212
Recipient:	MCLO63-LBWG6V-PLJD4O-MADZ37-W6QXQE-DPC3H3-EGQ
Message:	N/A
Mosaic (1/1):	1 3BF3AF8B22CB53D8
Signature details	
Payload:	B000000000000000639B3F893989DC69FA3DF8D9BA294FD787F7644D68918FA8 F5E6A5162FE57CD34B03E944AA48F4B22790A50ECEA4A130EBB89299BDB173A4 556BBAFF8092A20B707902962A0A2E32226243D1E7B98D2DD40261E9D3649543 E7C28A0F024D4A380000000001605441000000000000000009C1F80300E000000 6096EF6D61B1BD57AD23E39801E77FB7A178106F16CFB21A0000010000000000 D853CB228BAFF33B0100000000000000
Hash:	0CAB966B0E7090AA19AE4D4F2BD2334A7F7466E5661107A15F8831EA48A5CE88
Signer:	707902962A0A2E32226243D1E7B98D2DD40261E9D3649543E7C28A0F024D4A38

```
✓ Do you want to announce this transaction? ⋯ yes

SUCCESS Transaction announced correctly

TIP To check if the network confirms or rejects the transaction, run the command
↪'symbol-cli transaction status'
```

Profile **mijin-a** の状態を確認します。
Mosaic **3BF3AF8B22CB53D8** を持っていないことを確認できます。

```
symbol-cli account info --profile mijin-a
```

(continues on next page)

(continued from previous page)

```
⠏ Processing(node:108) [DEP0091] DeprecationWarning: crypto.DEFAULT_ENCODING is
deprecated.
(node:108) [DEP0010] DeprecationWarning: crypto.createCredentials is deprecated. Use
tls.createSecureContext instead.
(node:108) [DEP0011] DeprecationWarning: crypto.Credentials is deprecated. Use tls.
↪SecureContext instead.
⠇ Processing
Account Information
┌──────────────────┬──────────────────────────────────────────────────────────────────┐
│ Property         │ Value                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address          │ MA36BR-7DCFZT-65BQZP-TM5QND-EZKSB7-HNE4DU-6TI                    │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address Height   │ 959                                                              │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key       │ 707902962A0A2E32226243D1E7B98D2DD40261E9D3649543E7C28A0F024D4A38 │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key Height │ 959                                                             │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance       │ 0                                                                │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance Height │ 0                                                               │
└──────────────────┴──────────────────────────────────────────────────────────────────┘
```

続いて、Profile **mijin-b** の状態を確認します。
Mosaic **3BF3AF8B22CB53D8** を所有していることが確認できます。

```
symbol-cli account info --profile mijin-b
⠏ Processing(node:119) [DEP0091] DeprecationWarning: crypto.DEFAULT_ENCODING is
deprecated.
(node:119) [DEP0010] DeprecationWarning: crypto.createCredentials is deprecated. Use
tls.createSecureContext instead.
(node:119) [DEP0011] DeprecationWarning: crypto.Credentials is deprecated. Use tls.
↪SecureContext instead.
⠇ Processing
Account Information
┌──────────────────┬──────────────────────────────────────────────────────────────────┐
│ Property         │ Value                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address          │ MCLO63-LBWG6V-PLJD4O-MADZ37-W6QXQE-DPC3H3-EGQ                    │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address Height   │ 17912                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key       │ 0000000000000000000000000000000000000000000000000000000000000000 │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key Height │ 0                                                               │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance       │ 0                                                                │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance Height │ 0                                                               │
└──────────────────┴──────────────────────────────────────────────────────────────────┘
Balance Information
┌──────────────────┬──────────────────┬──────────────────┬──────────────────┐
```

(continues on next page)

(continued from previous page)

```
| Mosaic Id          | Relative Amount | Absolute Amount | Expiration Height |
|--------------------|-----------------|-----------------|-------------------|
| 3BF3AF8B22CB53D8 | 1               | 1               | Never             |
└────────────────────┴─────────────────┴─────────────────┴───────────────────┘
```

3.2.3.4 (番外) 手数料ありモード時の、基軸通貨の移動方法

手数料ありモードで mijin を作成した場合、トランザクション送信などに全て基軸通貨から手数料が取られるようになります。
そのため、アカウント作成時に基軸通貨の残高を送信する手順を説明します。

基軸通貨の保持アカウント登録

注釈:

AWS Marketplace の | mijin | を使用した場合、AWS Cloudformation Stack 内にある Outouts タブを選択します。
Key「mijinLBEndpint」又は「mijinbEndpoint」の URL を控えてください。
次に、Key「HarvestAddress」の Value の URL をクリックしてください。
api の中にある一番はじめの Vaule にある private_key の値を控えてください。(ここでは 055E で始まる値)
詳しくは、既存 VPC 上に、mijin をデプロイする 又は新規 VPC を作成し、mijin をデプロイする を参照してください。

---

symbol-cli にて上記の private_key を import します。

項目	入力値
Select the network type	ネットワークを指定して下さい。 構築時に指定した `CatapultNetwork` の値 `MIJIN` または `MIJIN_TEST`
Enter the Symbol node URL	控えていた `mijinEndpoint` もしくは ロードバランサーが有効であれば `mijinLBEndpoint` の URL を入力
Enter a profile name	任意のプロファイル名を入力
Enter your wallet password	任意のパスワードを入力
Do you want to set the account as the default profile	DefaultProfile にするかどうか、ここでは `Yes` を選択
Select an import type	`PrivateKey`
Enter your account private key	控えていた `private_key` を入力

```
$ symbol-cli profile import

✓ Select the network type: › MIJIN
✓ Enter the Symbol node URL. (Example: http://localhost:3000): … http://MIJIN-
↪CATAPULT-E1-nlb-rest-XXXXXXXXX.elb.ap-northeast-1.amazonaws.com:3000
```

(continues on next page)

(continued from previous page)

```
✓ Enter a profile name: ⋯ mijin-harvest
✓ Enter your wallet password: ⋯ ********
✓ Do you want to set the account as the default profile? ⋯ no
✓ Select an import type: › PrivateKey
✓ Enter your account private key: ⋯
************************************************************

Account
┌─────────────┬──────────────────────────────────────────────────────────────────┐
│ Property    │ Value                                                            │
├─────────────┼──────────────────────────────────────────────────────────────────┤
│ Address     │ MAQUY5-KOJVPE-DDCTD6-3SZYHM-EQOFF4-HTUYZU-3WQ                    │
├─────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key  │ 29800CB9DF988622AD4B940F578569321F4B7F08127C478A0C0C28ACC61B8A2C │
├─────────────┼──────────────────────────────────────────────────────────────────┤
│ Private Key │ 2EC8FF52B5B922E0F509FBEE6CE3C4B3512E9347DB800A76A6EF993C43C0D5BC │
├─────────────┼──────────────────────────────────────────────────────────────────┤
│ Password    │ Test1234                                                         │
└─────────────┴──────────────────────────────────────────────────────────────────┘

SUCCESS Stored mijin-harvest profile
```

これでインポートすることができました。
アカウントの情報を確認します。
このアカウントでは、初期時２つの Mosaic を持っています。
Balance Information の 04A125F887094D2A が基軸通貨 (cat.currency) となり、49DB43B9FA374EF2 は Harvest 有効用の Mosaic です。

警告: この ID は作成毎に違う値になり、基軸通貨は Amount が大きい ID となります。

```
$ symbol-cli account info --profile mijin-harvest

⠏ Processing(node:141) [DEP0091] DeprecationWarning: crypto.DEFAULT_ENCODING is
deprecated.
(node:141) [DEP0010] DeprecationWarning: crypto.createCredentials is deprecated. Use
tls.createSecureContext instead.
(node:141) [DEP0011] DeprecationWarning: crypto.Credentials is deprecated. Use tls.
↪SecureContext instead.
⠇ Processing
Account Information
┌──────────────────┬──────────────────────────────────────────────────────────────────┐
│ Property         │ Value                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address          │ MAQUY5-KOJVPE-DDCTD6-3SZYHM-EQOFF4-HTUYZU-3WQ                    │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address Height   │ 1                                                                │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key       │ 29800CB9DF988622AD4B940F578569321F4B7F08127C478A0C0C28ACC61B8A2C │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key Height │ 1                                                               │
```

(continues on next page)

(continued from previous page)

```
┌──────────────────┬──────────────────────────────────────────────────────────────┐
│ Importance       │ 2850000                                                      │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Importance Height│ 18080                                                        │
└──────────────────┴──────────────────────────────────────────────────────────────┘
Balance Information
┌──────────────────┬──────────────────┬──────────────────┬──────────────────┐
│ Mosaic Id        │ Relative Amount  │ Absolute Amount  │ Expiration Height│
├──────────────────┼──────────────────┼──────────────────┼──────────────────┤
│ 04A125F887094D2A │ 1,799,799,999.6  │ 1799799999600000 │ Never            │
├──────────────────┼──────────────────┼──────────────────┼──────────────────┤
│ 49DB43B9FA374EF2 │ 3,000            │ 3000000          │ Never            │
└──────────────────┴──────────────────┴──────────────────┴──────────────────┘
```

基軸通貨の確認

項目	説明	値
Enter the mosaic id in hexadecimal format	モザイク ID を指定することで対象モザイクの情報を取得できます	ここでは 04A125F887094D2A

Divisibility にて可分性が確認できます。
以下では可分性が 6 となります。

```
symbol-cli mosaic info
? Enter the mosaic id in hexadecimal format: › (node:163) [DEP0091] DeprecationWarning:
crypto.DEFAULT_ENCODING is deprecated.
(node:163) [DEP0010] DeprecationWarning: crypto.createCredentials is deprecated. Use
tls.createSecureContext instead.
(node:163) [DEP0011] DeprecationWarning: crypto.Credentials is deprecated. Use tls.
↪SecureContext instead.
✓ Enter the mosaic id in hexadecimal format: … 04A125F887094D2A

Mosaic Information
┌──────────────────┬──────────────────────────────────────────────────────────────┐
│ Property         │ Value                                                        │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Record Id        │ 609DEB554D4B851AA429AE2C                                     │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Mosaic Id        │ 04A125F887094D2A                                             │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Divisibility     │ 6                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Transferable     │ true                                                         │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Supply Mutable   │ false                                                        │
├──────────────────┼──────────────────────────────────────────────────────────────┤
│ Height           │ 1                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────┤
```

(continues on next page)

(continued from previous page)

```
| Expiration          | Never                                                    |
├─────────────────────┼──────────────────────────────────────────────────────────┤
| Owner               | MAO5AR-GSMLGK-ZCDV35-IJWDVL-JFOCZT-XHM3KJ-RHA            |
├─────────────────────┼──────────────────────────────────────────────────────────┤
| Supply (Absolute)   | 8998999998000000                                         |
├─────────────────────┼──────────────────────────────────────────────────────────┤
| Supply (Relative)   | 8,998,999,998                                            |
└─────────────────────┴──────────────────────────────────────────────────────────┘
```

基軸通貨を送信する

登録したアカウント (Profile mijin-a) に対して、基軸通貨を 1000cat.currency を送信してみます。

項目	値
Mosaic	cat.currency
転送 amount	1000000000(可分性が 6 なので 1000.000000)
転送先アドレス (Profile mijin-a)	MA36BR-7DCFZT-65BQZP-TM5QND-EZKSB7-HNE4DU-6TI

項目	説明	値
Enter your wallet password	設定したパスワードを指定してください	任意
Mosaics to transfer in the format (mosaicId(hex)|@aliasName)::absoluteAmount)	転送するモザイク ID(またはエイリアス)にコロンを二つ追加した後、転送 amount を指定します。	@cat.currency::1000000000
Enter the recipient address or @alias	転送先アドレスを指定します	MA36BR-7DCFZT-65BQZP-TM5QND-EZKSB7-HNE4DU-6TI
Enter a message	転送トランザクションにメッセージを追加できます	任意
Enter the maximum fee (absolute amount)	トランザクション手数料を指定します。 これは手数料モードで変わります。 手数料あり 20000 程度 (0.2cat.currency) 手数料なし 0	0
Select the transaction announce mode	トランザクションをアナウンスする方法を指定します。	normal

ここではトランザクション手数料を 0 にしていますが、手数料ありの場合、トランザクション手数料がかかりますので 0 ではなく、200000 程度必要になります。
手数料の計算方法は、Symbol と同様となります。

<https://docs.symbol.dev/concepts/fees.html#transaction-fee>

```
$ symbol-cli transaction transfer --profile mijin-harvest

? Enter your wallet password: › (node:196) [DEP0091] DeprecationWarning: crypto.
↪DEFAULT_ENCODING is deprecated.
(node:196) [DEP0010] DeprecationWarning: crypto.createCredentials is deprecated. Use
tls.createSecureContext instead.
(node:196) [DEP0011] DeprecationWarning: crypto.Credentials is deprecated. Use tls.
```

(continues on next page)

(continued from previous page)

```
↪SecureContext instead.
✓ Enter your wallet password: ⋯ ********
✓ Mosaics to transfer in the format (mosaicId(hex)|@aliasName)::absoluteAmount, (Ex:
sending 1 symbol.xym, @symbol.xym::1000000). Add multiple mosaics separated by commas:
⋯ @cat.currency::1000000000
✓ Enter the recipient address or @alias: ⋯ MA36BR-7DCFZT-65BQZP-TM5QND-EZKSB7-HNE4DU-
↪6TI
✓ Enter a message: ⋯
✓ Enter the maximum fee (absolute amount): ⋯ 0
✓ Select the transaction announce mode: › normal
```

TRANSFER	
Max fee:	0
Network type:	MIJIN
Deadline:	2021-05-17 17:46:42.643
Recipient:	MA36BR-7DCFZT-65BQZP-TM5QND-EZKSB7-HNE4DU-6TI
Message:	N/A
Mosaic (1/1):	1,000,000,000 cat.currency (85BBEA6CC462B244)
Signature details	
Payload:	B00000000000000071E0E34240C563025C466C31BFA995D6DA4CADFFB0AD29A3 B2EFF280AD596BDA4CF5DC01B3654D9C34F56346B5DF41112EF27858C0ED1FCE CF409B5A450FC30E29800CB9DF988622AD4B940F578569321F4B7F08127C478A 0C0C28ACC61B8A2C0000000001605441000000000000000005382BC300E000000 6037E0C7E311733F7430CBE6CEC1A3265520FCED27074F4D0000010000000000 44B262C46CEABB8500CA9A3B00000000
Hash:	864EF99D58E8DA837879D85DE08DF29398766E04F967F09732A2FD02115469FB
Signer:	29800CB9DF988622AD4B940F578569321F4B7F08127C478A0C0C28ACC61B8A2C

```
✓ Do you want to announce this transaction? ⋯ yes

SUCCESS Transaction announced correctly

TIP To check if the network confirms or rejects the transaction, run the command
↪'symbol-cli transaction status'
```

Profile mijin-a のアカウントを確認します。
1000cat.currency を持っていることを確認できます。

```
symbol-cli account info --profile mijin
⠏ Processing(node:207) [DEP0091] DeprecationWarning: crypto.DEFAULT_ENCODING is
deprecated.
(node:207) [DEP0010] DeprecationWarning: crypto.createCredentials is deprecated. Use
tls.createSecureContext instead.
```

(continues on next page)

(continued from previous page)

```
(node:207) [DEP0011] DeprecationWarning: crypto.Credentials is deprecated. Use tls.
↪SecureContext instead.
⁝ Processing
Account Information
┌──────────────────┬──────────────────────────────────────────────────────────────────┐
│ Property         │ Value                                                            │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address          │ MA36BR-7DCFZT-65BQZP-TM5QND-EZKSB7-HNE4DU-6TI                    │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Address Height   │ 959                                                              │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key       │ 707902962A0A2E32226243D1E7B98D2DD40261E9D3649543E7C28A0F024D4A38 │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Public Key Height │ 959                                                             │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance       │ 0                                                                │
├──────────────────┼──────────────────────────────────────────────────────────────────┤
│ Importance Height │ 0                                                               │
└──────────────────┴──────────────────────────────────────────────────────────────────┘
Balance Information
┌──────────────┬─────────────────┬─────────────────┬─────────────────┐
│ Mosaic Id    │ Relative Amount │ Absolute Amount │ Expiration Height │
├──────────────┼─────────────────┼─────────────────┼─────────────────┤
│ 04A125F887094D2A │ 1,000       │ 1000000000      │ Never           │
└──────────────┴─────────────────┴─────────────────┴─────────────────┘
```

3.3 トラブルシューティング

mijin Catapult(v.2) のトラブルシューティングを纏めます。

3.3.1 mijin Catapult(v.2) のノード再同期

mijin Catapult(v.2) ノードが何らかの理由で同期が止まってしまったなどの場合の再同期手順を説明します。

3.3.1.1 対象

- ノードのブロックが進まない
- /node/peers で確認できなくなった
- docker-compose ps コマンドで一部コンテナが Exit になっており、復旧できない。

3.3.1.2 Step.1

ノードにリモートログインします。
AWS MarketPlace の mijin Catapult(v.2) にログインしたい場合は、mijin Catapult(v.2) EC2 インスタンスログイン方法 を参照してください。

3.3.1.3 Step.2

mijin を起動している「catapult」ユーザーにスイッチします。

```
$ sudo su - catapult
catapult@api1:~$
```

3.3.1.4 Step.3

mijin の起動ファイルがあるディレクトリに移動します。
API,PEER ノードでディレクトリが違うことに注意してください。

ノード	ディレクトリ
API/Dual	mijin-catapult-package/package/**api**/catapult/
PEER	mijin-catapult-package/package/**peer**/catapult/

API ノードの場合は、以下のように移動します。

```
catapult@api1:~$ cd mijin-catapult-package/package/api/catapult/
catapult@api1:~/mijin-catapult-package/package/api/catapult$
```

3.3.1.5 Step.4

mijin では docker による複数コンテナを起動しています。
複数のコンテナを確認します。

API ノードの場合

４つのコンテナが起動しています。全ての State が Up であることを確認してください。何かしら異常がある場合は、Up`ではなく `Exit になり、ダウンした状態になります。

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ docker-compose ps
          Name                        Command               State           Ports
-------------------------------------------------------------------------------------
 ↪-----
catapult_api-node-broker_1   bash -c /bin/bash /scripts ...   Up
catapult_api-node_1          bash -c perl /scripts/wait ...   Up      0.0.0.0:7900->
 ↪7900/tcp
catapult_db_1                docker-entrypoint.sh bash  ...   Up      27017/tcp
catapult_rest-gateway_1      docker-entrypoint.sh ash - ...   Up      0.0.0.0:3000->
 ↪3000/tcp
```

PEER ノードの場合

１つのコンテナが起動しています。
全ての State が Up であることを確認してください。
何かしら異常がある場合は、Up`ではなく`Exit になり、ダウンした状態になります。

```
catapult@peer1:~/mijin-catapult-package/package/peer/catapult$ docker-compose ps
        Name                       Command               State           Ports
-------------------------------------------------------------------------------------
catapult_peer-node_1   bash -c /bin/bash /scripts ...   Up      0.0.0.0:7900->7900/tcp
```

3.3.1.6 Step.5

docker-compose を使って、mijin コンテナすべてを停止し、再度起動します。

コンテナ全停止

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ docker-compose down
Stopping catapult_rest-gateway_1 ... done
Stopping catapult_api-node_1     ... done
Stopping catapult_db_1           ... done
Removing catapult_api-node-broker_1 ... done
Removing catapult_rest-gateway_1    ... done
Removing catapult_api-node_1        ... done
Removing catapult_db_1              ... done
Removing network catapult_default
```

停止後、ロックファイルを確認し、あれば削除する

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ ls -la /mnt/mijin/blocks/
↪data/*.lock
---------- 1 catapult catapult 0 Jul 14 02:17 /mnt/mijin/blocks/data/broker.lock
---------- 1 catapult catapult 0 Jul 14 02:17 /mnt/mijin/blocks/data/server.lock
```

上記のように停止後にロックファイルが存在する場合、異常停止したと考えられます。
そのため、ロックファイルを削除します。

```
$ rm -rf /mnt/mijin/blocks/data/broker.lock /mnt/mijin/blocks/data/server.lock
```

再度起動

```
$ docker-compose up -d
Creating network "catapult_default" with the default driver
Creating catapult_db_1 ... done
Creating catapult_rest-gateway_1    ... done
Creating catapult_api-node-broker_1 ... done
Creating catapult_api-node_1        ... done
```

起動後、Step.4 の項目を実行し、すべてのコンテナが Up になっていることを確認してください。
もし、同様な状態になっている場合は、Step.9 に進んでください。

3.3.1.7 Step.6

ノードのブロックデータをリセットし復旧します。冗長化されている状態であれば、データは自動で他ノードから取得し復旧することができます。

コンテナ全停止

```
$ docker-compose down
Stopping catapult_rest-gateway_1 ... done
Stopping catapult_api-node_1     ... done
Stopping catapult_db_1           ... done
Removing catapult_api-node-broker_1 ... done
Removing catapult_rest-gateway_1    ... done
Removing catapult_api-node_1        ... done
Removing catapult_db_1              ... done
Removing network catapult_default
```

ブロックデータ及び mongo データを削除

mijin が起動している catapult ユーザーには sudo 権限がなく、ディレクトリを削除することができません。
そのため、catapult ユーザーに sudo 権限を付与します。
この sudo 設定は初回のみ必要です。

```
# catapult ユーザーからログアウト Log out of catapult user
$ logout
# root ユーザーにスイッチ Switch to root user
$ sudo su -
# catapult ユーザーに sudo 権限を付与する Grant sudo privileges to the catapult user
# echo "catapult ALL=(ALL) NOPASSWD:ALL" > /etc/sudoers.d/catapult
# catapult ユーザーにスイッチ Switch to catapult user
# su - catapult
```

sudo 権限でブロックデータを削除します。

```
$ sudo rm -rf /mnt/mijin/blocks/data
```

API ノードが対象の場合は、mongo データの削除も必要です

```
$ sudo rm -rf /mnt/mijin/mongo/db
```

リカバリーコマンドの実行

docker-compose のあるファイルに移動し、recover スクリプトを実行します。

API ノードが対象の場合

```
catapult@api1:~$ cd mijin-catapult-package/package/api/catapult/
catapult@api1:~/mijin-catapult-package/package/api/catapult$
```

PEER ノードが対象の場合

```
catapult@peer1:~$ cd mijin-catapult-package/package/peer/catapult/
catapult@peer1:~/mijin-catapult-package/package/peer/catapult$
```

スクリプトの実行をします。(共通)

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ bash scripts/recover.sh
2021/07/14 02:52 Start: mijin Recovery
2021/07/14 02:52 Check: /home/catapult/mijin-catapult-package/package/api/catapult
2021/07/14 02:52 Check: /home/catapult/mijin-catapult-package/package/api/catapult OK
2021/07/14 02:52 Check: Started docker?
2021/07/14 02:52 Check: Started docker Stop OK
2021/07/14 02:52 Check: Block Directory
2021/07/14 02:52 Check: Block Directory Empty OK
2021/07/14 02:52 Start: Make Block Directory
2021/07/14 02:52 Check: mongo Directory
2021/07/14 02:52 Check: mongo Directory OK
2021/07/14 02:52 Start: Make mongo Directory
2021/07/14 02:52 Start: Create mongo Init Data
about to fork child process, waiting until server is ready for connections.
forked process: 10
child process started successfully, parent exiting
[+] Preparing db
MongoDB shell version v4.2.5
connecting to: mongodb://localhost:27017/catapult?compressors=disabled&
↪gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c296d026-dfdf-4ea2-ba03-91d7404e21c9") }
MongoDB server version: 4.2.5
Loading LockHash
Loading LockSecret
Loading Metadata
Loading Mosaic
Loading Multisig
Loading Namespace
Loading RestrictionAccount
Loading RestrictionMosaic
===== accountRestrictions INDEXES =====
{ "_id" : 1 }
{ "accountRestrictions.address" : 1 }
===== accounts INDEXES =====
```

(continues on next page)

(continued from previous page)

```
{ "_id" : 1 }
{ "account.publicKey" : 1 }
{ "account.address" : 1 }
===== addressResolutionStatements INDEXES =====
{ "_id" : 1 }
{ "statement.height" : 1, "statement.unresolved" : 1 }
===== blocks INDEXES =====
{ "_id" : 1 }
{ "block.signerPublicKey" : 1 }
{ "block.timestamp" : -1 }
{ "block.height" : -1 }
{ "block.type" : 1, "block.height" : -1 }
{ "block.signerPublicKey" : 1, "block.height" : -1 }
{ "block.beneficiaryAddress" : 1, "block.height" : -1 }
===== finalizedBlocks INDEXES =====
{ "_id" : 1 }
{ "block.finalizationEpoch" : -1 }
{ "block.height" : -1 }
===== hashLocks INDEXES =====
{ "_id" : 1 }
{ "lock.hash" : 1 }
{ "lock.ownerAddress" : 1 }
===== metadata INDEXES =====
{ "_id" : 1 }
{ "metadataEntry.compositeHash" : 1 }
{
        "metadataEntry.sourceAddress" : 1,
        "metadataEntry.metadataType" : 1,
        "metadataEntry.scopedMetadataKey" : 1
}
{
        "metadataEntry.targetAddress" : 1,
        "metadataEntry.metadataType" : 1,
        "metadataEntry.scopedMetadataKey" : 1
}
===== mosaicResolutionStatements INDEXES =====
{ "_id" : 1 }
{ "statement.height" : 1, "statement.unresolved" : 1 }
===== mosaicRestrictions INDEXES =====
{ "_id" : 1 }
{ "mosaicRestrictionEntry.compositeHash" : 1 }
===== mosaics INDEXES =====
{ "_id" : 1 }
{ "mosaic.id" : 1 }
{ "mosaic.ownerAddress" : 1 }
===== multisigs INDEXES =====
{ "_id" : 1 }
{ "multisig.accountAddress" : 1 }
===== namespaces INDEXES =====
{ "_id" : 1 }
{ "namespace.level0" : 1, "meta.index" : 1, "namespace.depth" : 1 }
{
        "meta.latest" : -1,
        "meta.index" : 1,
        "namespace.level0" : 1,
        "namespace.depth" : 1
}
```

(continues on next page)

(continued from previous page)

```
{ "meta.latest" : -1, "namespace.level1" : 1, "namespace.depth" : 1 }
{ "meta.latest" : -1, "namespace.level2" : 1, "namespace.depth" : 1 }
{ "meta.latest" : -1, "namespace.ownerAddress" : 1 }
===== partialTransactions INDEXES =====
{ "_id" : 1 }
{ "transaction.signerPublicKey" : 1, "_id" : -1 }
{ "transaction.recipientAddress" : 1, "_id" : -1 }
{ "meta.hash" : 1 }
{ "meta.addresses" : 1 }
{ "meta.aggregateId" : 1 }
{ "meta.aggregateHash" : 1 }
===== secretLocks INDEXES =====
{ "_id" : 1 }
{ "lock.compositeHash" : 1 }
{ "lock.ownerAddress" : 1 }
===== system.profile INDEXES =====
===== transactionStatements INDEXES =====
{ "_id" : 1 }
{
        "statement.height" : 1,
        "statement.source.primaryId" : 1,
        "statement.source.secondaryId" : 1
}
===== transactionStatuses INDEXES =====
{ "_id" : 1 }
{ "status.hash" : 1 }
{ "status.deadline" : -1 }
===== transactions INDEXES =====
{ "_id" : 1 }
{ "transaction.signerPublicKey" : 1, "_id" : -1 }
{ "transaction.recipientAddress" : 1, "_id" : -1 }
{ "meta.hash" : 1 }
{ "meta.addresses" : 1 }
{ "meta.aggregateId" : 1 }
{ "meta.height" : -1 }
{ "transaction.deadline" : -1 }
{ "transaction.cosignatures.signerPublicKey" : 1 }
{ "transaction.id" : 1, "transaction.type" : 1 }
===== unconfirmedTransactions INDEXES =====
{ "_id" : 1 }
{ "transaction.signerPublicKey" : 1, "_id" : -1 }
{ "transaction.recipientAddress" : 1, "_id" : -1 }
{ "meta.hash" : 1 }
{ "meta.addresses" : 1 }
{ "meta.aggregateId" : 1 }
{ "meta.aggregateHash" : 1 }
bye
[.] (exit code: 0)
/
[+] db prepared, checking account indexes
MongoDB shell version v4.2.5
connecting to: mongodb://localhost:27017/catapult?compressors=disabled&
↪gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("a9d9d654-d0c9-4760-bf52-94138f4c6871") }
MongoDB server version: 4.2.5
[
        {
```

(continues on next page)

(continued from previous page)

```
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "catapult.accounts"
        },
        {
                "v" : 2,
                "key" : {
                        "account.publicKey" : 1
                },
                "name" : "account.publicKey_1",
                "ns" : "catapult.accounts"
        },
        {
                "v" : 2,
                "unique" : true,
                "key" : {
                        "account.address" : 1
                },
                "name" : "account.address_1",
                "ns" : "catapult.accounts"
        }
]
2021-07-14T02:52:54.345+0000 I  CONTROL  [main] Automatically disabling TLS 1.0, to
force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2021-07-14T02:52:54.349+0000 W  ASIO     [main] No TransportLayer configured during
NetworkInterface startup
killing process with pid: 10
2021/07/14 02:52 Start: mijin docker-compose
Creating network "catapult_default" with the default driver
Creating catapult_db_1 ... done
Creating catapult_api-node-broker_1 ... done
Creating catapult_rest-gateway_1    ... done
Creating catapult_api-node_1        ... done
2021/07/14 02:52 End: ALL Success
```

3.3.1.8 Step.7

動作を確認します。

コンテナの動作確認

State がすべて Up であることを確認してください。

```
$ docker-compose ps
          Name                        Command               State           Ports
-------------------------------------------------------------------------------------
 ↪-----
catapult_api-node-broker_1   bash -c /bin/bash /scripts ...   Up
catapult_api-node_1          bash -c perl /scripts/wait ...   Up      0.0.0.0:7900->
 ↪7900/tcp
```

(continues on next page)

(continued from previous page)

```
catapult_db_1                 docker-entrypoint.sh bash  ...   Up      27017/tcp
catapult_rest-gateway_1       docker-entrypoint.sh ash - ...   Up      0.0.0.0:3000->
↪3000/tcp
```

ブロックがすすんでいるかを確認

ブロックが積み上がっているか確認します。
他のノードでも実行し、同じブロックで進んでいるかを確認します。

API ノードの場合

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ docker-compose logs --
↪tail=20 api-node| grep heights
api-node_1          | 2021-07-14 02:59:36.958509 0x00007f95631c8700: <info>
(chain::ChainSynchronizer.cpp@217) peer returned 1 blocks (heights 400 - 400)
api-node_1          | 2021-07-14 02:59:36.958645 0x00007f95631c8700: <debug>
(disruptor::Disruptor.cpp@43) disruptor queuing element 27 (1 blocks (heights 400 -
400) [00000000] from Remote_Pull with size 424B)
api-node_1          | 2021-07-14 02:59:37.046493 0x00007f953bfff700: <info>
(disruptor::ConsumerDispatcher.cpp@44) completing processing of element 27 (1 blocks
(heights 400 - 400) [F15ACAA0] from Remote_Pull with size 424B), last consumer is 0
elements behind
```

PEER ノードの場合

```
$ docker-compose logs --tail=20 peer-node| grep heights
peer-node_1  | 2021-07-14 03:00:26.168343 0x00007fb1c396a700: <debug>
(disruptor::Disruptor.cpp@43) disruptor queuing element 942 (1 blocks (heights 403 -
403) [00000000] from Remote_Push with size 376B)
peer-node_1  | 2021-07-14 03:00:26.257224 0x00007fb1a57fa700: <info>
(disruptor::ConsumerDispatcher.cpp@44) completing processing of element 942 (1 blocks
(heights 403 - 403) [5C675B6B] empty from Remote_Push with size 376B), last consumer is
0 elements behind
peer-node_1  | 2021-07-14 03:00:26.317777 0x00007fb1c3169700: <debug>
(disruptor::Disruptor.cpp@43) disruptor queuing element 943 (1 blocks (heights 403 -
403) [00000000] from Remote_Push with size 376B)
peer-node_1  | 2021-07-14 03:00:26.368263 0x00007fb1a57fa700: <info>
(disruptor::ConsumerDispatcher.cpp@44) completing processing of element 943 (1 blocks
(heights 403 - 403) [5C675B6B] from Remote_Push with size 376B), last consumer is 0
elements behind
```

API ノードの場合、rest が接続できるか確認

ノード情報が取得できるか確認します。

```
$ curl -Ss http://localhost:3000/node/info | jq -r
{
  "version": 16777216,
  "publicKey": "E4BF3706483B4D42243F3DCB2625021C3E3AE7C253CC466154EEDF9775012C20",
  "networkGenerationHashSeed":
↪"2DE20B93EBE048A3BA132CC9874BCABBC21C87E18FE9836B8D5D002E57640D4B",
```

(continues on next page)

(continued from previous page)

```
  "roles": 70,
  "port": 7900,
  "networkIdentifier": 96,
  "host": "api1.mijin.internal",
  "friendlyName": "api1.mijin.internal",
  "nodePublicKey": "27E7EEAF5819493D60CA848BAA48145A1A97DF63596ED41394563C791303C778"
}
```

3.3.2 mijin Catapult(v.2) のノードログを確認する

mijin Catapult(v.2) ノードのログを確認するまでの手順となります。

3.3.2.1 対象

- ノードのブロックが進まない
- エラーが出て原因がわからない

3.3.2.2 Step.1

ノードにリモートログインします。
AWS MarketPlace の mijin Catapult(v.2) にログインしたい場合は、mijin Catapult(v.2) EC2 インスタンスログイン方法 を参照してください。

3.3.2.3 Step.2

mijin を起動している「catapult」ユーザーにスイッチします。

```
$ sudo su - catapult
catapult@api1:~$
```

3.3.2.4 Step.3

mijin の起動ファイルがあるディレクトリに移動します。
API,PEER ノードでディレクトリが違うことに注意してください。

ノード	ディレクトリ
API/Dual	mijin-catapult-package/package/**api**/catapult/
PEER	mijin-catapult-package/package/**peer**/catapult/

API ノードの場合は、以下のように移動します。

```
catapult@api1:~$ cd mijin-catapult-package/package/api/catapult/
catapult@api1:~/mijin-catapult-package/package/api/catapult$
```

3.3.2.5 Step.4

mijin では docker による複数コンテナを起動しています。複数のコンテナを確認します。

API ノードの場合

４つのコンテナが起動しています。
全ての State が Up であることを確認してください。

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ docker-compose ps
          Name                        Command               State           Ports
-------------------------------------------------------------------------------------
↪-----
catapult_api-node-broker_1   bash -c /bin/bash /scripts ...   Up
catapult_api-node_1          bash -c perl /scripts/wait ...   Up      0.0.0.0:7900->
↪7900/tcp
catapult_db_1                docker-entrypoint.sh bash  ...   Up      27017/tcp
catapult_rest-gateway_1      docker-entrypoint.sh ash - ...   Up      0.0.0.0:3000->
↪3000/tcp
```

PEER ノードの場合

１つのコンテナが起動しています。
全ての State が Up であることを確認してください。

```
catapult@peer1:~/mijin-catapult-package/package/peer/catapult$ docker-compose ps
        Name                      Command               State           Ports
-------------------------------------------------------------------------------------
catapult_peer-node_1   bash -c /bin/bash /scripts ...   Up      0.0.0.0:7900->7900/tcp
```

3.3.2.6 Step.5

ログを確認するのは docker-compose を使った以下のコマンドで確認できます。

```
docker-compose logs
```

docker-compose について詳しく知りたい場合は、以下のサイトをご確認ください。
https://docs.docker.com/compose/

以下では、確認する方法の例を示します。

各コンテナの直近 XX 行ログを確認します

```
docker-compose logs --tail=10
```

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ docker-compose logs --
↪tail=10
Attaching to catapult_rest-gateway_1, catapult_api-node_1, catapult_api-node-broker_1,
catapult_db_1
api-node-broker_1  | 2021-06-14 00:41:32.447154 0x00007f59efb66700: <debug>
(subscribers::BrokerMessageReaders.h@90) preparing to process 1 messagesfrom /data/
↪spool/block_change
api-node-broker_1  | 2021-06-14 00:41:32.447137 0x00007f59ef365700: <debug>
(subscribers::BrokerMessageReaders.h@90) preparing to process 2 messagesfrom /data/
↪spool/state_change
api-node-broker_1  | 2021-06-14 00:41:50.955992 0x00007f59efb66700: <debug>
(subscribers::BrokerMessageReaders.h@90) preparing to process 1 messagesfrom /data/
↪spool/block_change
api-node-broker_1  | 2021-06-14 00:41:50.955975 0x00007f59ef365700: <debug>
(subscribers::BrokerMessageReaders.h@90) preparing to process 2 messagesfrom /data/
↪spool/state_change
api-node-broker_1  | 2021-06-14 00:42:05.963816 0x00007f59efb66700: <debug>
(subscribers::BrokerMessageReaders.h@90) preparing to process 2 messagesfrom /data/
↪spool/state_change
api-node-broker_1  | 2021-06-14 00:42:05.966239 0x00007f59ef365700: <debug>
(subscribers::BrokerMessageReaders.h@90) preparing to process 1 messagesfrom /data/
↪spool/block_change
api-node-broker_1  | 2021-06-14 00:42:23.972546 0x00007f59efb66700: <debug>
(subscribers::BrokerMessageReaders.h@90) preparing to process 2 messagesfrom /data/
↪spool/state_change
api-node-broker_1  | 2021-06-14 00:42:23.972928 0x00007f59ef365700: <debug>
(subscribers::BrokerMessageReaders.h@90) preparing to process 1 messagesfrom /data/
↪spool/block_change
api-node-broker_1  | 2021-06-14 00:42:41.982065 0x00007f59ef365700: <debug>
(subscribers::BrokerMessageReaders.h@90) preparing to process 1 messagesfrom /data/
↪spool/block_change
api-node-broker_1  | 2021-06-14 00:42:41.982048 0x00007f59efb66700: <debug>
(subscribers::BrokerMessageReaders.h@90) preparing to process 2 messagesfrom /data/
↪spool/state_change
api-node_1         | 2021-06-14 00:42:41.523496 0x00007f1bda429700: <debug>
(chain::CompareChains.cpp@119) comparing chain scores: 7676859692801638166 (local) vs
7676974281840495032 (remote)
api-node_1         | 2021-06-14 00:42:41.523569 0x00007f1bda429700: <debug>
(chain::CompareChains.cpp@145) comparing hashes with local height 67006,starting height
66976, max hashes 1440
api-node_1         | 2021-06-14 00:42:41.529042 0x00007f1bdac2a700: <debug>
(chain::ChainSynchronizer.cpp@309) pulling blocks from remote with common height 67006
(fork depth = 0) from DEC1EF1767E76BC31DF2FDADC75C23F6FDA6ECCB22554E4F4C790F81F869F797
@ 10.0.3.199
api-node_1         | 2021-06-14 00:42:41.618591 0x00007f1bdac2a700: <info>
(chain::ChainSynchronizer.cpp@217) peer returned 1 blocks (heights 67007 - 67007)
api-node_1         | 2021-06-14 00:42:41.618764 0x00007f1bdac2a700: <debug>
(chain::ChainSynchronizer.cpp@223) completing chain synchronization with1 blocks (fork
depth = 0)
api-node_1         | 2021-06-14 00:42:41.618805 0x00007f1bdac2a700: <debug>
(disruptor::Disruptor.cpp@43) disruptor queuing element 66554 (1 blocks (heights 67007
- 67007) [00000000] from Remote_Pull with size 376B)
api-node_1         | 2021-06-14 00:42:41.618960 0x00007f1bdac2a700: <info>
```

(continues on next page)

(continued from previous page)

```
(chain::RemoteApiForwarder.h@69) completed 'synchronizer task' (peer2.mijin.internal @
peer2.mijin.internal:7900) with result Success
api-node_1          | 2021-06-14 00:42:41.676588 0x00007f1bb0ff9700: <debug>
(cache::SupplementalDataStorage.cpp@32) wrote last recalculation height 67000 last
finalized height 66976 dynamic fee multiplier 0 total transactions 34 (score = [0,
7676974281840495032], height = 67007)
api-node_1          | 2021-06-14 00:42:41.714555 0x00007f1bb07f8700: <info>
(disruptor::ConsumerDispatcher.cpp@44) completing processing of element 66554 (1 blocks
(heights 67007 - 67007) [833B95DA] from Remote_Pull with size 376B), last consumer is 0
elements behind
api-node_1          | 2021-06-14 00:42:50.247964 0x00007f1bdac2a700: <debug>
(chain::RoundContext.cpp@89) not completable - Erv == g(Vrv) and descendant can reach
g(Crv) (total weight 15000000, cumulative precommit weight 3000000)
db_1                | 2021-06-02T09:37:24.913+0000 I  NETWORK  [listener] connection
accepted from 172.20.0.9:47924 #16 (15 connections now open)
db_1                | 2021-06-02T09:37:24.913+0000 I  NETWORK  [conn16] received client
metadata from 172.20.0.9:47924 conn16: { driver: { name: "nodejs", version: "3.6.0" },
os: { type: "Linux", name: "linux", architecture: "x64", version: "5.4.0-1029-aws" },
platform: "'Node.js v12.18.1, LE (legacy)" }
db_1                | 2021-06-02T09:39:05.968+0000 I  NETWORK  [listener] connection
accepted from 172.20.0.9:47942 #17 (16 connections now open)
db_1                | 2021-06-02T09:39:05.969+0000 I  NETWORK  [conn17] received client
metadata from 172.20.0.9:47942 conn17: { driver: { name: "nodejs", version: "3.6.0" },
os: { type: "Linux", name: "linux", architecture: "x64", version: "5.4.0-1029-aws" },
platform: "'Node.js v12.18.1, LE (legacy)" }
db_1                | 2021-06-02T09:42:43.866+0000 I  NETWORK  [listener] connection
accepted from 172.20.0.9:47948 #18 (17 connections now open)
db_1                | 2021-06-02T09:42:43.869+0000 I  NETWORK  [conn18] received client
metadata from 172.20.0.9:47948 conn18: { driver: { name: "nodejs", version: "3.6.0" },
os: { type: "Linux", name: "linux", architecture: "x64", version: "5.4.0-1029-aws" },
platform: "'Node.js v12.18.1, LE (legacy)" }
db_1                | 2021-06-08T05:37:00.594+0000 I  NETWORK  [listener] connection
accepted from 172.20.0.9:58874 #19 (18 connections now open)
db_1                | 2021-06-08T05:37:00.596+0000 I  NETWORK  [conn19] received client
metadata from 172.20.0.9:58874 conn19: { driver: { name: "nodejs", version: "3.6.0" },
os: { type: "Linux", name: "linux", architecture: "x64", version: "5.4.0-1029-aws" },
platform: "'Node.js v12.18.1, LE (legacy)" }
db_1                | 2021-06-12T02:52:21.305+0000 I  NETWORK  [listener] connection
accepted from 172.20.0.9:37814 #20 (19 connections now open)
db_1                | 2021-06-12T02:52:21.310+0000 I  NETWORK  [conn20] received client
metadata from 172.20.0.9:37814 conn20: { driver: { name: "nodejs", version: "3.6.0" },
os: { type: "Linux", name: "linux", architecture: "x64", version: "5.4.0-1029-aws" },
platform: "'Node.js v12.18.1, LE (legacy)" }
rest-gateway_1      | > node _build/index.js "/userconfig/rest.json"
rest-gateway_1      |
rest-gateway_1      | [winston] Attempt to write logs with no transports {"message":
↪"loading config from /userconfig/rest.json","level":"info"}
rest-gateway_1      | info: loading config from /userconfig/rest.json
rest-gateway_1      | verbose: finished loading rest server config {"network":{"name":
↪"mijin","description":"mijin network"},"port":3000,"crossDomain":{"allowedHosts":["*
↪"],"allowedMethods":["GET","POST","PUT","OPTIONS"]},"extensions":["accountLink",
↪"aggregate","lockHash","lockSecret","mosaic","metadata","multisig","namespace",
↪"receipts","restrictions","transfer"],"db":{"url":"mongodb://db:27017/","name":
↪"catapult","pageSizeMin":10,"pageSizeMax":100,"maxConnectionAttempts":7,
↪"baseRetryDelay":750},"apiNode":{"host":"api-node","port":7900,
↪"tlsClientCertificatePath":"/userconfig/resources/cert/node.crt.pem",
↪"tlsClientKeyPath":"/userconfig/resources/cert/node.key.pem","tlsCaCertificatePath":
```

(continues on next page)

(continued from previous page)

```
↪"/userconfig/resources/cert/ca.cert.pem","timeout":1000,"networkPropertyFilePath":"/
↪api-node-config/config-network.properties","nodePropertyFilePath":"/api-node-config/
↪config-node.properties"},"websocket":{"mq":{"host":"api-node-broker","port":7902,
↪"monitorInterval":500,"connectTimeout":10000,"monitorLoggingThrottle":60000},
↪"allowOptionalAddress":true},"throttling":{"burst":100,"rate":30},"logging":{"console
↪":{"formats":["colorize","simple"],"level":"verbose","handleExceptions":true},"file":
↪{"formats":["prettyPrint"],"level":"verbose","handleExceptions":true,"filename":
↪"catapult-rest.log","maxsize":20971520,"maxFiles":100}},"numBlocksTransactionFeeStats
↪":300,"timestamp":"2021-06-02T09:27:19.336Z"}
rest-gateway_1     | info: connecting to mongodb://db:27017/ (database:catapult) {
↪"timestamp":"2021-06-02T09:27:19.407Z"}
rest-gateway_1     | (node:24) DeprecationWarning: current Server Discovery and
Monitoring engine is deprecated, and will be removed in a future version. To use the new
Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the
MongoClient constructor.
rest-gateway_1     | verbose: connected to mongo at mongodb://db:27017/catapult {
↪"timestamp":"2021-06-02T09:27:19.443Z"}
rest-gateway_1     | info: registering routes {"timestamp":"2021-06-02T09:27:19.445Z"}
rest-gateway_1     | info: listening on port 3000 {"timestamp":"2021-06-02T09:27:19.462Z
↪"}
```

指定したコンテナごとに確認します

コンテナのホスト名を指定することで、特定のコンテナに絞ってログを出力します。
ホスト名は固定されています。

- api-node
- peer-node
- api-node-broker
- db
- rest-gateway

```
docker-compose logs [コンテナ名 Container name]
```

```
catapult@api1:~/mijin-catapult-package/package/api/catapult$ docker-compose logs rest-
↪gateway
Attaching to catapult_rest-gateway_1
rest-gateway_1     |
rest-gateway_1     | > catapult-api-rest@0.0.0 start /app/catapult-rest/rest
rest-gateway_1     | > node _build/index.js "/userconfig/rest.json"
rest-gateway_1     |
rest-gateway_1     | [winston] Attempt to write logs with no transports {"message":
↪"loading config from /userconfig/rest.json","level":"info"}
rest-gateway_1     | info: loading config from /userconfig/rest.json
rest-gateway_1     | verbose: finished loading rest server config {"network":{"name":
↪"mijin","description":"mijin network"},"port":3000,"crossDomain":{"allowedHosts":["*
↪"],"allowedMethods":["GET","POST","PUT","OPTIONS"]},"extensions":["accountLink",
↪"aggregate","lockHash","lockSecret","mosaic","metadata","multisig","namespace",
↪"receipts","restrictions","transfer"],"db":{"url":"mongodb://db:27017/","name":
↪"catapult","pageSizeMin":10,"pageSizeMax":100,"maxConnectionAttempts":7,
↪"baseRetryDelay":750},"apiNode":{"host":"api-node","port":7900,
```

(continues on next page)

(continued from previous page)

```
↪"tlsClientCertificatePath":"/userconfig/resources/cert/node.crt.pem",
↪"tlsClientKeyPath":"/userconfig/resources/cert/node.key.pem","tlsCaCertificatePath":
↪"/userconfig/resources/cert/ca.cert.pem","timeout":1000,"networkPropertyFilePath":"/
↪api-node-config/config-network.properties","nodePropertyFilePath":"/api-node-config/
↪config-node.properties"},"websocket":{"mq":{"host":"api-node-broker","port":7902,
↪"monitorInterval":500,"connectTimeout":10000,"monitorLoggingThrottle":60000},
↪"allowOptionalAddress":true},"throttling":{"burst":100,"rate":30},"logging":{"console
↪":{"formats":["colorize","simple"],"level":"verbose","handleExceptions":true},"file":
↪{"formats":["prettyPrint"],"level":"verbose","handleExceptions":true,"filename":
↪"catapult-rest.log","maxsize":20971520,"maxFiles":100}},"numBlocksTransactionFeeStats
↪":300,"timestamp":"2021-06-02T09:27:19.336Z"}
rest-gateway_1     | info: connecting to mongodb://db:27017/ (database:catapult) {
↪"timestamp":"2021-06-02T09:27:19.407Z"}
rest-gateway_1     | (node:24) DeprecationWarning: current Server Discovery and
Monitoring engine is deprecated, and will be removed in a future version. To use the new
Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the
MongoClient constructor.
rest-gateway_1     | verbose: connected to mongo at mongodb://db:27017/catapult {
↪"timestamp":"2021-06-02T09:27:19.443Z"}
rest-gateway_1     | info: registering routes {"timestamp":"2021-06-02T09:27:19.445Z"}
rest-gateway_1     | info: listening on port 3000 {"timestamp":"2021-06-02T09:27:19.462Z
↪"}
```

3.3.3 ノード間の暗号化通信の更新

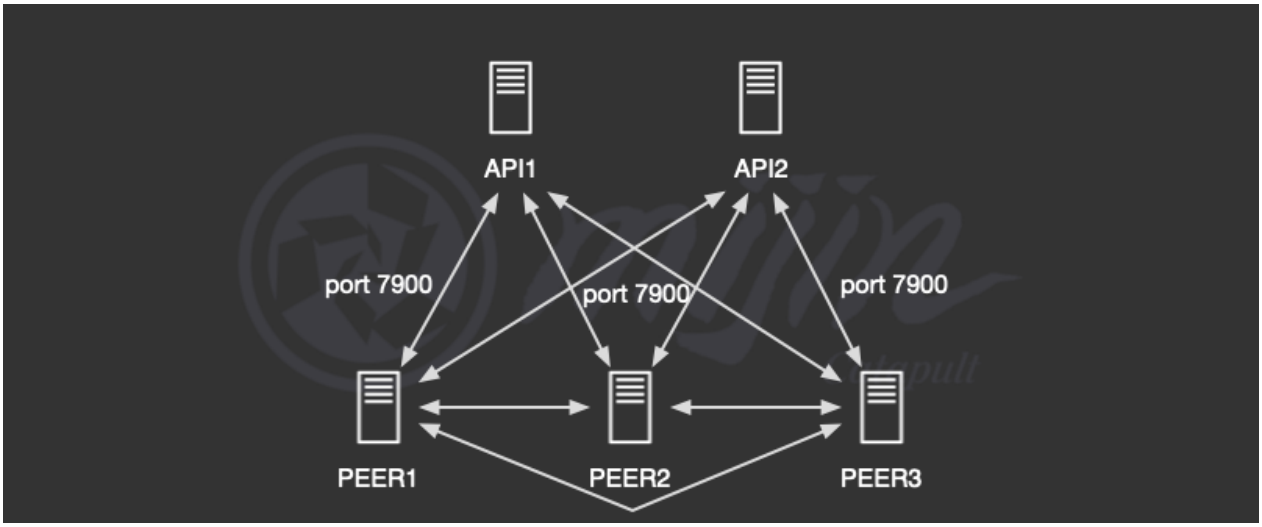
本章では、ノード間通信で使用する証明書を更新したい場合の更新方法を説明します。

注釈:

AWS MarketPlace 経由にて、mijin Catapult(v.2) を deploy した場合、AWS Systems Manager Parameter Store に初期データがバックアップされています。
以下のパラメータ値が更新対象となり、ノードと AWS Systems Manager Parameter Store で差分が発生することに留意してください。
差分があることで、動作に影響することはありません。
・/デプロイ時に指定した冠名/shares/new-cert/各ノード/CA/[*].pem
・/デプロイ時に指定した冠名/shares/nemesis_addresses_harvesting.json
・/デプロイ時に指定した冠名/shares/nemesis_addresses_harvesting_voting.json
・/デプロイ時に指定した冠名/shares/nemesis_addresses_harvesting_vrf.json

3.3.3.1 mijin Catapult(v.2) のノード間の暗号化通信について

mijin Catapult(v.2) のノード間通信は、**TCP ポート/7900** にて TLS1.3 による SSL 暗号化通信を行い、認識しているノードのみ正しいノードとして通信を行います。
SSL 通信には、自己署名の証明書が各ノードに適用されており、ノードは全てのノードの自己署名の証明書の KeyPair の公開鍵を事前に登録しています。
また、この自己署名の証明書から作成された KeyPair は、ブロックチェーンを生成できる権限を持つアカウントとして使用されます。



3.3.3.2 ノードの SSL 証明書の更新方法

ノード間で使用する SSL 証明書の更新の手順は以下です。

1. CA 及び署名ノード SSL 証明書作成 (ノード間の通信で使用)
2. 1 の SSL 証明書の KeyPair から秘密鍵を取り出し、mijin Catapult(v.2) ブロックチェーン上にノードのブロック生成を有効にするトランザクションを発行
3. 2 の秘密鍵に紐づいた秘密鍵とファイナライズ用 dat ファイルを作成し、2 の秘密鍵と紐づけるトランザクションを発行
4. 該当ノードにて、SSL 証明書と dat ファイルの置き換える。
5. 全ノードのコンフィグにある対象ノードの公開鍵を 2 の鍵に置き換える。

警告:

2022/10 現在、SSL 証明書の更新手順は複雑であり、環境ごとに異なるため、mijin Support にお問合せください。
将来、ツールによる容易な更新方法を提供する予定です。

3.4 mijin Catapult(v.2) データディレクトリ構造

mijin のデータ構造について説明します。

3.4.1 データ配置のディレクトリ

ディレクトリ	説明
/home/catapult/mijin-catapult-package	mijin の config ファイルなどパッケージに関するディレクトリ |
/mnt/mijin/blocks	mijin のブロックデータディレクトリ
/mnt/mijin/mongo	mijin の mongo データディレクトリ

3.4.2 mijin パッケージの構造

3.4.2.1 API ノード

```
/home/catapult/mijin-catapult-package
├── default # mijin 構築時に使用するディレクトリ (データとしては不要) Directory to be used when
building mijin (not required as data)
|   ├── catapult
|   |   ├── bin-mount
|   |   |   ├── await
|   |   |   ├── wait
|   |   |   └── waitmongo
|   |   ├── mongo
|   |   |   ├── mongoDbDrop.js
|   |   |   ├── mongoDbPrepare.js
|   |   |   ├── mongoDeploy.sh
|   |   |   ├── mongoLockHashDbPrepare.js
|   |   |   ├── mongoLockSecretDbPrepare.js
|   |   |   ├── mongoMetadataDbPrepare.js
|   |   |   ├── mongoMosaicDbPrepare.js
|   |   |   ├── mongoMultisigDbPrepare.js
|   |   |   ├── mongoNamespaceDbPrepare.js
|   |   |   ├── mongoRestrictionAccountDbPrepare.js
|   |   |   ├── mongoRestrictionMosaicDbPrepare.js
|   |   |   └── mongors.sh
|   |   ├── nemgen
|   |   |   └── nemgen.sh
|   |   ├── scripts
|   |   |   ├── prepare.sh
|   |   |   ├── runServerRecover.sh
|   |   |   ├── startApiServer.sh
|   |   |   ├── startBroker.sh
|   |   |   └── startServer.sh
|   |   └── tools
|   |       ├── clean-all.sh
|   |       └── clean-data.sh
|   └── usr
|       └── catapult
|           └── bin-mount
├── package # mijin 構築パッケージ Package for building mijin
|   └── api # api ノード用のパッケージ Package for api node
|       └── catapult
|           ├── docker-compose.yml # docker コンテナを起動する compose ファイル compose file
to start docker container
|           ├── scripts
|           |   ├── prepare.sh
|           |   ├── recover.sh # ブロックリセット時に使用するスクリプト Script to be used when
```

(continues on next page)

(continued from previous page)

```
resetting the block
|            |    ├── runServerRecover.sh
|            |    ├── startApiServer.sh
|            |    ├── startBroker.sh
|            |    ├── startServer.sh
|            |    └── waitmongo.pl
|            └── userconfig
|                ├── block-properties-file.properties # ネメシス（ジェネシス）ブロックを作成する際に使用するプロパティファイル Property file used to create the Nemesis (Genesis) block
|                ├── resources
|                |    ├── cert # ノードの証明書 Node Certificates
|                |    |    ├── ca.cert.pem
|                |    |    ├── ca.pubkey.pem
|                |    |    ├── node.crt.pem
|                |    |    ├── node.full.crt.pem
|                |    |    └── node.key.pem
|                |    ├── config-database.properties # データベースに関する設定 Database Settings
|                |    ├── config-extensions-broker.properties # broker ノードに関する設定 Broker node settings
|                |    ├── config-extensions-recovery.properties # ブロックリカバリに関する設定 Settings for block recovery
|                |    ├── config-extensions-server.properties # ノードで起動する拡張領域関する設定 Settings related to the extension area to be activated in the node
|                |    ├── config-finalization.properties # ファイナライズに関する設定 Finalization Settings
|                |    ├── config-inflation.properties # インフレーションに関する設定 Inflation-related settings
|                |    ├── config-logging-broker.properties # broker のログに関する設定 Settings for broker logging
|                |    ├── config-logging-recovery.properties # リカバリのログに関する設定 Recovery log settings
|                |    ├── config-logging-server.properties # サーバログに関する設定 Server log settings
|                |    ├── config-messaging.properties # MQ に関する設定 MQ-related settings
|                |    ├── config-network.properties # mijin ネットワークに関する設定 mijin network settings
|                |    ├── config-networkheight.properties # ネットワークのブロック高を知るためのノード数の設定 Set the number of nodes to know the block height of the network
|                |    ├── config-node.properties # ノードに関する設定 Node-related settings
|                |    ├── config-pt.properties # Partial トランザクションに関する設定 Partial Transaction Settings
|                |    ├── config-task.properties # タスクに関する設定 Task-related settings
|                |    ├── config-timesync.properties # 時間に関する設定 Time-related settings
|                |    ├── config-user.properties # ストレージに関する設定 Storage Settings
|                |    ├── peers-api.json # API ノードに関する設定 API Node Settings
|                |    ├── peers-p2p.json # PEER ノードに関する設定 PEER Node Settings
|                |    └── votingkey # Voting に使用するファイル Files used for Voting
|                |        └── private_key_tree1.dat
|                ├── rest.json # rest に関する設定 Settings for REST
|                ├── transactions # Nemesis ブロック作成時にアナウンスするトランザクション Transaction to be announced at the time of Nemesis block creation
|                |    ├── voting_tx0.bin
|                |    ├── voting_tx1.bin
|                |    ├── voting_tx2.bin
|                |    ├── voting_tx3.bin
|                |    ├── voting_tx4.bin
```

(continues on next page)

(continued from previous page)

```
|                   |   ├── voting_tx5.bin
|                   |   ├── voting_tx6.bin
|                   |   ├── voting_tx7.bin
|                   |   ├── voting_tx8.bin
|                   |   ├── voting_tx9.bin
|                   |   ├── vrf_tx0.bin
|                   |   ├── vrf_tx1.bin
|                   |   ├── vrf_tx2.bin
|                   |   ├── vrf_tx3.bin
|                   |   ├── vrf_tx4.bin
|                   |   ├── vrf_tx5.bin
|                   |   ├── vrf_tx6.bin
|                   |   ├── vrf_tx7.bin
|                   |   ├── vrf_tx8.bin
|                   |   └── vrf_tx9.bin
|                   └── votingkeys # 全ノードの Voting ファイル Voting files for all nodes
|                       ├── api
|                       |   ├── 0
|                       |   |   └── private_key_tree1.dat
|                       |   ├── 1
|                       |   |   └── private_key_tree1.dat
|                       |   └── 2
|                       |       └── private_key_tree1.dat
|                       └── peer
|                           ├── 0
|                           |   └── private_key_tree1.dat
|                           ├── 1
|                           |   └── private_key_tree1.dat
|                           ├── 2
|                           |   └── private_key_tree1.dat
|                           ├── 3
|                           |   └── private_key_tree1.dat
|                           ├── 4
|                           |   └── private_key_tree1.dat
|                           ├── 5
|                           |   └── private_key_tree1.dat
|                           └── 6
|                               └── private_key_tree1.dat
└── tools
    └── address.py # アドレス作成時に変換するツール Tools to convert when creating addresses
```

3.4.2.2 PEER ノード

```
/home/catapult/mijin-catapult-package
├── default # mijin 構築時に使用するディレクトリ (データとしては不要) Directory to be used when
building mijin (not required as data)
|   └── catapult
|       ├── bin-mount
|       |   ├── await
|       |   ├── wait
|       |   └── waitmongo
|       ├── mongo
|       |   ├── mongoDbDrop.js
|       |   ├── mongoDbPrepare.js
|       |   ├── mongoDeploy.sh
```

(continues on next page)

(continued from previous page)

```
|       |   ├── mongoLockHashDbPrepare.js
|       |   ├── mongoLockSecretDbPrepare.js
|       |   ├── mongoMetadataDbPrepare.js
|       |   ├── mongoMosaicDbPrepare.js
|       |   ├── mongoMultisigDbPrepare.js
|       |   ├── mongoNamespaceDbPrepare.js
|       |   ├── mongoRestrictionAccountDbPrepare.js
|       |   ├── mongoRestrictionMosaicDbPrepare.js
|       |   └── mongors.sh
|       ├── nemgen
|       |   └── nemgen.sh
|       ├── scripts
|       |   ├── prepare.sh
|       |   ├── runServerRecover.sh
|       |   ├── startApiServer.sh
|       |   ├── startBroker.sh
|       |   └── startServer.sh
|       └── tools
|           ├── clean-all.sh
|           └── clean-data.sh
├── package
|   └── peer
|       └── catapult
|           ├── docker-compose.yml # docker コンテナを起動する compose ファイル compose file
to start docker container
|           ├── scripts
|           |   ├── prepare.sh
|           |   ├── recover.sh # ブロックリセット時に使用するスクリプト Script to be used when
resetting the block
|           |   ├── runServerRecover.sh
|           |   ├── startApiServer.sh
|           |   ├── startBroker.sh
|           |   ├── startServer.sh
|           |   └── waitmongo.pl
|           └── userconfig
|               ├── block-properties-file.properties # ネメシス（ジェネシス）ブロックを作成す
る際に使用するプロパティファイル Property file used to create the Nemesis (Genesis) block
|               ├── resources
|               |   ├── cert # ノードの証明書 Node Certificates
|               |   |   ├── ca.cert.pem
|               |   |   ├── ca.pubkey.pem
|               |   |   ├── node.crt.pem
|               |   |   ├── node.full.crt.pem
|               |   |   └── node.key.pem
|               |   ├── config-database.properties # データベースに関する設定 Database
Settings
|               |   ├── config-extensions-recovery.properties # ブロックリカバリに関する設定
Settings for block recovery
|               |   ├── config-extensions-server.properties # ノードで起動する拡張領域関する
設定 Settings related to the extension area to be activated in the node
|               |   ├── config-finalization.properties # ファイナライズに関する設定
Finalization Settings
|               |   ├── config-harvesting.properties # ハーベストに関する設定 Harvest-
↪related settings
|               |   ├── config-inflation.properties # インフレーションに関する設定 Inflation-
↪related settings
|               |   ├── config-logging-recovery.properties # リカバリのログに関する設定
```

(continues on next page)

(continued from previous page)

```
Recovery log settings
|               |   ├── config-logging-server.properties # サーバログに関する設定 Server log
settings
|               |   ├── config-messaging.properties # MQ に関する設定 MQ-related settings
|               |   ├── config-network.properties # mijin ネットワークに関する設定 mijin
network settings
|               |   ├── config-networkheight.properties # ネットワークのブロック高を知るための
ノード数の設定 Set the number of nodes to know the block height of the network
|               |   ├── config-node.properties # ノードに関する設定 Node-related settings
|               |   ├── config-pt.properties # Partial トランザクションに関する設定 Partial
Transaction Settings
|               |   ├── config-task.properties # タスクに関する設定 Task-related settings
|               |   ├── config-timesync.properties # 時間に関する設定 Time-related settings
|               |   ├── config-user.properties # ストレージに関する設定 Storage Settings
|               |   ├── peers-api.json # API ノードに関する設定 API Node Settings
|               |   ├── peers-p2p.json # PEER ノードに関する設定 PEER Node Settings
|               |   └── votingkey # Voting に使用するファイル Files used for Voting
|               |       └── private_key_tree1.dat
|               ├── transactions # Nemesis ブロック作成時にアナウンスするトランザクション
Transaction to be announced at the time of Nemesis block creation
|               |   ├── voting_tx0.bin
|               |   ├── voting_tx1.bin
|               |   ├── voting_tx2.bin
|               |   ├── voting_tx3.bin
|               |   ├── voting_tx4.bin
|               |   ├── voting_tx5.bin
|               |   ├── voting_tx6.bin
|               |   ├── voting_tx7.bin
|               |   ├── voting_tx8.bin
|               |   ├── voting_tx9.bin
|               |   ├── vrf_tx0.bin
|               |   ├── vrf_tx1.bin
|               |   ├── vrf_tx2.bin
|               |   ├── vrf_tx3.bin
|               |   ├── vrf_tx4.bin
|               |   ├── vrf_tx5.bin
|               |   ├── vrf_tx6.bin
|               |   ├── vrf_tx7.bin
|               |   ├── vrf_tx8.bin
|               |   └── vrf_tx9.bin
|               └── votingkeys # 全ノードの Voting ファイル Voting files for all nodes
|                   ├── api
|                   |   ├── 0
|                   |   |   └── private_key_tree1.dat
|                   |   ├── 1
|                   |   |   └── private_key_tree1.dat
|                   |   └── 2
|                   |       └── private_key_tree1.dat
|                   └── peer
|                       ├── 0
|                       |   └── private_key_tree1.dat
|                       ├── 1
|                       |   └── private_key_tree1.dat
|                       ├── 2
|                       |   └── private_key_tree1.dat
|                       ├── 3
|                       |   └── private_key_tree1.dat
```

(continues on next page)

(continued from previous page)

```
|                         ├── 4
|                         |   └── private_key_tree1.dat
|                         ├── 5
|                         |   └── private_key_tree1.dat
|                         └── 6
|                             └── private_key_tree1.dat
└── tools
    └── address.py # アドレス作成時に変換するツール Tools to convert when creating addresses
```

3.4.3 ブロックデータの構造

```
/mnt/mijin/blocks/
├── data # ブロックチェーンのデータディレクトリ Blockchain Data Directory
|   ├── 00000
|   ├── audit
|   ├── catapult_server0000.log
|   ├── commit_step.dat
|   ├── importance
|   ├── index.dat
|   ├── logs
|   ├── proof.index.dat
|   ├── server.lock
|   ├── spool
|   ├── startup
|   ├── state
|   ├── statedb
|   ├── summary.txt
|   ├── transfer_message
|   ├── voting
|   └── voting_status.dat
└── seed # Nemesis ブロック Nemesis block
    ├── 00000
    ├── index.dat
    ├── proof.index.dat
    └── summary.txt
```

3.4.4 Mongo データ構造

```
/mnt/mijin/mongo/
└── db # mongo のデータディレクトリ mongo data directory
    ├── WiredTiger
    ├── WiredTiger.lock
    ├── WiredTiger.turtle
    ├── WiredTiger.wt
    ├── WiredTigerLAS.wt
    ├── _mdb_catalog.wt
    ├── collection-0--1310205274663118138.wt
    ├── collection-0--3714664905013916938.wt
    ├── collection-108--1310205274663118138.wt
    ├── collection-115--1310205274663118138.wt
    ├── collection-122--1310205274663118138.wt
    ├── collection-129--1310205274663118138.wt
```

(continues on next page)

(continued from previous page)

```
    ├── collection-138--1310205274663118138.wt
    ├── collection-145--1310205274663118138.wt
    ├── collection-150--1310205274663118138.wt
    ├── collection-162--1310205274663118138.wt
    ├── collection-167--1310205274663118138.wt
    ├── collection-2--1310205274663118138.wt
    ├── collection-28--1310205274663118138.wt
    ├── collection-35--1310205274663118138.wt
    ├── collection-4--1310205274663118138.wt
    ├── collection-56--1310205274663118138.wt
    ├── collection-61--1310205274663118138.wt
    ├── collection-66--1310205274663118138.wt
    ├── collection-71--1310205274663118138.wt
    ├── collection-78--1310205274663118138.wt
    ├── collection-8--1310205274663118138.wt
    ├── collection-9--1310205274663118138.wt
    ├── collection-93--1310205274663118138.wt
    ├── diagnostic.data
    ├── index-1--1310205274663118138.wt
    ├── index-1--3714664905013916938.wt
    ├── index-10--1310205274663118138.wt
    ├── index-102--1310205274663118138.wt
    ├── index-104--1310205274663118138.wt
    ├── index-106--1310205274663118138.wt
    ├── index-109--1310205274663118138.wt
    ├── index-11--1310205274663118138.wt
    ├── index-110--1310205274663118138.wt
    ├── index-113--1310205274663118138.wt
    ├── index-116--1310205274663118138.wt
    ├── index-117--1310205274663118138.wt
    ├── index-120--1310205274663118138.wt
    ├── index-123--1310205274663118138.wt
    ├── index-124--1310205274663118138.wt
    ├── index-127--1310205274663118138.wt
    ├── index-13--1310205274663118138.wt
    ├── index-130--1310205274663118138.wt
    ├── index-131--1310205274663118138.wt
    ├── index-134--1310205274663118138.wt
    ├── index-136--1310205274663118138.wt
    ├── index-139--1310205274663118138.wt
    ├── index-140--1310205274663118138.wt
    ├── index-143--1310205274663118138.wt
    ├── index-146--1310205274663118138.wt
    ├── index-147--1310205274663118138.wt
    ├── index-151--1310205274663118138.wt
    ├── index-152--1310205274663118138.wt
    ├── index-154--1310205274663118138.wt
    ├── index-156--1310205274663118138.wt
    ├── index-158--1310205274663118138.wt
    ├── index-16--1310205274663118138.wt
    ├── index-160--1310205274663118138.wt
    ├── index-163--1310205274663118138.wt
    ├── index-164--1310205274663118138.wt
    ├── index-168--1310205274663118138.wt
    ├── index-169--1310205274663118138.wt
    ├── index-19--1310205274663118138.wt
    ├── index-22--1310205274663118138.wt
```

(continues on next page)

(continued from previous page)

```
    ├── index-25--1310205274663118138.wt
    ├── index-29--1310205274663118138.wt
    ├── index-3--1310205274663118138.wt
    ├── index-30--1310205274663118138.wt
    ├── index-32--1310205274663118138.wt
    ├── index-36--1310205274663118138.wt
    ├── index-37--1310205274663118138.wt
    ├── index-39--1310205274663118138.wt
    ├── index-41--1310205274663118138.wt
    ├── index-44--1310205274663118138.wt
    ├── index-46--1310205274663118138.wt
    ├── index-48--1310205274663118138.wt
    ├── index-5--1310205274663118138.wt
    ├── index-50--1310205274663118138.wt
    ├── index-52--1310205274663118138.wt
    ├── index-54--1310205274663118138.wt
    ├── index-57--1310205274663118138.wt
    ├── index-58--1310205274663118138.wt
    ├── index-6--1310205274663118138.wt
    ├── index-62--1310205274663118138.wt
    ├── index-63--1310205274663118138.wt
    ├── index-67--1310205274663118138.wt
    ├── index-68--1310205274663118138.wt
    ├── index-72--1310205274663118138.wt
    ├── index-73--1310205274663118138.wt
    ├── index-75--1310205274663118138.wt
    ├── index-79--1310205274663118138.wt
    ├── index-80--1310205274663118138.wt
    ├── index-82--1310205274663118138.wt
    ├── index-84--1310205274663118138.wt
    ├── index-87--1310205274663118138.wt
    ├── index-89--1310205274663118138.wt
    ├── index-91--1310205274663118138.wt
    ├── index-94--1310205274663118138.wt
    ├── index-95--1310205274663118138.wt
    ├── index-97--1310205274663118138.wt
    ├── index-99--1310205274663118138.wt
    ├── journal
    ├── mongod.lock
    ├── sizeStorer.wt
    └── storage.bson
```

3.5 mijin Catapult(v.2) 環境構築オプション表

通常 mijin 環境の構築は Ansible を使用しており、Playbook の ansible オプションをここで示します。

No	項目名	Default 値	入力値	説明
1	service	peer	peer,api	ansible で構築するモードを指定します。 [peer] peer モードで構築します [api] api モードで構築します

次のページに続く

表 1 – 前のページからの続き

No	項目名	Default 値	入力値	説明
2	share_mode	dir	dir,ssm,s3	Nemesis ブロック作成時に使用した初期アドレスを配置します。 [dir] share_directory のみに保存されます。 [ssm] AWS SSM パラメータストアに保存します。 [s3] AWS S3 に保存します。
3	aws_region	ap-northeast-1	String	AWS のリージョンを指定します。リージョンを指定することで、S3 及び SSM への取得に使われます。 <https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/using-regions-availability-zones.html#concepts-available-regions>
4	ssm_ps_name			share_mode: ssm の場合、SSM パラメータストアに保存するパスを指定します。
5	s3_bucket_name			share_mode: s3 の場合、S3 バケット名を指定します。
6	api_dual_mode	FALSE	true,false	service: api の場合、dual モードで起動するかを指定します。 [Yes] dual モードで起動します。(Harvest の有効化) [No] 通常モードで起動します。
7	api_hosts	[192.168.96.131]	Array[String, String]	API ノードのホスト名又は IP アドレスを配列で指定します。
8	peer_hosts	[192.168.96.132, 192.168.96.133]	Array[String, String]	PEER ノードのホスト名又は IP アドレスを配列で指定します。
9	unix_user	catapult	String	mijin を立ち上げるユーザーを指定します。
10	repo_name	mijin-catapult-package	String	mijin のアプリケーションを配置するディレクトリを指定します。 /home/[unix_user]/[repo_name]
11	catapult_version	v10038	v10038,v10037	catapult のバージョンを指定します。 v10038
12	network_identifier	mijin	mijin,mijin-test	mijin で構築するネットワークタイプを指定します。
13	mongo_docker_version	4.2.5	String	API ノードが使用する mongo のコンテナイメージのバージョンを指定します。 <https://hub.docker.com/_/mongo?tab=tags>
14	mongo_host	db	String	API ノードの mongo コンテナ名を指定します。
15	mongo_max_connection	7	Int	API ノードの mongo への接続制限を指定します。
16	mongo_base_retry_delay	750	Int	API ノードの mongo への接続リトライ時間を指定します。
17	python_docker_version	3.9.0	String	python の docker version を指定します。
18	enable_pip_install	FALSE	true,false	pip install を実行するかを指定します。
19	share_directory	/opt/mijin/shares	String	Nemesis ブロック作成時にアドレスデータを保管するディレクトリを指定します。 s3/ssm を指定しても、一時保存場所として使用されます。
20	block_directory	/opt/mijin/blocks	String	各ノードのブロックデータの保存ディレクトリを指定します。
21	mongo_directory	/opt/mijin/mongo	String	API ノードの mongo データの保存ディレクトリを指定します。

次のページに続く

表 1 – 前のページからの続き

No	項目名	Default 値	入力値	説明
22	docker_network_range	172.20.0.0/24	String	docker が使用するネットワークレンジを指定します。
23	rest_ip_address	172.20.0.9	String	docker ネットワーク上での rest-gateway コンテナの IP アドレスです。
24	domain	mijin.internal	String	各ノード間の通信で使用する SSL 証明書の CN を指定します。
25	node_port	7900	Int	各ノード間通信で使用する TCP ポート番号を指定します。
26	enable_cache_database_storage	TRUE	true,false	キャッシュデータ保存を有効化します。 無効化すると処理性能が向上する場合があります。
27	enable_auto_sync_cleanup	TRUE	true,false	一時同期ファイルの自動削除を指定します。
28	base_namespace	cat	String	基軸通貨で使用するルートネームスペースを指定します。
29	base_currency_name	currency	String	基軸通貨で使用するサブネームスペースを指定します。
30	base_harvest_name	harvest	String	Harvest モザイクのネームスペースを指定します。
31	currency_supply	8'998'999'998'000'000	String	基軸通貨の発行量を指定します。クォート付き。
32	harvest_supply	15'000'000	String	Harvest モザイクの発行量を指定します。
33	block_generation_target_time	15s	String	ブロック生成間隔を指定します（例: 15s）
34	rest_gateway_private_key_num	1	Int	rest_gateway のアドレス作成数を指定します。
35	nemesis_generation_hash_num	1	Int	Nemesis 用の GenerationHash アドレス作成数。
36	nemesis_addresses_harvesting_num	4	Int	Harvest 用アドレス作成数（ノード数と同数必要）
37	nemesis_signer_private_key_num	1	Int	署名用アドレス作成数（通常は 1）
38	nemesis_addresses_num	10	Int	空アドレス（未使用）の作成数。
39	transaction_selections_trategy	oldest	String	oldest, maximize-fee, minimize-fee などから選択。
40	max_time_behind_pull_transaction_start	5m	String	Pull トランザクションの最大遅延許容時間。
41	min_fee_multiplier	100	Int	最小手数料乗数。0 で無料。
42	default_dynamic_fee_multiplier	1'000	String	動的手数料のベース係数。
43	root_namespace_rental_fee_per_block	1	Int	ルートネームスペースのブロック単位のレンタル料。
44	child_namespace_rental_fee	100	Int	子ネームスペースのレンタル料。
45	mosaic_rental_fee	500	Int	モザイクのレンタル料。
46	rest_throttling_burst	100	Int	バースト時の REST 接続最大数。
47	rest_throttling_rate	30	Int	通常時の REST 接続処理レート。
48	unconfirmd_cache_max_response_size	20MB	String	未承認トランザクション受信の最大サイズ。
49	unconfirmd_cache_max_size	5MB	String	未承認トランザクションのキャッシュ最大サイズ。
50	cache_max_response_size	20MB	String	（旧バージョン向け）未承認 Tx の応答最大サイズ。
51	cache_max_size	50'000	String	（旧バージョン向け）キャッシュ Tx の最大数。
52	block_disruptor_slot_count	4096	Int	ブロック処理用スロット数。
53	block_element_trace_interval	1	Int	ブロック要素のトレース間隔。
54	block_disruptor_max_memory_size	300MB	String	ブロックディスラプターの最大メモリ。
55	transaction_disruptor_slot_count	8192	Int	トランザクション処理用スロット数。
56	transaction_element_trace_interval	10	Int	トランザクション要素のトレース間隔。
57	max_transaction_per_block	6'000	String	1 ブロック内の最大トランザクション数。
58	min_transaction_failures_count_for_ban	8	Int	トランザクション失敗回数の BAN 閾値。
59	min_transaction_failures_percent_for_ban	10	Int	トランザクション失敗率の BAN 閾値。
60	partial_cache_max_response_size	5MB	String	部分トランザクション応答サイズの最大値。
61	partial_cache_max_size	20MB	String	部分トランザクションキャッシュ全体の最大サイズ。
62	enable_finalization	TRUE	true,false	ファイナライゼーションプラグインを有効化するか。
63	max_rollback_blocks	0	Int	ロールバック可能な最大ブロック数（0 で確定的）。
64	enable_voting	TRUE	true,false	Voting 機能を有効にするか。
65	voting_set_grouping	160	Int	Voting ラウンドのブロック数（importance_grouping の倍数）。
66	votingkey_start_epoch	1	Int	VotingKey の最小有効期間（エポック単位）。

次のページに続く

表 1 – 前のページからの続き

No	項目名	Default 値	入力値	説明
67	votingkey_end_epoch	26280	Int	VotingKey の最大有効期間（例: 約 821 日）。
68	voting_key_dilution	128	Int	投票キー希釈レベル（再利用の間隔）。
69	enable_revote_on_boot	FALSE	true,false	再起動時に自動的に再投票するか。
70	importance_grouping	40	Int	Importance ラウンド数（影響度スコア更新間隔）。
71	max_transaction_lifetime	24h	String	トランザクションの有効期間（例: 24h）。
72	max_block_future_time	500ms	String	未来ブロック受付の最大許容時間。
73	max_transactions_per_aggregate	1'000	String	アグリゲート Tx に含められる最大 Tx 数（Symbol では 100）。
74	max_cosignatures_per_aggregate	25	Int	アグリゲート Tx に署名できる最大署名数。
75	max_bonded_transaction_lifetime	48h	String	アグリゲートボンデッド Tx の有効期間。
76	locked_funds_per_aggregate	10'000'000	String	アグリゲート Tx のロック保証金。
77	max_hash_lock_duration	2d	String	ハッシュロックの有効期間。
78	max_secret_lock_duration	30d	String	シークレットロックの最大有効期間。
79	min_proof_size	1	Int	シークレットプルーフの最小バイト数。
80	max_proof_size	1000	Int	シークレットプルーフの最大バイト数。
81	max_meta_value_size	1024	Int	メタデータ Value の最大サイズ（バイト）。
82	max_cosignatories_per_account	25	Int	アカウントの連署者数の上限。
83	max_cosigned_accounts_per_account	25	Int	1 アカウントが連署できるアカウント数の上限。
84	max_multisig_depth	3	Int	マルチシグの階層の深さの上限。
85	max_mosaics_per_account	1'000	String	1 アカウントが保有可能なモザイク数。
86	max_mosaic_duration	3650d	String	モザイクの最大有効期間（日単位）。
87	max_mosaic_divisibility	6	Int	モザイクの小数点以下の最大桁数。
88	max_name_size	64	Int	ネームスペース名の最大長（文字数）。
89	max_child_namespaces	256	Int	親ネームスペースが持てる子ネームスペース数。
90	max_namespace_depth	3	Int	ネームスペース階層の最大深度。
91	min_namespace_duration	1m	String	ネームスペースの最小有効期間。
92	max_namespace_duration	3650d	String	ネームスペースの最大有効期間。
93	namespace_grace_period_duration	30d	String	ネームスペース期限後の猶予期間。
94	max_account_restriction_values	512	Int	アカウント制限に設定できる最大値数。
95	max_mosaic_restriction_values	20	Int	モザイク制限に設定できる最大値数。
96	max_message_size	1024	Int	トランザクションメッセージの最大バイト数。