

自宅とAWSをVPN接続してみた

#AWS VPN #Amazon VPC #AWS



Takuya Shibata

2019.05.03



37



79



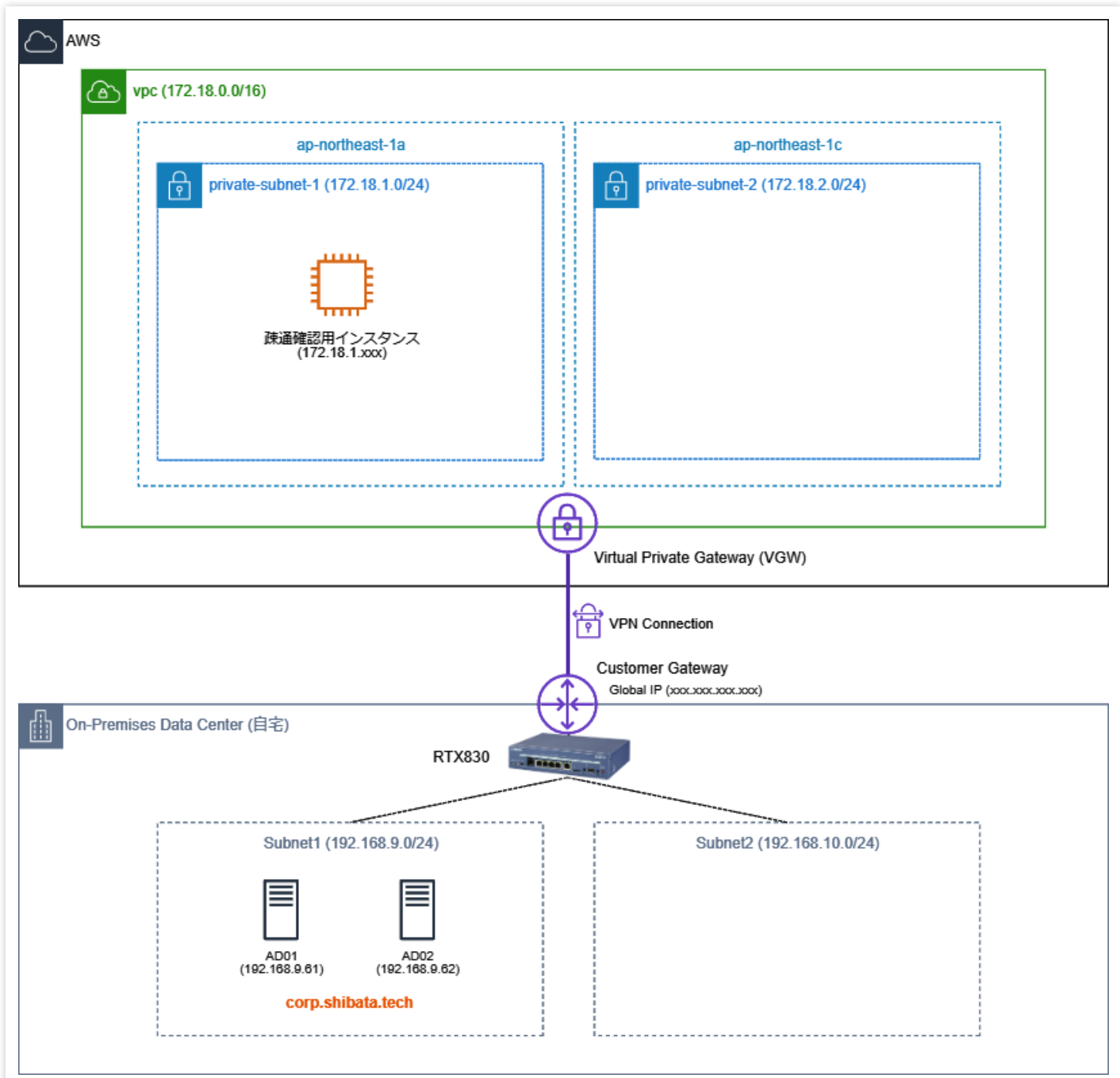
46

しばたです。

個人的な興味半分、業務上の理由半分でオンプレミスとAWSをDirect ConnectまたはVPNで接続した環境でのActive Directory関連の動作検証をしたくなり、Direct Connectを個人で試すのは難しいため、自宅をオンプレ環境に見立てVPNでAWSと接続し検証環境を作ることになりました。

構成図

構成図はこんな感じです。



AWS側VPCのCIDRは172.18.0.0/16、プライベートなサブネット172.18.1.0/24、172.18.2.0/24を用意しています。オンプレ(自宅)側は192.168.9.0/24と192.168.10.0/24の2つのネットワークと二台のドメインコントローラー(Windows Server 2012 R2)を用意しています。

構築手順

本記事ではAWS側環境を作り、オンプレ環境とVPN接続するところまでを解説します。

方式について

目的は違うのですが、つい先日弊社中山によってRTX830の「クラウド接続」機能をつかってお手軽にVPN環境を構築する手順が公開されました。

YAMAHA RTX830で手軽VPN接続

本記事では私自身の勉強を兼ねて上述の「クラウド接続」は使わずCloudFormationでAWS側のリソースを作成、手作業でRTX830のコンフィグを設定という手順で環境構築を行っています。環境構築という目的を果たすだけであれば「クラウド接続」の方が楽かと思いますのでそういった場合は中山の記事を参考にすると良いでしょう。

CloudFormationテンプレート

各種リソースを一つずつ作るのは面倒なのでCloudFormationで必要なネットワークリソースを一気に作成します。以下のテンプレートで、

- VPC
- NACL
- Subnet x 2
- RouteTable
- Security Group(SSH, RDP, ICMP)
- VGW
- VPNConnection
- Customer Gateway

を作成することができます。

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  SystemName:
    Description: "System name of each resource names."
    Type: String
    Default: "devio"
  EnvironmentName:
    Description: "Environment name of each resource names."
    Type: String
    Default: "test"
  AZ1:
    Description: "AZ1"
```

```

    Type: AWS::EC2::AvailabilityZone::Name
    Default: "ap-northeast-1a"
AZ2:
    Description: "AZ2"
    Type: AWS::EC2::AvailabilityZone::Name
    Default: "ap-northeast-1c"
CGWIP:
    Description: "IP address of Customer Gateway"
    Type: String
    Default: "xxx.xxx.xxx.xxx"
Outputs:
VPC1:
    Value:
        Ref: VPC1
    Export:
        Name:
            Fn::Sub: "${SystemName}-${EnvironmentName}-vpc"
PrivateSubnet1:
    Value:
        Ref: PrivateSubnet1
    Export:
        Name:
            Fn::Sub: "${SystemName}-${EnvironmentName}-private-subnet-1"
PrivateSubnet2:
    Value:
        Ref: PrivateSubnet2
    Export:
        Name:
            Fn::Sub: "${SystemName}-${EnvironmentName}-private-subnet-2"
PrivateRouteTable1:
    Value:
        Ref: PrivateRouteTable1
    Export:
        Name:
            Fn::Sub: "${SystemName}-${EnvironmentName}-private-rtb"
Resources:
# VPC
VPC1:
    Type: AWS::EC2::VPC
    Properties:
        CidrBlock: 172.18.0.0/16
        EnableDnsSupport: true
        EnableDnsHostnames: true
        Tags:
            - Key: Name
              Value:
                  Fn::Sub: "${SystemName}-${EnvironmentName}-vpc"
# Subnet
PrivateSubnet1:

```

```
Type: AWS::EC2::Subnet
Properties:
  AvailabilityZone:
    Ref: AZ1
  VpcId:
    Ref: VPC1
  CidrBlock: 172.18.1.0/24
  MapPublicIpOnLaunch: False
  Tags:
    - Key: Name
      Value:
        Fn::Sub: "${SystemName}-${EnvironmentName}-private-subnet-1"
PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    AvailabilityZone:
      Ref: AZ2
    VpcId:
      Ref: VPC1
    CidrBlock: 172.18.2.0/24
    MapPublicIpOnLaunch: False
    Tags:
      - Key: Name
        Value:
          Fn::Sub: "${SystemName}-${EnvironmentName}-private-subnet-2"
# Customer Gateway
CustomerGateway1:
  Type: AWS::EC2::CustomerGateway
  Properties:
    Type: "ipsec.1"
    BgpAsn: 65000
    IpAddress:
      Ref: CGWIP
    Tags:
      - Key: Name
        Value:
          Fn::Sub: "${SystemName}-${EnvironmentName}-cgw"
# VPN Gateway
VPNGateway1:
  Type: AWS::EC2::VPNGateway
  Properties:
    Type: ipsec.1
    Tags:
      - Key: Name
        Value:
          Fn::Sub: "${SystemName}-${EnvironmentName}-vpw"
  AttachVpnGateway1:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
```

```
VpcId:
  Ref: VPC1
VpnGatewayId:
  Ref: VPNGateway1
# VPN Connection
VPNConnection1:
  Type: AWS::EC2::VPNConnection
  Properties:
    Type: ipsec.1
    StaticRoutesOnly: False
    CustomerGatewayId:
      Ref: CustomerGateway1
    VpnGatewayId:
      Ref: VPNGateway1
# Route Tables
PrivateRouteTable1:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId:
      Ref: VPC1
    Tags:
      - Key: Name
        Value:
          Fn::Sub: "${SystemName}-${EnvironmentName}-private-rtb"
PrivateRoute1:
  Type: AWS::EC2::Route
  DependsOn: AttachVpnGateway1
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable1
    DestinationCidrBlock: 192.168.0.0/16
    GatewayId:
      Ref: VPNGateway1
PrivateRTAssociation1:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable1
    SubnetId:
      Ref: PrivateSubnet1
PrivateRTAssociation2:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    RouteTableId:
      Ref: PrivateRouteTable1
    SubnetId:
      Ref: PrivateSubnet2
# NACL (Default only)
NACL1:
```

```
Type: AWS::EC2::NetworkAcl
Properties:
  Tags:
    - Key: Name
      Value:
        Fn::Sub: "${SystemName}-${EnvironmentName}-nACL"
  VpcId:
    Ref: VPC1
NACLEntry1:
  Type: AWS::EC2::NetworkAclEntry
  Properties:
    Egress: true
    CidrBlock: 0.0.0.0/0
    Protocol: -1
    RuleAction : allow
    RuleNumber : 100
    NetworkAclId:
      Ref: NACL1
NACLEntry2:
  Type: AWS::EC2::NetworkAclEntry
  Properties:
    Egress: false
    CidrBlock: 0.0.0.0/0
    Protocol: -1
    RuleAction : allow
    RuleNumber : 100
    NetworkAclId:
      Ref: NACL1
SubnetNACLAssociation1:
  Type: AWS::EC2::SubnetNetworkAclAssociation
  Properties:
    SubnetId:
      Ref: PrivateSubnet1
    NetworkAclId:
      Ref: NACL1
SubnetNACLAssociation2:
  Type: AWS::EC2::SubnetNetworkAclAssociation
  Properties:
    SubnetId:
      Ref: PrivateSubnet2
    NetworkAclId:
      Ref: NACL1
# Security Group (SSH)
SecurityGroupSSH:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName:
      Fn::Sub: "${SystemName}-${EnvironmentName}-ssh-sg"
    GroupDescription:
```

```
    Fn::Sub: "${SystemName}-${EnvironmentName}-ssh-sg"
VpcId:
  Ref: VPC1
SecurityGroupIngress:
  - IpProtocol: tcp
    FromPort: 22
    ToPort: 22
    CidrIp: 172.18.0.0/16
  - IpProtocol: tcp
    FromPort: 22
    ToPort: 22
    CidrIp: 192.168.0.0/16
Tags:
  - Key: Name
    Value:
      Fn::Sub: "${SystemName}-${EnvironmentName}-ssh-sg"
# Security Group (RDP)
SecurityGroupRDP:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName:
      Fn::Sub: "${SystemName}-${EnvironmentName}-rdp-sg"
    GroupDescription:
      Fn::Sub: "${SystemName}-${EnvironmentName}-rdp-sg"
    VpcId:
      Ref: VPC1
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 3389
        ToPort: 3389
        CidrIp: 172.18.0.0/16
      - IpProtocol: tcp
        FromPort: 3389
        ToPort: 3389
        CidrIp: 192.168.0.0/16
    Tags:
      - Key: Name
        Value:
          Fn::Sub: "${SystemName}-${EnvironmentName}-rdp-sg"
# Security Group (Ping)
SecurityGroupICMP:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupName:
      Fn::Sub: "${SystemName}-${EnvironmentName}-icmp-sg"
    GroupDescription:
      Fn::Sub: "${SystemName}-${EnvironmentName}-icmp-sg"
    VpcId:
      Ref: VPC1
```


SecurityGroupIngress:

- IpProtocol: icmp
FromPort: 8
ToPort: -1
CidrIp: 172.18.0.0/16
- IpProtocol: icmp
FromPort: 8
ToPort: -1
CidrIp: 192.168.0.0/16

Tags:

- Key: Name
Value:
Fn::Sub: "\${SystemName}-\${EnvironmentName}-icmp-sg"

The screenshot shows the AWS CloudFormation console. At the top, there's a navigation bar with 'CloudFormation' and 'スタック' (Stacks). Below it, there are tabs for 'スタックの作成' (Create Stack), 'アクション' (Actions), and 'テンプレートのデザイン' (Template Design). A filter bar shows 'フィルター: アクティブ' and 'スタックの名前でフィルター'. The main table lists stacks, with 'devio-test-network' highlighted. Below the table, there are tabs for '概要' (Overview), '出力' (Outputs), 'リソース' (Resources), 'イベント' (Events), 'テンプレート' (Template), 'パラメータ' (Parameters), 'タグ' (Tags), 'スタックのポリシー' (Stack Policies), '変更セット' (Change Sets), and 'ロールバックトリガー' (Rollback Triggers). The 'イベント' tab is selected, showing a list of events for the 'devio-test-network' stack. The events include the creation of various resources, all in a 'CREATE_COMPLETE' state, except for the last one which is 'CREATE_IN_PROGRESS'.

| スタックの名前 | 作成時間 | 状況 | ドリフトステータス | 説明 |
|--------------------|------------------------------|-----------------|-------------|----|
| devio-test-network | 2019-04-05 10:57:31 UTC+0900 | CREATE_COMPLETE | NOT_CHECKED | |

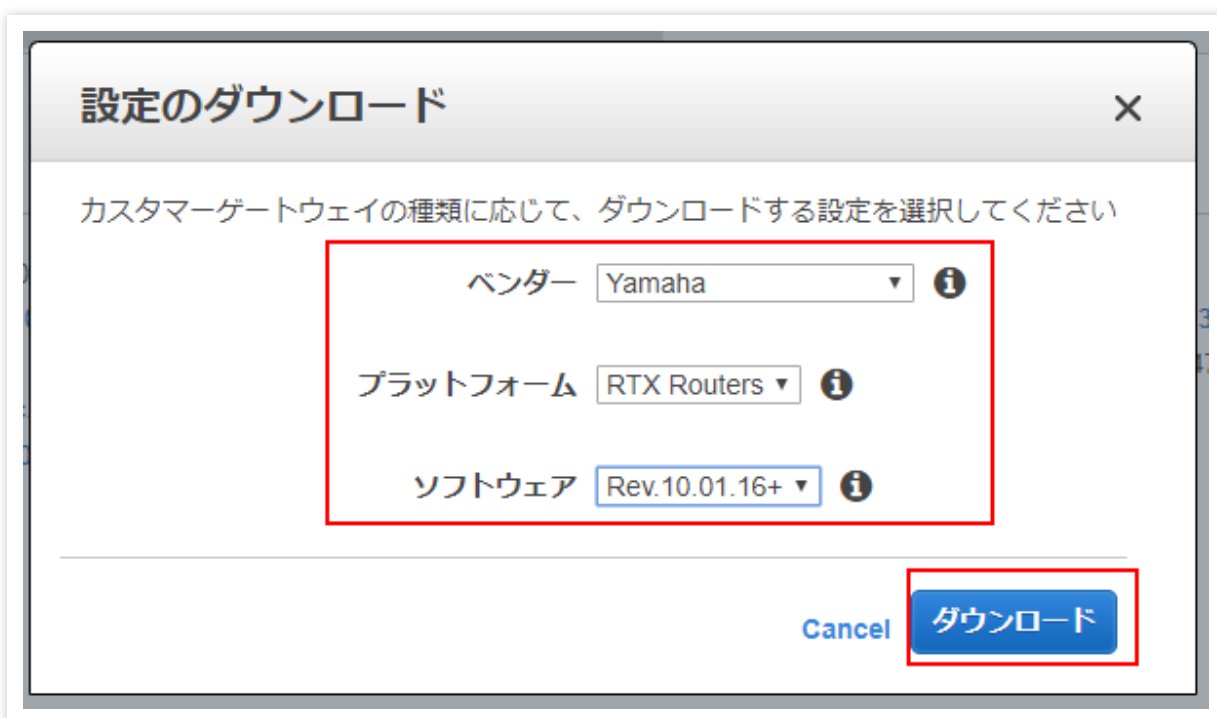
| 2019-04-05 | 状況 | タイプ | 論理 ID | 状況の理由 |
|---------------------|--------------------|---------------------------------------|------------------------|-----------------------------|
| ▶ 11:01:04 UTC+0900 | CREATE_COMPLETE | AWS::CloudFormation::Stack | devio-test-network | |
| ▶ 11:01:00 UTC+0900 | CREATE_COMPLETE | AWS::EC2::VPNConnection | VPNConnection1 | |
| ▶ 10:58:36 UTC+0900 | CREATE_COMPLETE | AWS::EC2::SubnetRouteTableAssociation | PrivateRTAssociation1 | |
| ▶ 10:58:36 UTC+0900 | CREATE_COMPLETE | AWS::EC2::SubnetRouteTableAssociation | PrivateRTAssociation2 | |
| ▶ 10:58:36 UTC+0900 | CREATE_COMPLETE | AWS::EC2::SubnetNetworkAclAssociation | SubnetNACLAssociation1 | |
| ▶ 10:58:36 UTC+0900 | CREATE_COMPLETE | AWS::EC2::SubnetNetworkAclAssociation | SubnetNACLAssociation2 | |
| ▶ 10:58:22 UTC+0900 | CREATE_COMPLETE | AWS::EC2::NetworkAclEntry | NACLEntry2 | |
| ▶ 10:58:22 UTC+0900 | CREATE_COMPLETE | AWS::EC2::Route | PrivateRoute1 | |
| ▶ 10:58:22 UTC+0900 | CREATE_COMPLETE | AWS::EC2::NetworkAclEntry | NACLEntry1 | |
| ▶ 10:58:21 UTC+0900 | CREATE_IN_PROGRESS | AWS::EC2::SubnetRouteTableAssociation | PrivateRTAssociation1 | Resource creation Initiated |

VPN接続

先述のCloudFormationテンプレートからスタックを作成するとVPN接続が1つ作成されているのでマネジメントコンソールから確認してみます。



この接続に対してコンソール上部にある「設定のダウンロード」ボタンをクリックするとオンプレ側ルーターに設定するコンフィグファイルをダウンロードすることができます。



ベンダー、プラットフォーム、ソフトウェアの欄を適切に選択して「ダウンロード」してください。

ダウンロードしたファイルは以下の様な感じでIKE、IPSec、BGPの設定内容が記載されています。

```

vpn-1.txt - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
# Amazon Web Services
# Virtual Private Cloud

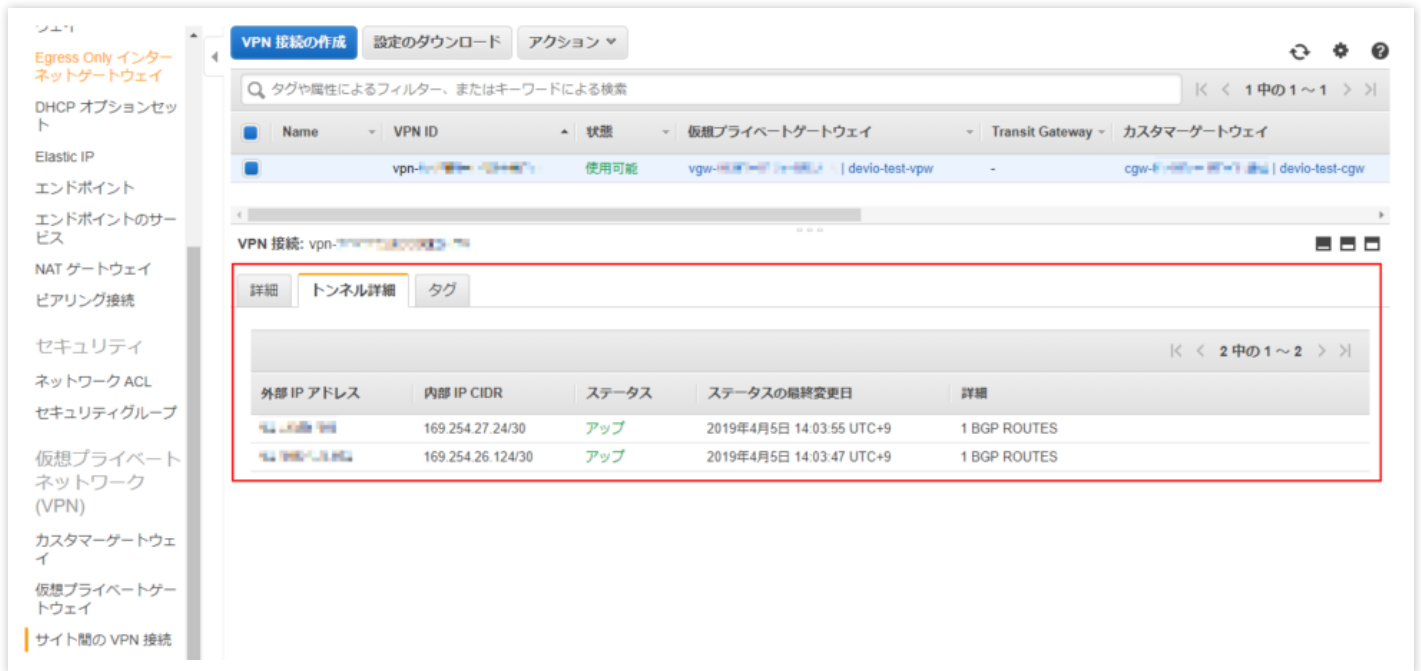
# AWS utilizes unique identifiers to manage the configuration of
# a VPN Connection. Each VPN Connection is assigned an identifier and is
# associated with two other identifiers, namely the
# Customer Gateway Identifier and Virtual Private Gateway Identifier.
#
# Your VPN Connection ID           : vpn-072600ba03223dc0b15d
# Your Virtual Private Gateway ID  : vgw-072600ba03223dc0b15d
# Your Customer Gateway ID         : cgw-05c71f1ce3d01a1548d4
#
# This configuration consists of two tunnels. Both tunnels must be
# configured on your Customer Gateway.
#
# -----
# IPSec Tunnel #1
# -----

# #1: Internet Key Exchange (IKE) Configuration
#
# A policy is established for the supported ISAKMP encryption,
# authentication, Diffie-Hellman, lifetime, and key parameters.
#
# Please note, these sample configurations are for the minimum requirement of AES128, SHA1, and DH Group 2.
# Category "VPN" connections in the GovCloud region have a minimum requirement of AES128, SHA2, and DH Group 14.
# You will need to modify these sample configuration files to take advantage of AES256, SHA256, or other DH groups like 2,
# Higher parameters are only available for VPNs of category "VPN," and not for "VPN-Classic".
# The address of the external interface for your customer gateway must be a static address.
# Your customer gateway may reside behind a device performing network address translation (NAT).
# To ensure that NAT traversal (NAT-T) can function, you must adjust your firewall rules to unblock UDP port 4500. If not
#
    tunnel select 1
        ipsec ike encryption 1 aes-cbc
        ipsec ike group 1 modp1024
        ipsec ike hash 1 sha

```

この内容をオンプレにあるRTX830に適用していきます。オンプレ側環境は千差万別ですので、コンフィグファイルの適用前にその内容はきちんと確認しておいた方が良いでしょう。(初めてVPNを張る場合といったシンプルな環境あればコンフィグの内容をまるっと実行しても問題ないかと思われます)

オンプレ側の設定が反映されれば下図の様にトンネルが張れているはずです。



VPN 接続: vpn-test-vpn

詳細 トンネル詳細 タグ

| 外部 IP アドレス | 内部 IP CIDR | ステータス | ステータスの最終変更日 | 詳細 |
|-------------------|------------|-------|--------------------------|--------------|
| 169.254.27.24/30 | | アップ | 2019年4月5日 14:03:55 UTC+9 | 1 BGP ROUTES |
| 169.254.26.124/30 | | アップ | 2019年4月5日 14:03:47 UTC+9 | 1 BGP ROUTES |

Ping確認

最後にAWS側に疎通確認用インスタンスを立ててオンプレ側のドメインコントローラー (192.168.9.61、192.168.9.62)にPingを打ってみるときちんと疎通できていることが確認できました。(EC2の構築手順は省略)

```
管理: Windows PowerShell
PS C:\Users\Administrator> ipconfig

Windows IP 構成

イーサネット アダプター イーサネット:

    接続固有の DNS サフィックス . . . . . : ap-northeast-1.compute.internal
    リンクローカル IPv6 アドレス . . . . . : fe80::fdc8:b07:2541:c686%2
    IPv4 アドレス . . . . . : 172.18.1.139
    サブネット マスク . . . . . : 255.255.255.0
    デフォルト ゲートウェイ . . . . . : 172.18.1.1

Tunnel adapter isatap.ap-northeast-1.compute.internal:

    メディアの状態 . . . . . : メディアは接続されていません
    接続固有の DNS サフィックス . . . . . : ap-northeast-1.compute.internal

Tunnel adapter ローカル エリア接続* 3:

    接続固有の DNS サフィックス . . . . . :
    IPv6 アドレス . . . . . : 2001:0:2841:aa90:1865:727:8d47:f16c
    リンクローカル IPv6 アドレス . . . . . : fe80::1865:727:8d47:f16c%5
    デフォルト ゲートウェイ . . . . . :
PS C:\Users\Administrator> ping 192.168.9.61

192.168.9.61 に ping を送信しています 32 バイトのデータ:
192.168.9.61 からの応答: バイト数 = 32 時間 = 22ms TTL=127
192.168.9.61 からの応答: バイト数 = 32 時間 = 21ms TTL=127
192.168.9.61 からの応答: バイト数 = 32 時間 = 21ms TTL=127
192.168.9.61 からの応答: バイト数 = 32 時間 = 21ms TTL=127

192.168.9.61 の ping 統計:
    パケット数: 送信 = 4、受信 = 4、損失 = 0 (0% の損失)、
    ラウンド トリップの概算時間 (ミリ秒):
        最小 = 21ms、最大 = 22ms、平均 = 21ms
PS C:\Users\Administrator> ping 192.168.9.62

192.168.9.62 に ping を送信しています 32 バイトのデータ:
192.168.9.62 からの応答: バイト数 = 32 時間 = 22ms TTL=127
192.168.9.62 からの応答: バイト数 = 32 時間 = 28ms TTL=127
192.168.9.62 からの応答: バイト数 = 32 時間 = 21ms TTL=127
192.168.9.62 からの応答: バイト数 = 32 時間 = 21ms TTL=127

192.168.9.62 の ping 統計:
    パケット数: 送信 = 4、受信 = 4、損失 = 0 (0% の損失)、
    ラウンド トリップの概算時間 (ミリ秒):
        最小 = 21ms、最大 = 28ms、平均 = 23ms
PS C:\Users\Administrator>
```

最後に

ざっとこんな感じです。この環境をベースに本来の目的である検証作業を進めていきたいと思ひます。