

# 令和なのに NAT インスタンスを手作りして使ってみた

IP フォワードするのと iptables の nat テーブルに書いてあげるだけで NAT インスタンスの出来上がりです。


#Amazon VPC

#Amazon EC2

#NAT Gateway

#NAT Instances

#AWS

 千葉 幸宏 (チバユキ)



2022.11.10



1



7



5

## 昔は NAT Gateway なんて便利なものはなかったんじやよ

---

コンバンハ、千葉（幸）です。

先日、むかしを懐かしむ機会がありました。2016 年 1 月の ACM の登場までは IAM に証明書をアップロードしていた、という内容です。

ほぼ時期を同じくして「いまではそれを使うのが当たり前」となっているサービスが登場したな、ということで思い返されたのは 2015 年 12 月に登場した **NAT Gateway** です。

NAT Gateway が登場するまではみんな EC2 で NAT インスタンスを構築して使っていました。NAT インスタンス用の AMI が提供されていたので、それをそのまま使うケースが多かったかと思っています。

久しぶりに NAT インスタンスのドキュメントを確認すると Linux インスタンスを NAT インスタンス化する手順が載っていたので、今回はそれを試してみます。

## 言うほど NAT インスタンスを使うのは「ナシ」なのか

---

「いまでは NAT Gateway を使うのが当たり前」という論調でここまで喋ってきましたが、NAT インスタンスを採用するのはそこまでアンチパターンなのでしょうか。

わたしとしては、積極的にオススメはしないものの「ACM でなく IAM で証明書を管理する」ほどの「ナシ」度ではないと考えています。

NAT Gateway と NAT インスタンスの比較の観点は以下にまとまっています。

- [NAT ゲートウェイと NAT インスタンスの比較 - Amazon Virtual Private Cloud](#)

いろいろ書いてありますが、簡単に言えば「NAT Gatewayの方が料金が高い代わりにいろいろマネージドでやってくれる」という考え方です。

コストを比較してみます。かなり決めうちで以下条件で試算しました。

- 東京リージョン
- NAT インスタンスは以下構成
  - Amazon Linux2 で t3.micro
  - EBS は gp3 で 8 GiB
  - 停止しない運用とする
- データの通信先はインターネットのみで 100 GB

1ヶ月を 720 時間として考えると月額料金（USD）は以下のようになります。<sup>\*1</sup>

	ゲートウェイ	インスタンス
デバイス	44.64	9.79
ストレージ	-	0.77
データ処理	6.20（※）	-
データ転送	11.40	11.40
合計	62.24	21.96

（※）データ処理は 100 GB の前提で計算していますが、戻りの通信も課金対象であるため実際にはもっと増えるはずです。

NAT Gateway は利用期間に応じた単価が高いのと、データ処理量に応じた料金が発生します。NAT インスタンスは、使わない時間は停止することでオンデマンド料金をさらに減らせます。

大抵の場合、NAT インスタンスの方がコストは安くつきます。

このコストの差を鑑みてもなお 本番環境では NAT Gateway の採用をお勧めします。NAT Gateway がマネージドでやってくれる部分や性能のメリットの方が大きいと考えるためです。検証環境や、万が一インターネットへのアウトバウンドが途切れてもクリティカルなことにはならない、というワークロードであればコスト抑制をモチベーションに NAT インスタンスを採用するのもアリだと思います。

それぞれの料金の以下をご参照ください。

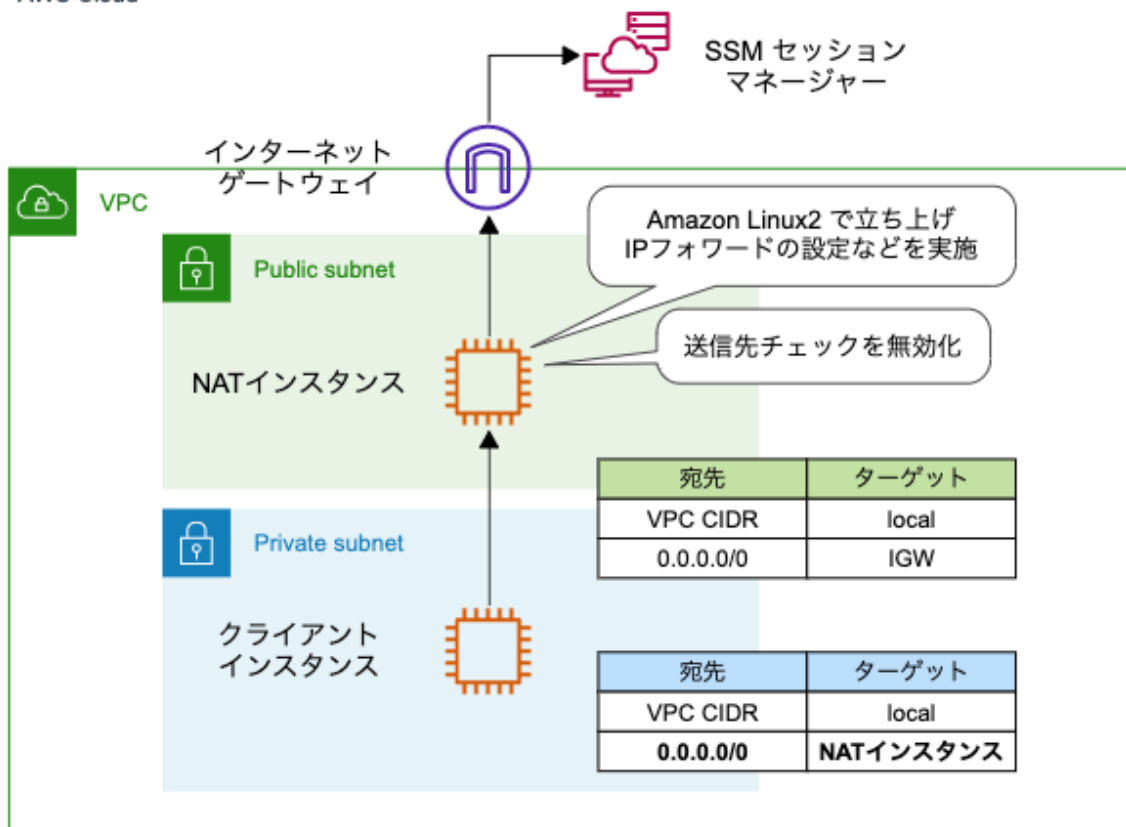
- [料金 - Amazon VPC | AWS](#)
- [オンデマンドインスタンスの料金 - Amazon EC2 \(仮想サーバー\) | AWS](#)
- [ハイパフォーマンスブロックストレージの料金](#) – [Amazon EBS の料金](#) – [Amazon Web Services](#)

## 個人の検証環境で使う分には便利じゃない？

はい、めっちゃ便利だと思います。

## NAT インスタンスの構成を作ってみた

今回は以下の構成を作成していきます。



最終的に、プライベートサブネットにあるクライアントインスタンスに SSM セッションマネージャーで接続できることをゴールとします。

(セッションマネージャー接続するためにはクライアントインスタンスから専用のエンドポイントに疎通できる必要がある。VPC エンドポイントが無い構成なのでインターネットゲートウェイ経由で接続できる必要がある。)

## 0. NAT インスタンスの作成（前置き）

作成は以下ページを参考にします。

- [NAT インスタンス - Amazon Virtual Private Cloud](#)

ページには以下の注意書きがあります。

**重要**

NAT AMI は、2020 年 12 月 31 日に標準サポートが終了した Amazon Linux の最新バージョン 2018.03 に基づいて構築されています。詳細については、ブログ投稿「[Amazon Linux AMI のサポート終了](#)」を参照してください。この AMI は、重要なセキュリティ更新だけを受け取ります (定期的な更新はありません)。

既存の NAT AMI を使用する場合は、AWS が [NATゲートウェイに移行](#)することを推奨します。NAT ゲートウェイでは、可用性と帯域幅に優れ、運用管理の手間を軽減できます。NAT インスタンスがユースケースに合致している場合は、独自の NAT AMI を作成できます。詳細については、「[NAT ゲートウェイと NAT インスタンスの比較](#)」を参照してください。

標準で用意されている NAT AMI は、2 ではない Amazon Linux をベースにしています。そのため、それをそのまま使用するのは避けるべきであり、いまから NAT インスタンスを利用するのであれば独自に作成する必要があります。

親切なことにセットアップ用のコマンドをドキュメントに記載してくれています。 <sup>\*2</sup>

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo /sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo yum install iptables-services
sudo service iptables save
```

前はこんな記述なかったよな.....? と思い調べたところ、2021年9月にドキュメントに追記されていました

- [Documentation updates · awsdocs/amazon-vpc-user-guide@c6a9b74](#)

## 1. NAT インスタンスの作成

以下の条件で EC2 インスタンスを作成しました。作成の詳細は割愛します。

- amzn2-ami-kernel-5.10-hvm-2.0.20221004.0-x86\_64-gp2 ( `ami-0de5311b2a443fb89` )
- パブリックサブネットに配置しパブリックIP 割り当て
- t3.micro、gp3、8 GiB
- IAM ロールを割り当て Systems Manager の権限付与
- SecurityGroup

- インバウンド：後続の手順で作成するクライアント用インスタンスからのすべてのトラフィック
- アウトバウンド：0.0.0.0/0 へのすべてのトラフィック

作成したインスタンスに SSM セッションマネージャーで接続し、先ほど確認したコマンドをベースに実行していきます。

IP フォワードの有効化を永続的に実施。

```
sh-4.2$ sudo sysctl -w net.ipv4.ip_forward=1 | sudo tee -a /etc/sysctl.conf
net.ipv4.ip_forward = 1
```

- 参考：[5.3. sysctl でカーネルパラメーターを永続的に設定 Red Hat Enterprise Linux 8 | Red Hat Customer Portal](#)

Iptablesでの NAT の設定。

```
sh-4.2$ sudo /sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

(せっかくなのでテーブルの中身を確認)

```
sh-4.2$ sudo iptables -t nat --list
Chain PREROUTING (policy ACCEPT)
target     prot opt source                destination

Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source                destination
MASQUERADE all  --  anywhere              anywhere
```

Iptables で設定した内容は再起動でクリアされるので、永続化するために `iptables-services` を導入。

```

sh-4.2$ sudo yum install iptables-services
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package iptables-services.x86_64 0:1.8.4-10.amzn2.1.2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                                Arch                                Version
=====
Installing:
 iptables-services                    x86_64                             1.8.4-10.amzn2.1.2

Transaction Summary
=====
Install 1 Package

Total download size: 58 k
Installed size: 24 k
Is this ok [y/d/N]: y
Downloading packages:
iptables-services-1.8.4-10.amzn2.1.2.x86_64.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : iptables-services-1.8.4-10.amzn2.1.2.x86_64
  Verifying  : iptables-services-1.8.4-10.amzn2.1.2.x86_64

Installed:
 iptables-services.x86_64 0:1.8.4-10.amzn2.1.2

Complete!

```

保存を実行します。

```

sh-4.2$ sudo service iptables save
iptables: Saving firewall rules to /etc/sysconfig/iptables:[ OK ]

```

iptables の自動起動を有効化します。

```
sh-4.2$ sudo systemctl enable iptables
Created symlink from /etc/systemd/system/basic.target.wants/iptables.service to /usr/lib/systemd/system/iptables.service.
```

これで OS 上のセットアップが完了しました。

## 1.a NAT インスタンスの再起動後の設定確認

手順として必要なわけではないですが、上記の手順で設定した内容が再起動によってクリアされないかを確認しておきます。

再起動してから接続し、各種設定を確認します。

`net.ipv4.ip_forward` が `1` であること。

```
sh-4.2$ sysctl -n net.ipv4.ip_forward
1
```

`iptables` が起動していること。

```
sh-4.2$ systemctl status iptables
• iptables.service - IPv4 firewall with iptables
   Loaded: loaded (/usr/lib/systemd/system/iptables.service; enabled; vendor preset: enabled)
   Active: active (exited) since Mon 2022-11-07 12:05:58 UTC; 4min 33s ago
   Process: 1770 ExecStart=/usr/libexec/iptables/iptables.init start (code=exited, status=0/SUCCESS)
   Main PID: 1770 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/iptables.service
```

`iptables` の `nat` テーブルの値が設定した通りであること。

```
sh-4.2$ sudo iptables -t nat --list
Chain PREROUTING (policy ACCEPT)
target     prot opt source                destination

Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```



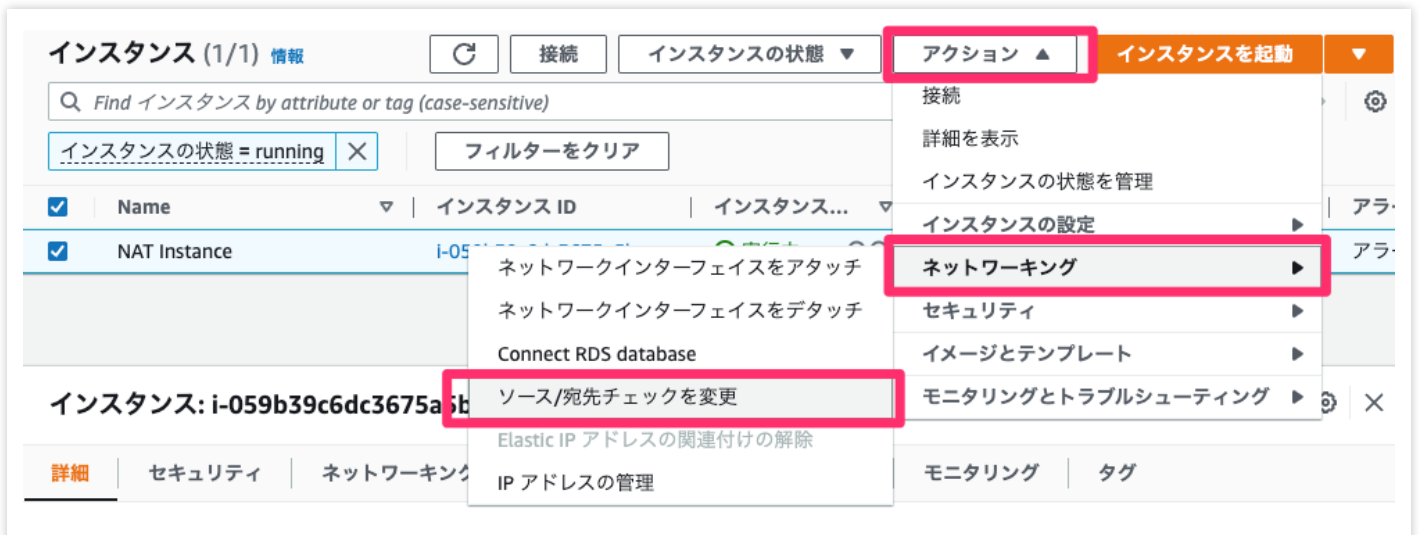
```
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE all  --  anywhere             anywhere
```

大丈夫そうですね！

## 2. NAT インスタンスの送信元/送信先チェックの無効化

EC2 インスタンス用の ENI はデフォルトで送信元/送信先チェック属性（ **SrcDestCheck** ）が有効になっています。有効な場合、自身が送信元/送信先でないトラフィックを破棄する挙動となるので、NAT インスタンスの場合は無効化してあげる必要があります。

「アクション」→「ネットワーキング」→「ソース/宛先チェックを変更」を選択します。



「停止」にチェックをつけ、「保存」を押下します。

### Change Source / destination check

The source / destination check ensures that the instance is the source or destination of all the traffic it sends and receives. Each EC2 instance performs source and destination checks by default. [詳細はこちら](#)

インスタンス ID  
**i-059b39c6dc3675a5b** (NAT Instance)

ネットワークインターフェイス  
**eni-085b4d83a6a63391f**

送信元/送信先チェック中  
Stop to allow your instance to send and receive traffic when the source or destination is not itself.

☒ 停止

キャンセル保存

ステータスはインスタンスの詳細画面のうち、「ネットワーキング」タブから確認できます。

インスタンス (1/1) 情報

接続

インスタンスの状態 ▼

アクション ▼

インスタンスを起動 ▼

インスタンスの状態 = running

フィルターをクリア

<input checked="" type="checkbox"/>	Name ▼	インスタンス ID	インスタンス... ▼	インスタンス... ▼	ステータスチェ...	アラ...
<input checked="" type="checkbox"/>	NAT Instance	i-059b39c6dc3675a5b	実行中	t3.micro	2/2 のチェックに1	アラ...

#### インスタンス: i-059b39c6dc3675a5b (NAT Instance)

アベイラビリティゾーン  
ap-northeast-1d

ゲスト OS ホスト名として RBN を使用  
無効

キャリア IP アドレス (一時的)  
-

RBN DNS ホスト名 IPv4 に応答  
有効

Outpost ID  
-

▼ ネットワークインターフェイス (1) 情報

ネット ID	終了時に削除	送信元/送信先チェック	セキュリティグループ	インターフェイスのタ...
i-051b5866e65...	有効	無効	sg-00f656e9c60e1f0d9 (launch-wi...	Elastic Network Interf...

### 3. クライアントインスタンスのサブネットルートテーブルの変更

このあとクライアントインスタンスを作成する先のサブネットのルートテーブルを編集します。

送信先を **0.0.0.0/0** とし、ターゲットを NAT インスタンスとしたルートを追加します。同じ VPC にあるインスタンスが候補として出てくるので選択するのは簡単です。

VPC > ルートテーブル > rtb-020184774e7e2cf47 > ルートを編集

#### ルートを編集

送信先	ターゲット	ステータス	伝播済み
172.31.0.0/16	local	アクティブ	いいえ
0.0.0.0/0	-	-	いいえ
	i-059b39c6dc3675a5b (NAT Instance)		

ルートを追加

キャンセル プレビュー **変更を保存**

ちなみに、変更が保存されると自動的にターゲットが NAT インスタンスの ENI に置き換わっていました。（はじめから ENI をターゲットとしてルート編集することも可能です。）

🟢 rtb-020184774e7e2cf47 / default-rt-pri-1dのルートを正常に更新しました

▶ 詳細

ℹ️ Reachability Analyzer でネットワーク接続を確認できるようになりました

Reachability Analyzer の実行

✕

詳細 情報

ルートテーブル ID  
📄 rtb-020184774e7e2cf47

VPC  
vpc-07ea4b9ce269fce6a

メイン  
📄 いいえ

所有者 ID  
📄

明示的なサブネットの関連付け  
subnet-0cba55e0e2b0a84be / default-pri-1d

Edge の関連付け  
-

ルート

サブネットの関連付け

Edge の関連付け

ルート伝播

タグ

ルート (2)

ルートを編集

🔍 ルートをフィルタリング

両方

< 1 >

⚙️

送信先	ターゲット	ステータス	伝播済み
0.0.0.0/0	eni-085b4d83a6a63391f	🟢 アクティブ	いいえ
172.31.0.0/16	local	🟢 アクティブ	いいえ

## 4. クライアントインスタンスの作成

以下設定で作成しました。詳細な手順は割愛します。

- amzn2-ami-kernel-5.10-hvm-2.0.20221004.0-x86\_64-gp2 ( `ami-0de5311b2a443fb89` )
- 先ほどルートテーブルを編集したプライベートサブネットに配置
- t3.micro、gp3、8 GiB
- IAM ロールを割り当て Systems Manager の権限付与
- SecurityGroup
  - インバウンド：なし
  - アウトバウンド：0.0.0.0/0 へのすべてのトラフィック

## 5. クライアントインスタンスからの疎通確認

クライアントインスタンスに対してセッションマネージャー接続を試みます。

インスタンス (1/2) 情報

🔄

接続

インスタンスの状態 ▼

アクション

🔍 Find インスタンス by attribute or tag (case-sensitive)

インスタンスの状態 = running ✕

フィルターをクリア

	Name ▼	インスタンス ID	インスタンス... ▼	イ... ▼	ステータスチェ... ▼	アベイラビリ... ▼	パブリック IPv4 ア...
<input type="checkbox"/>	NAT Instance	i-059b39c6dc...	🟢 実行中	🔍	t3.micro	🟢 2/2 のチェックに...	ap-northeast-1d 3.114.209.236
<input checked="" type="checkbox"/>	Client Instance	i-0d2c593c0b...	🟢 実行中	🔍	t3.micro	🟢 2/2 のチェックに...	ap-northeast-1d -

問題なく接続が完了しました。また、 `checkip.amazonaws.com` への curl も成功し、NAT インスタンスが持つパブリック IP アドレスが返却されました。

セッション ID: cm-chiba.yukihiro-      インスタンス ID: i-0d2c593c0b6bfa09f

```
sh-4.2$ curl http://checkip.amazonaws.com/
3.114.209.236
sh-4.2$
```

NAT インスタンスが正常に機能していることが確認できました。

## コストを取るかマネージドのメリットを取るか

NAT インスタンスを Amazon Linux2 の AMI から手作りで作ってみました。

NAT Gateway に比べればランニングコストは安い（ことが大抵の場合 期待できる）ですが、何かあった時に復旧作業するのに動く人のコストは？パッチ適用などを継続的に行うための運用コストは？などを考えると、マネージドの方に寄せたほうが嬉しいことが多い気がします。

何かあっても別に困らない、開発環境だし運用らしい運用も別にしないからランニングコストを低いことを優先する、といったワークロードでは採用してみるのもいいかもしれません。

以上、 [チバユキ \(@batchicchi\)](#) がお送りしました。

## あわせて読みたい

# 参考

---

- [Linuxで作るファイアウォール \[NAT設定編\] : ゼロから始めるLinuxセキュリティ \(4\)](#)  
(1/2 ページ) - @IT
- [natテーブルを利用したLinuxルータの作成: 習うより慣れろ! iptablesテンプレート集 \(2\)](#)  
(1/6 ページ) - @IT

# 脚注

---

1. 正確にはもっと課金要素がありますが、主要なものだけを記しています。 [↩](#)
2. とは言えここに書いてあるコマンドをそのまま使うだけだと再起動後に困ったことになります。 [↩](#)