An abstract 3D graphic at the top of the page features several rectangular blocks of varying sizes and orientations, appearing to float in a white space. Some blocks are white, some are grey, and some are red. They are arranged in a way that suggests depth and movement, with some blocks overlapping others.

FUJITSU Hybrid IT Service FJcloud-O

IaaS API ユーザーズガイド

Version 1.19.4
FUJITSU LIMITED

All Rights Reserved, Copyright 富士通株式会社 2015-2023

まえがき

本書の目的

本書は FUJITSU Hybrid IT Service FJcloud-O IaaS (以降、本サービス) が提供するサービスを API (Application Programming Interface) で使用するための環境を構築し、チームでの構築のためのユーザー管理までを説明しています。また、付録では、最も基本的な構成を構築する手順を示しています。

本書の読者

本サービスは基本的にすべての機能を API で実行でき、API により環境構築、運用を行うことを前提としています。

本ガイドは API 操作による環境構築、運用を行うにあたり、実行環境の設定方法およびプロジェクト・ユーザー・ユーザーのグループを管理する方法を実行サンプル例とともに解説します。

本ガイドが対象としている API のユーザーは、事前に所定のユーザー申請を行い、FUJITSU Hybrid IT Service FJcloud-O のユーザーアカウント (ユーザー名およびパスワード) を取得されていることを前提とします。

API の利用に際し、以下の前提知識が必要です。

- Web サービスに関する基本的事項
- UNIX 環境に関する基本的事項



本ガイドに掲載している shell およびコマンドにつきましてはセキュリティおよび動作を保証するものではありませんので、内容をご確認の上ご利用ください。また、オペレーション・出力例は OS バージョンやクライアント環境によって異なり記載通りではない場合があります。適宜読み替えてご対応願います。

本書の対象リージョン

本書の対象リージョンは、東日本リージョン1、東日本リージョン2、西日本リージョン1、西日本リージョン2です。

マニュアル体系

目的・用途に合わせて、以下の関連マニュアルもお読みください

マニュアル名称	目的・用途
IaaS API ユーザーズガイド (本ガイド)	REST API の使い方について、API 実行環境の構築方法、利用シーケンスにあわせたサンプルスクリプトなどを説明した資料です。
IaaS 機能説明書 (以下、機能説明書)	本サービスが提供する機能詳細を解説した資料です。API 操作時、ユーザー入力が必要な項目の制限値に関する記載は機能説明書にございます。必要に応じてご参照ください。
IaaS API リファレンスマニュアル (以下、API リファレンスマニュアル)	REST API を利用する際の詳細リファレンスとしてご参照・ご利用ください。
IaaS HEAT テンプレート解説書	オーケストレーション機能を利用する際に記述する、Heat Orchestration Template (HOT) の書式について説明した資料です。
IaaS ポータルユーザーズガイド	IaaS ポータル (Web GUI) を使用して、IaaS が提供する機能の使い方を説明した資料です。
FJcloud ポータルユーザーズガイド (以降ポータルユーザーズガイドと略)	FUJITSU Hybrid IT Service FJcloud サービス全体を管理する FJcloud ポータル (以降ポータルと記載) が提供する機能 (利用登録、ユーザ情報管理など) の使い方を説明した資料です。

本文中の略称

本書では、製品名を以下のように表記しています。

正式名称	略称	
FUJITSU Hybrid IT Service FJcloud-O IaaS	IaaS	
Microsoft® Windows Server® 2012 SE R2	Windows 2012 R2	Windows
Microsoft® Windows Server® 2008 SE R2	Windows 2008 R2	
Microsoft® Windows Server® 2008 EE R2		
Red Hat® Enterprise Linux®	-	Linux
Community Enterprise Operating System	CentOS	
Red Hat Update Infrastructure	RHUI	
Windows Server Update Services	WSUS	

商標

- Microsoft、Windows、Windows Serverまたはその他のマイクロソフト製品の名称および製品名は、米国Microsoft Corporationの、米国およびその他の国における登録商標または商標です。
 - Javaは、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。
 - Xeonは、アメリカ合衆国および/またはその他の国における Intel Corporation の商標です。
 - Linux®は米国及びその他の国におけるLinus Torvaldsの登録商標です。
 - Red Hat、Red Hat Enterprise Linuxは米国およびその他の国において登録されたRed Hat, Inc.の商標です。
 - Ubuntuは、Canonical Ltd.の登録商標です。
 - OpenStackのワードマークは、米国とその他の国におけるOpenStack Foundationの登録商標/サービスマークまたは商標/サービスマークのいずれかであり、OpenStack Foundationの許諾の下に使用されています。
 - そのほか、本書に記載されている会社名および製品名は、それぞれ各社の商標または登録商標です。
- なお、本書では、システム名または製品名に付記される登録表示(™または®)は、省略しています。

輸出管理規制

本ドキュメントを輸出または第三者へ提供する場合は、お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ、必要な手続きをおとりください。

お願い

- 本書は、予告なしに変更されることがあります。
- 本書を無断で他に転用しないようお願いします。
- 本書に記載されたデータの使用に起因する第三者の特許権およびその他の権利の侵害については、当社はその責を負いません。

変更履歴

版数	更新日	変更箇所	概要
1.1	2016年1月18日	全般	誤字の訂正
		全般	図の訂正
		shellの準備	説明追記
1.2	2016年2月18日	shellの準備	説明追記
		ロールの付与および参照	誤記修正
		全般	目次ジャンプ機能追加
1.3	2016年4月1日	shellの準備	説明追記
		全般	ユーザー操作APIについての記述を削除
1.4	2016年10月6日	APIを利用上の注意点	「API利用上の注意点」を追加
		環境の説明	説明追記
		shellの準備	説明追記
		グループ作成	説明追記
		グループにユーザーを追加	説明追記
		グループ削除	説明追記
		全般	誤記修正
1.5	2016年12月27日	仮想サーバ作成準備(仮想サーバタイプ・フレーバー一覧の取得)	説明修正
		全般	誤記修正
		shellの準備	誤記修正
		セキュリティグループルール作成	誤記修正
1.6	2017年2月20日	システム構成図	説明修正
		SSL-VPN用証明書作成(V1サービス)	説明追加
		SSL-VPN用証明書登録(V1サービス)	説明追加
		SSL-VPN用鍵コンテナ作成(V1サービス)	説明追加
		SSL-VPNへのグローバルIPアドレスの割当て(V1サービス)	説明追加
		VPNサービスの作成(V1サービス)	説明追加
		SSL-VPN接続の作成(V1サービス)	説明追加
		VPNクライアントとの接続設定(V1サービス)	説明追加
		SSL-VPN接続の削除(V1サービス)	説明追加

版数	更新日	変更箇所	概要
		VPNサービスの削除 (V1サービス)	説明追加
		SSL-VPN用グローバルIPアドレスの削除 (V1サービス)	説明追加
		SSL-VPN用鍵コンテナの削除 (V1サービス)	説明追加
		SSL-VPN用証明書の削除 (V1サービス)	説明追加
		第 4 章:付録	構成修正
1.7	2017年4月20日	システム構成図	説明修正
		SSL-VPN接続 (V2サービス/クライアント証明書利用)	説明追加
		SSL-VPN接続 (V2サービス/自己署名証明書利用)	説明追加
		SSL-VPN接続削除 (V2サービス)	説明追加
		SSL-VPN接続 (V1サービス)	説明削除
		SSL-VPN接続削除 (V1サービス)	説明削除
		全般	構成修正
1.8	2017年6月6日	SSL-VPN接続の作成 (V2サービス/クライアント証明書利用)	誤記修正
		VPNクライアントとの接続設定 (V2サービス/クライアント証明書利用)	誤記修正
		SSL-VPN用証明書登録 (V2サービス/自己署名証明書利用)	誤記修正
		SSL-VPN接続の作成 (V2サービス/自己署名証明書利用)	誤記修正
		VPNクライアントとの接続設定 (V2サービス/自己署名証明書利用)	誤記修正
		SSL-VPN接続の削除 (V2サービス)	誤記修正
		SSL-VPN接続の削除 (V2サービス)	誤記修正
		SSL-VPN用鍵コンテナの削除 (V2サービス)	誤記修正
		SSL-VPN用証明書の削除 (V2サービス)	誤記修正
1.9	2017年6月23日	shellの準備	説明修正
		tokenの取得	説明修正
		システム構成図	誤記修正
		VPNクライアントとの接続設定 (V2サービス/クライアント証明書利用)	説明修正
		証明書の形式変換	説明追加
1.10	2017年7月26日	はじめに	説明追加
		セキュリティグループルール作成	説明修正
		セキュリティグループルールの設定例	誤記修正
		ファイアウォールルール (IPアドレス指定) 作成	説明修正

版数	更新日	変更箇所	概要
		ファイアウォールルール(ポート番号指定)作成	説明修正
		ファイアウォールルール(拒否ルール)作成	説明修正
		ファイアウォールポリシー作成	説明修正
		ファイアウォール作成	説明修正
		サブネット作成	説明修正
		SSL-VPNコネクションの作成(V2サービス/クライアント証明書利用)	説明修正
		VPNクライアントとの接続設定(V2サービス/クライアント証明書利用)	説明修正
		SSL-VPNコネクションの作成(V2サービス/自己署名証明書利用)	説明修正
		VPNクライアントとの接続設定(V2サービス/自己署名証明書利用)	説明修正
		ポート作成	説明修正
		証明書の形式変換	説明追加
1.11	2017年9月1日	全般	コマンドの改行位置に「¥」を挿入
		キーペア作成	説明修正
		shellの準備	誤記修正
1.12	2017年10月19日	VPNクライアントとの接続設定(V2サービス/クライアント証明書利用)	説明削除
		VPNクライアントとの接続設定(V2サービス/自己署名証明書利用)	説明削除
1.13	2017年12月4日	まえがき	誤記修正
		サブネット作成	説明修正
		VPNサービスの作成(V2サービス/クライアント証明書利用)	説明修正
		VPNサービスの作成(V2サービス/自己署名証明書利用)	説明修正
		VPNクライアントとの接続設定(V2サービス/クライアント証明書利用)	説明追加
		VPNクライアントとの接続設定(V2サービス/自己署名証明書利用)	説明追加
		セキュリティグループルール作成	コマンドの改行位置を変更
1.14	2017年12月22日	環境の説明	説明修正
		VPNクライアントとの接続設定(V2サービス/クライアント証明書利用)	説明追加
		APIアクセス環境の設定(Windows)	説明追加

版数	更新日	変更箇所	概要
1.15	2018年1月26日	tokenの取得	出力結果修正
		SSL-VPN接続の作成 (V2サービス/クライアント証明書利用)	説明追加
		VPNクライアントとの接続設定 (V2サービス/クライアント証明書利用)	説明修正
		SSL-VPN接続の作成 (V2サービス/自己署名証明書利用)	説明追加
		VPNクライアントとの接続設定 (V2サービス/自己署名証明書利用)	説明修正
1.16	2018年2月22日	tokenの取得	誤記修正
		仮想ルータの情報変更(外部ネットワークにアタッチ)	説明追加
		SSL-VPN接続 (V2サービス/クライアント証明書利用)	説明追加
		VPNクライアントとの接続設定 (V2サービス/クライアント証明書利用)	説明修正
		SSL-VPN接続 (V2サービス/自己署名証明書利用)	説明追加
		SSL-VPN用証明書登録 (V2サービス/自己署名証明書利用)	説明修正
		SSL-VPN用鍵コンテナ作成 (V2サービス/自己署名証明書利用)	説明修正
		VPNクライアントとの接続設定 (V2サービス/自己署名証明書利用)	説明修正
1.17	2018年3月22日	サブネット作成	説明追加
		セキュリティグループルール作成	誤記修正
		セキュリティが高いネットワーク構成のSSL-VPN接続 (V2サービス)	説明追加
1.18	2018年4月19日	セキュリティグループルール作成	説明修正
		ファイアウォール作成	説明修正
		簡易構成のSSL-VPN接続 (V2サービス)	章名修正
		SSL-VPN接続の作成 (V2サービス/クライアント証明書利用)	説明修正
		SSL-VPN接続の作成 (V2サービス/自己署名証明書利用)	説明修正
1.18.1	2018年6月29日	まえがき	本書の対象リージョンを記載
		shellの準備	誤記修正
1.18.2	2018年12月10日	全体	レイアウトを変更 (HTML版のみ)
		表紙他	サービス名称変更

版数	更新日	変更箇所	概要
1.18.3	2020年6月11日	まえがき	本書の対象リージョンを変更
		表紙他	サービス名称変更
1.18.4	2020年7月7日	VPNクライアントとの接続設定 (V2サービス/クライアント証明書利用)	一部のリージョンの中間証明書、root証明書を変更
1.19	2020年8月3日	VPNクライアントとの接続設定 (V2サービス/クライアント証明書利用)	東1,西2リージョンで使用する中間証明書、root証明書を変更 (1.18.4と同じ変更の展開)
1.19.1	2021年4月6日	まえがき他	FJcloudポータルのも名称、参照マニュアルの名称を最新に修正
1.19.2	2021年12月20日	VPNクライアントとの接続設定 (V2サービス/自己署名証明書利用)	自己署名証明書利用の場合の注意事項を追加
1.19.3	2022年10月6日	VPNクライアントとの接続設定 (V2サービス/クライアント証明書利用)	使用する中間証明書を変更
1.19.4	2023年10月27日	まえがき	説明修正
		VPNクライアントとの接続設定 (V2サービス/クライアント証明書利用)	使用する中間証明書,root証明書を変更

目次

第1章 はじめに.....	1
1.1 本書で使用するコマンド.....	2
第2章 環境構築の準備.....	4
2.1 API利用環境の構築.....	5
2.1.1 環境の説明.....	5
2.1.2 shellの準備.....	5
2.1.3 tokenの取得.....	9
2.1.4 APIを利用上の注意点.....	10
2.2 プロジェクト管理.....	11
2.2.1 プロジェクト作成.....	11
2.3 グループ管理.....	13
2.3.1 グループ作成.....	13
2.3.2 グループにユーザーを追加.....	14
2.3.3 グループ削除.....	15
2.4 ユーザー/グループのロール管理.....	17
2.4.1 プリセットロール一覧の確認.....	17
2.4.2 ロールの付与および参照.....	18
第3章 リソースの作成.....	20
3.1 想定システム構成.....	21
3.1.1 システム構成図.....	21
3.2 ネットワーク環境の構築.....	24
3.2.1 ネットワーク作成.....	24
3.2.2 サブネット作成.....	26
3.2.2.1 サブネット作成.....	26
3.2.2.2 ルーティングの追加.....	28
3.2.3 仮想ルータ作成.....	29
3.2.4 仮想ルータの情報変更(外部ネットワークにアタッチ).....	31
3.2.5 仮想ルータの情報変更(サブネットにアタッチ).....	32
3.3 セキュリティグループ設定.....	33
3.3.1 セキュリティグループ作成.....	33
3.3.2 セキュリティグループルール作成.....	34
3.3.3 セキュリティグループルールの設定例.....	36
3.4 ファイアウォール作成.....	38
3.4.1 ファイアウォールルール(IPアドレス指定)作成.....	38
3.4.2 ファイアウォールルール(ポート番号指定)作成.....	39
3.4.3 ファイアウォールルール(ICMP許可)作成.....	41

3.4.4	ファイアーウォールルール(拒否ルール)作成.....	43
3.4.5	ファイアーウォールポリシー作成.....	45
3.4.6	ファイアーウォール作成.....	47
3.5	簡易構成のSSL-VPN接続(V2サービス).....	49
3.5.1	SSL-VPN接続(V2サービス/クライアント証明書利用).....	49
3.5.1.1	VPNサービスの作成(V2サービス/クライアント証明書利用).....	49
3.5.1.2	SSL-VPNコネクションの作成(V2サービス/クライアント証明書利用).....	50
3.5.1.3	VPNクライアントとの接続設定(V2サービス/クライアント証明書利用).....	54
3.5.2	SSL-VPN接続(V2サービス/自己署名証明書利用).....	56
3.5.2.1	SSL-VPN用証明書作成(V2サービス/自己署名証明書利用).....	56
3.5.2.2	SSL-VPN用証明書登録(V2サービス/自己署名証明書利用).....	59
3.5.2.3	SSL-VPN用鍵コンテナ作成(V2サービス/自己署名証明書利用).....	64
3.5.2.4	VPNサービスの作成(V2サービス/自己署名証明書利用).....	65
3.5.2.5	SSL-VPNコネクションの作成(V2サービス/自己署名証明書利用).....	67
3.5.2.6	VPNクライアントとの接続設定(V2サービス/自己署名証明書利用).....	70
3.6	仮想サーバ作成.....	72
3.6.1	キーペア作成.....	72
3.6.2	仮想サーバ作成準備(仮想サーバイメージの取得).....	73
3.6.3	仮想サーバ作成準備(仮想サーバタイプ・フレーバーの取得).....	75
3.6.4	ポート作成.....	76
3.6.5	仮想サーバ作成(CentOS・ポート指定).....	77
3.6.6	仮想サーバ作成(CentOS・DHCP取得).....	79
3.6.7	仮想サーバ作成(Windows・ポート指定).....	81
3.6.8	仮想サーバ作成(Windows・DHCP取得).....	84
3.6.9	仮想サーバポートの取得.....	87
3.6.10	グローバルIPアドレスの取得と仮想サーバへの割り当て.....	87
3.7	接続確認.....	90
3.7.1	仮想サーバ(CentOS)へのログイン.....	90
3.7.2	仮想サーバ(Windows)へのログイン.....	90
3.8	セキュリティが高いネットワーク構成のSSL-VPN接続(V2サービス).....	91
3.8.1	システム構成図.....	91
3.8.2	ネットワーク環境の構築.....	91
3.8.2.1	管理用ネットワーク構築.....	91
3.8.2.2	業務サービス用ネットワーク構築.....	92
第4章	リソースの削除.....	96
4.1	基本構成の削除.....	97
4.1.1	削除の順番について.....	97
4.2	SSL-VPN接続削除(V2サービス).....	98
4.2.1	SSL-VPNコネクションの削除(V2サービス).....	98
4.2.2	VPNサービスの削除(V2サービス).....	98
4.2.3	SSL-VPN用鍵コンテナの削除(V2サービス).....	99
4.2.4	SSL-VPN用証明書の削除(V2サービス).....	99
4.3	ファイアーウォール削除.....	101
4.3.1	ファイアーウォールの削除.....	101

4.3.2	ファイアウォールポリシーの削除.....	101
4.3.3	ファイアウォールルールの削除.....	101
4.4	仮想サーバ削除.....	103
4.4.1	グローバルIPアドレスの削除.....	103
4.4.2	仮想サーバの削除.....	103
4.4.3	ポートの削除.....	103
4.4.4	キーペアの削除.....	104
4.5	セキュリティグループ削除.....	105
4.5.1	セキュリティグループルールの削除.....	105
4.5.2	セキュリティグループの削除.....	105
4.6	ネットワーク環境の削除.....	106
4.6.1	仮想ルータとサブネットをデタッチ.....	106
4.6.2	仮想ルータの削除.....	106
4.6.3	サブネットの削除.....	106
4.6.4	ネットワークの削除.....	107
A:	付録.....	108
A.1	証明書登録におけるpayloadの指定方法.....	108
A.2	証明書の形式変換.....	109
A.3	APIアクセス環境の設定 (Windows).....	110

第 1 章: はじめに

トピック:

- ・ [本書で使用するコマンド](#)

1.1 本書で使用するコマンド

本書に記載しているコマンドとコマンドオプションの概要を説明します。

本書に記載しているコマンドの取り扱い

本書の執筆にあたっては本サービスのAPIを利用する上で複数のOSSコマンドを使用して動作確認を実施しておりますが、本書に記載したOSSコマンド自体をサポートするものではありません。

下記のOSSコマンド説明に記載しているオプション以外を利用する場合は、ユーザーの責任において調査の上、ご利用ください。

本書に記載しているコマンド例は「bash」で実行することを想定しています。

cURL コマンド

さまざまなプロトコルに対応したデータ通信を行うOSSコマンドです。本書ではHTTPS通信によるAPI実行のために使用しています。

本書で使用しているコマンドオプション

オプション	説明
-s	通信の進捗状況やコマンド構文エラーの表示を抑止する場合に指定します。
-S	「-s」オプションを利用する際、コマンド構文エラーを表示する場合に指定します。
-i	レスポンスのHTTPヘッダーを表示する場合に指定します。
-H	リクエストに使用するHTTPヘッダーを指定します。
-X	リクエストメソッドを指定します。
-d	リクエストボディとして送信するデータを指定します。
--cert	クライアント認証で使用するクライアント証明書を指定します。
--key	クライアント認証で使用するクライアント証明書に対応する秘密鍵を指定します。

jq コマンド

コマンドラインからJSON形式のデータを操作するOSSコマンドです。本書ではレスポンスボディに含まれるJSONデータに改行などを加え、整形して表示するために使用しています。

本書ではコマンドオプションを使用していません。

Openssl コマンド

暗号化に使用する証明書の作成や形式変換など、暗号化関連の操作が行えるOSSコマンドです。本書ではクライアント認証に使用する本サービスのクライアント証明書の形式変換に使用しています。

本書で使用しているコマンドオプション

オプション	説明
x509	X.509形式の証明書を操作するために指定します。X.509形式の証明書にはPEM,DERなどの形式が含まれます。
pkcs12	PKCS#12形式の証明書を操作するために指定します。

オプション	説明
rsa	秘密鍵を操作するために指定します。
-in	操作対象の証明書を指定します。
-out	形式変換などを行う場合に出力ファイル名を指定します。
-clcerts	PKCS#12形式のクライアント証明書からPEM形式のクライアント証明書に変換する場合に指定します。
-nokeys	PKCS#12形式のクライアント証明書から秘密鍵のみ抽出する場合に指定します。
-nodes	PKCS#12形式のクライアント証明書から秘密鍵を抽出する際、パスフレーズによる暗号化を実施しない場合に指定します。
-modulus	証明書と秘密鍵の整合性確認において必要な情報を表示するために指定します。
-noout	証明書や秘密鍵の操作においてファイルを出力しない場合に指定します。

第 2 章: 環境構築の準備

トピック:

- ・ [API利用環境の構築](#)
- ・ [プロジェクト管理](#)
- ・ [グループ管理](#)
- ・ [ユーザー/グループのロール管理](#)

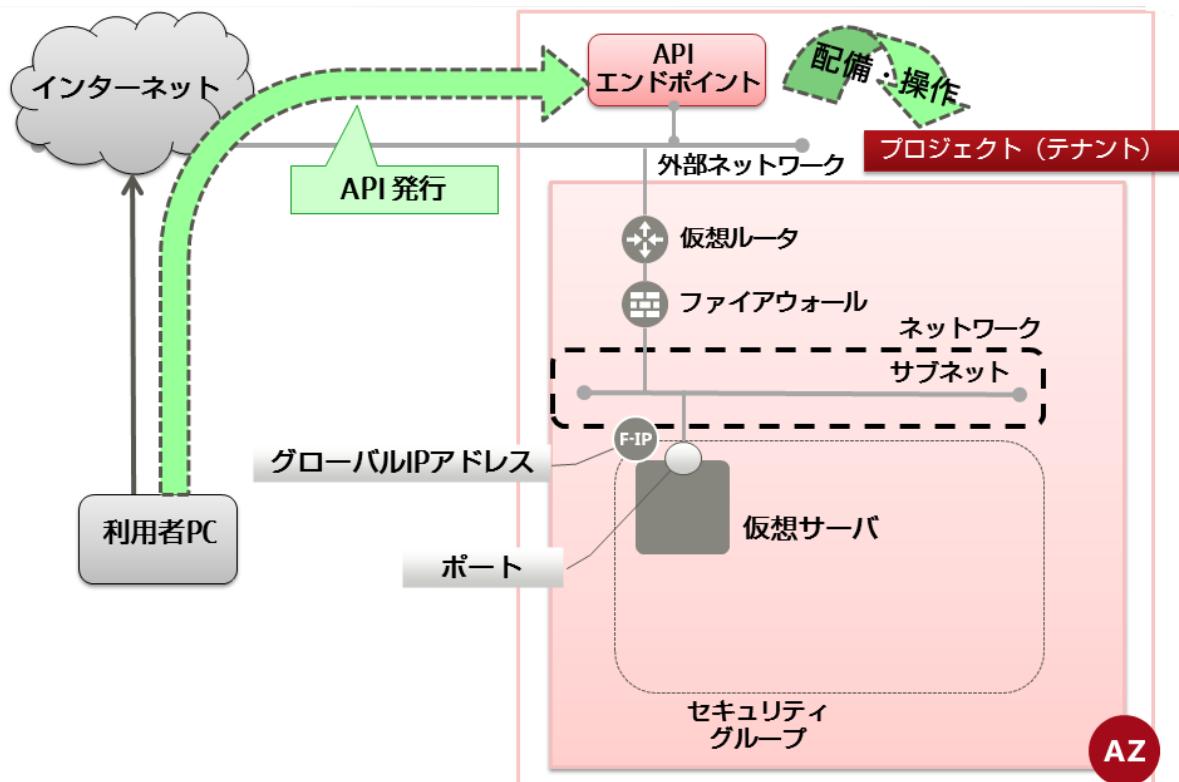
2.1 API利用環境の構築

2.1.1 環境の説明

これから構築するAPI利用環境の概要について説明します。

ユーザーPC(クライアントPC:今回、ユーザー環境はCentOS「bash」を前提として記述しています。Windows OSを使用する場合はLinux コマンドを実行するためCygwinをインストールしてご利用ください。)からインターネット経由でアクセスします。(外部エンドポイント、仮想サーバなど)

なお、Cygwinのインストールにつきましては、付録「APIアクセス環境の設定(Windows)」をご参照ください。



2.1.2 shellの準備

APIの実行をするための設定をするshellを準備します。

作業ディレクトリを作成および環境変数の設定、サービスの認証のための2つのスクリプトの作成、および格納方法について説明します。

1. 作業ディレクトリを作成します。

ユーザーのホームディレクトリ配下に作業ディレクトリを新規作成し、移動します。

```
$ mkdir <任意の作業ディレクトリ名>
```

```
$ cd <任意の作業ディレクトリ名>
```

2. init.shとinit_global.shを作成します。

現在のディレクトリにアカウント情報などを環境変数に設定するスクリプト"init.sh"と"init_global.sh"を作成します。



重要

"init.sh"はリージョナルサービス、"init_global.sh"はグローバルサービスを利用するための環境変数の設定を行います。リージョナルサービス、グローバルサービスの説明については「機能説明書」を参照してください。

init.sh

```
#!/bin/bash
# アカウント情報.
DOMAIN_NAME=<契約番号 (ドメイン)>
DOMAIN_ID=<ドメインID>
TENANT_ID=<プロジェクトID>
PROJECT_ID=$TENANT_ID
USER_NAME=<ユーザー名>
USER_PW=<パスワード>

# エンドポイントショートカット.
echo "EP初期設定"
TOKEN=https://identity.<リージョン識別子>.cloud.global.fujitsu.com
IDENTITY=$TOKEN
NETWORK=https://networking.<リージョン識別子>.cloud.global.fujitsu.com
COMPUTE=https://compute.<リージョン識別子>.cloud.global.fujitsu.com
CEILOMETER=https://telemetry.<リージョン識別子>.cloud.global.fujitsu.com
TELEMETRY=$CEILOMETER
DB=https://database.<リージョン識別子>.cloud.global.fujitsu.com
BLOCKSTORAGE=https://blockstorage.<リージョン識別子>¥
.cloud.global.fujitsu.com
HOST_BLOCKSTORAGEV2=$BLOCKSTORAGE
OBJECTSTORAGE=https://objectstorage.<リージョン識別子>¥
.cloud.global.fujitsu.com
ORCHESTRATION=https://orchestration.<リージョン識別子>¥
.cloud.global.fujitsu.com
LB=https://loadbalancing.<リージョン識別子>.cloud.global.fujitsu.com
AUTOSCALE=https://autoscale.<リージョン識別子>.cloud.global.fujitsu.com
IMAGE=https://image.<リージョン識別子>.cloud.global.fujitsu.com
MAILSERVICE=https://mail.<リージョン識別子>.cloud.global.fujitsu.com
NETWORK_EX=https://networking-ex.<リージョン識別子>.cloud.global.fujitsu.com
DNS=https://dns.gls.cloud.global.fujitsu.com
COMPUTE_SAP=https://compute-w.<リージョン識別子>.cloud.global.fujitsu.com
KEYMANAGEMENT=https://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com
SOFTWARE=https://software.<リージョン識別子>.cloud.global.fujitsu.com
VMIMPORT=https://vmimport.<リージョン識別子>.cloud.global.fujitsu.com
VMEXPORT=https://import-export.<リージョン識別子>.cloud.global.fujitsu.com

# 初期設定
NAME_FORMAT="TES_$(date "+%m%d")_$(who am I | cut -d " " -f1)_"
# その他
alias curl='curl --tlsv1.2'
SCRIPT_PATH=`pwd`
RES_DIR=response
RES_PATH=$SCRIPT_PATH/$RES_DIR
```

init_global.sh

```
#!/bin/bash
# アカウント情報.
DOMAIN_NAME=<契約番号 (ドメイン)>
DOMAIN_ID=<ドメインID>
TENANT_ID=<プロジェクトID>
PROJECT_ID=$TENANT_ID
USER_NAME=<ユーザー名>
USER_PW=<パスワード>

# エンドポイントショートカット.
echo "EP初期設定"
```

```
TOKEN=https://identity.gls.cloud.global.fujitsu.com
IDENTITY=$TOKEN
CONTRACT=https://contract.gls.cloud.global.fujitsu.com
BILL=https://billing.gls.cloud.global.fujitsu.com
DNS=https://dns.gls.cloud.global.fujitsu.com
CATALOG=https://catalog.gls.cloud.global.fujitsu.com
```

```
# 初期設定
NAME_FORMAT="TES_$(date "+%m%d")_$(who am I | cut -d " " -f1)_"
# その他
alias curl='curl --tlsv1.2'
SCRIPT_PATH=`pwd`
RES_DIR=response
RES_PATH=$SCRIPT_PATH/$RES_DIR
```



重要

- ドメインID、プロジェクト名、プロジェクトID、ユーザーIDなど本シェルで必要な情報についてはポータル「IaaS 管理」をご確認いただき、記入ください。
- リージョン識別子は「機能説明書」を参照し、自身の使用するリージョンに対応したリージョン識別子に書き換えて使用ください。
- 本ガイドでは、cURLというコマンドライン・ツールを使用しています。本ガイドはcURLのバージョン7.45.0において動作を確認しています。
- DNSサービスを利用する場合、以下の作業が必要です。
 - 「東日本リージョン1(jp-east-1)」にプロジェクトを作成し、DNSサービスを利用するユーザーをそのプロジェクトに登録してください。
 - リージョナルトークンを使用してください。
 - メール配信サービスは「東日本リージョン1(jp-east-1)」でのみ提供されます。

3. get_token.shとget_global_token.shを作成します。

サービスの認証を行うためのトークンを取得するスクリプト"get_token.sh"と"get_global_token.sh"を作成します。



重要

- "get_token.sh"はリージョナルトークン、"get_global_token.sh"はグローバルトークンを取得するためのスクリプトです。利用するサービスにより、実行するスクリプトを選択してください。
- 認証方式に「証明書+パスワード認証」を使用する場合は、cURL コマンド行に以下オプションを追加してください。

```
--cert <クライアント証明書名> --key <クライアント証明書用秘密鍵名>
```

本章に記載している"get_token.sh"と"get_global_token.sh"には「パスワード認証」「証明書+パスワード認証」の両パターンのコマンドを記載しています。必要に応じて不要な行をコメントアウトして使用してください。



注

「証明書+パスワード認証」を利用する場合、事前にポータルにて認証方式の変更と、証明書発行を実施しておく必要があります。詳細については「ポータルユーザーズガイド」を参照ください。

また、cURL コマンドで証明書を利用する場合はPEM形式への変換が必要となります。証明書の変換については[証明書の形式変換](#)を参照してください。

本手順で記載しているcURLで使用している秘密鍵は[証明書の形式変換](#)の「暗号化されていない秘密鍵」を使用することを前提としています。

```
get_token.sh
```

```
#!/bin/bash
## トークンを取得するスクリプト
. ~/<任意の作業ディレクトリ名>/init.sh
TMPFILE=~/<任意の作業ディレクトリ名>/token.txt
```

```

echo ""
echo "*****"
echo "**          トークン取得します          **"
echo "** (トークン取得後、キーを表示します) **"
echo "*****"
echo ""
echo ' ■設定内容'
echo ' endpoint':$TOKEN
echo ' domain_name':$DOMAIN_NAME
echo ' domain_id':$DOMAIN_ID
echo ' user_name':$USER_NAME
echo ' user_pw':$USER_PW
echo ' project_id':$PROJECT_ID

echo "■CURL"
echo ' curl -X POST '$TOKEN'/v3/auth/tokens -H "Content-Type:application/json"
-H "Accept:application/json" -d'
echo ' {"auth":{"identity":{"methods":["password"],"password":{"user":{"domain":
{"name":"$DOMAIN_NAME"}, "name":"$USER_NAME", "password":
"$USER_PW"}}, "scope":{"project":{"id":"$PROJECT_ID"}}}}' | jq .

echo -n "***** Hit Enter Key *****"

read

# パスワード認証を利用する場合
curl -X POST -s $TOKEN/v3/auth/tokens -H "Content-Type: application/json" -H ¥
"Accept:application/json" -d '{"auth":{"identity":{"methods":["password"],"password":
{"user":{"domain":{"name":"$DOMAIN_NAME"}, "name":"$USER_NAME", "password":
"$USER_PW"}}, "scope":{"project":{"id":"$PROJECT_ID"}}}}' | ¥
awk '/X-Subject-Token/ {print $2}' > $TMPFILE | tr -d '¥r¥n'

# 証明書+パスワード認証を利用する場合
curl -X POST -s $TOKEN/v3/auth/tokens --cert <クライアント証明書名> --key ¥
<クライアント証明書用秘密鍵名> -H "Content-Type: application/json" -H ¥
"Accept:application/json" -d '{"auth":{"identity":{"methods":["password"],"password":
{"user":{"domain":{"name":"$DOMAIN_NAME"}, "name":"$USER_NAME", "password":
"$USER_PW"}}, "scope":{"project":{"id":"$PROJECT_ID"}}}}' | ¥
awk '/X-Subject-Token/ {print $2}' > $TMPFILE | tr -d '¥r¥n'

OS_AUTH_TOKEN=`cat $TMPFILE | tr -d '¥r¥n'`

echo "=== ここから、取得された認証トークン ==="
echo $OS_AUTH_TOKEN
echo "=== ここまで、取得された認証トークン ==="

```

get_global_token.sh

```

#!/bin/bash
## トークンを取得するスクリプト
~/<任意の作業ディレクトリ名>/init_global.sh
TMPFILE=~/<任意の作業ディレクトリ名>/token.txt

echo ""
echo "*****"
echo "**          トークン取得します          **"
echo "** (トークン取得後、キーを表示します) **"
echo "*****"
echo ""
echo ' ■設定内容'
echo ' endpoint':$TOKEN
echo ' domain_name':$DOMAIN_NAME
echo ' domain_id':$DOMAIN_ID
echo ' user_name':$USER_NAME
echo ' user_pw':$USER_PW
echo ' project_id':$PROJECT_ID

```

```

echo "■CURL"
echo `curl -X POST '$TOKEN'/v3/auth/tokens -H "Content-Type:application/json"
-H "Accept:application/json" -d'
echo '{"auth":{"identity":{"methods":["password"],"password":{"user":{"domain":
{"name":"$DOMAIN_NAME"},"name":"$USER_NAME"},"password":"$USER_PW"}},
"scope":{"project":{"id":"$PROJECT_ID"}}}' | jq .

echo -n "***** Hit Enter Key *****"

read

# パスワード認証を利用する場合
curl -X POST -s '$TOKEN'/v3/auth/tokens -H "Content-Type: application/json" -H ¥
"Accept:application/json" -d '{"auth":{"identity":{"methods":["password"],"password":
{"user":{"domain":{"name":"$DOMAIN_NAME"},"name":"$USER_NAME"},"password":
"$USER_PW"}}, "scope":{"project":{"id":"$PROJECT_ID"}}}' | ¥
awk '/X-Subject-Token/ {print $2}' > $TMPFILE | tr -d '¥r¥n'

# 証明書+パスワード認証を利用する場合
curl -X POST -s '$TOKEN'/v3/auth/tokens --cert <クライアント証明書名> --key ¥
<クライアント証明書用秘密鍵名> -H "Content-Type: application/json" -H ¥
"Accept:application/json" -d '{"auth":{"identity":{"methods":["password"],"password":
{"user":{"domain":{"name":"$DOMAIN_NAME"},"name":"$USER_NAME"},"password":
"$USER_PW"}}, "scope":{"project":{"id":"$PROJECT_ID"}}}' | ¥
awk '/X-Subject-Token/ {print $2}' > $TMPFILE | tr -d '¥r¥n'

OS_AUTH_TOKEN=`cat $TMPFILE | tr -d '¥r¥n'`

echo "=== ここから、取得された認証トークン ==="
echo $OS_AUTH_TOKEN
echo "=== ここまで、取得された認証トークン ==="

```

4. 確認

作成したスクリプトを確認します。

```
$ ls
get_global_token.sh get_token.sh init.sh init_global.sh
```

2.1.3 tokenの取得

APIを利用するための認証に必要なtokenの取得方法を説明します。



作業ディレクトリの作成と2つのスクリプト作成が完了していることを前提とします。

注

1. 作業ディレクトリに移動します。

```
$ cd <任意の作業ディレクトリ名>
```

2. コマンドを実行します。

以下のコマンドを実施することで"OS_AUTH_TOKEN"変数にtokenが書き込まれます。

```
$ . ./get_token.sh
```

コンソール画面に以下のようにレスポンスが出力されます。

EP初期設定.

```

*****
** トークン取得します **
** (トークン取得後、キーを表示します) **
*****

```

■設定内容

```
endpoint:https://identity.<リージョン識別子>.cloud.global.fujitsu.com
domain_name:<契約番号 (ドメイン)>
domain_id:<ドメインID>
user_name:<ユーザー名>
user_pw:<ユーザーパスワード>
project_id:<プロジェクトID>
```

■CURL

```
curl -X POST https://identity.<リージョン識別子>.cloud.global.fujitsu.com/v3/auth/tokens
-H "Content-Type:application/json" -H "Accept:application/json" -d
{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "domain": {
            "name": "<契約番号 (ドメイン)>"
          },
          "name": "<ユーザー名>",
          "password": "<ユーザーパスワード>"
        }
      }
    },
    "scope": {
      "project": {
        "id": "<プロジェクトID>"
      }
    }
  }
}
```

```
=== ここから、取得された認証トークン ===
<任意の文字列 (半角英数字32文字)>
=== ここまで、取得された認証トークン ===
```

認証トークンが確認できれば、この環境でAPI操作が可能となっています。

2.1.4 APIを利用上の注意点

APIを利用するうえでの注意点を記載します。



API実行時、5xx系のエラーが返されましたら、少し時間をおいて再度APIを実行してください。

注

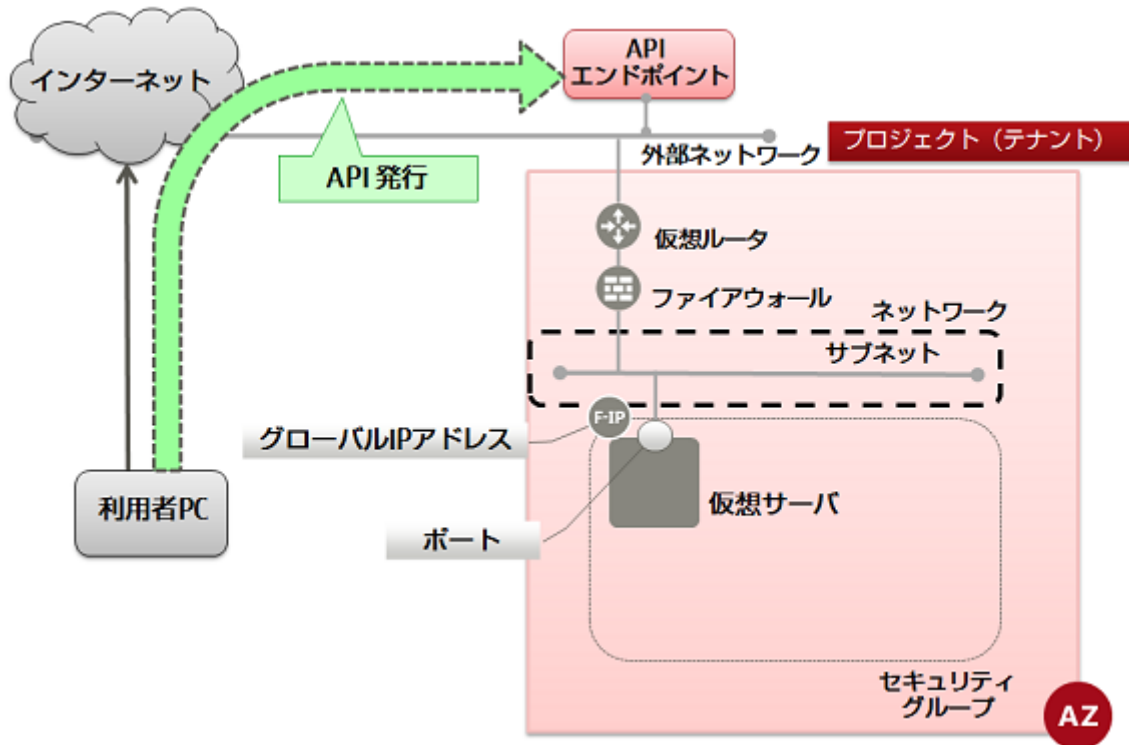
2.2 プロジェクト管理


2.2.1 プロジェクト作成

プロジェクトを作成し、作成したプロジェクトを確認するまでの手順を解説します。

本サービスでは契約内で利用する仮想リソース、グループ、およびユーザーを、プロジェクトという単位で分割して管理します。

APIの中では、プロジェクトはTENANT(テナント)と呼ばれることがあるため、注意が必要です。



 作成したプロジェクトは削除することが出来ません。

注

1. 必要な設定を行います。

```
$ TENANT_NAME=<プロジェクト名 (任意)>
```

```
$ DESCRIPTION=<プロジェクト説明 (任意)>
```

2. 次のAPIを実行します。

```
$ curl -X POST -Ss $IDENTITY/v3/projects -H "X-Auth-Token:$OS_AUTH_TOKEN" \
-H "Content-Type:application/json" -d '{"project": {"name": "$TENANT_NAME",
"description": "$DESCRIPTION", "domain_id": "$DOMAIN_ID", "enabled": true}}' | jq .
```

3. 以下のレスポンスが返ります。

```
{
  "project": {
    "description": "<プロジェクト説明>",
    "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/projects/<作成したプロジェクトID>"
    }
  },
}
```

```
    "enabled": true,
    "id": "<作成したプロジェクトID>",
    "domain_id": "<ユーザーが所属する契約組織番号>",
    "name": "<作成したプロジェクト名>"
  }
}
```

4. 作成したプロジェクトを確認するため、以下のAPIを実行します。

```
$ curl -X GET -Ss $IDENTITY/v3/projects?domain_id=$DOMAIN_ID ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

5. 以下のように、指定したプロジェクト名を含んだプロジェクトが出力されれば、作成は完了です。

```
{
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/projects",
    "previous": null,
    "next": null
  },
  "projects": [
    ...
    {
      "description": "<プロジェクト説明>",
      "links": {
        "self": "http://identity.cloud.global.fujitsu.com/v3/projects/<作成したプロジェ
クトID>"
      },
      "enabled": true,
      "id": "<作成したプロジェクトID>",
      "domain_id": "<ユーザーが所属する契約組織番号>",
      "name": "<作成したプロジェクト名>"
    },
    ...
  ]
}
```

2.3 グループ管理

2.3.1 グループ作成

複数のユーザーを含み、ロール管理に役立つグループの作成および、作成したグループを確認する手順について解説します。



注

グループはドメイン内に複数作成できますが、グループをグループに所属させることはできません。



注

本APIを実行する際はグローバルトークンを使用する必要があります。

1. 以下の項目を設定します。

```
$ TEMP_GROUP_NAME=<新規グループ名>
```

```
$ DOMAIN_ID=<ドメインID>
```

```
$ DESCRIPTION=<グループの説明>
```

2. 次のAPIを実行します。

```
$ curl -X POST -Ss $IDENTITY/v3/groups -H "X-Auth-Token:$OS_AUTH_TOKEN" \
-H "Content-Type:application/json" -d '{"group": {"description": "$DESCRIPTION",
"domain_id": "$DOMAIN_ID", "name": "$TEMP_GROUP_NAME"}}' | jq .
```

レスポンスは以下のようになります。

```
{
  "group": {
    "domain_id": "<グループが所属するドメイン名>",
    "description": "<グループの説明>",
    "id": "<新規グループのID>",
    "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/groups/<新規グループのID>"
    },
    "name": "<新規グループ名>"
  }
}
```

3. 以下のAPIでグループの一覧を取得し、作成されたグループを確認することが出来ます。取得できるのは同一ドメインのグループのみです。

```
$ curl -X GET -Ss $IDENTITY/v3/groups?domain_id=$DOMAIN_ID \
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

以下のように、設定したグループ名を含んだ一覧が返っておれば、作成は完了です。

```
{
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/groups",
    "previous": null,
    "next": null
  },
  "groups": [
    ...
  ]
}
```



```
{
  "domain_id": "<ドメインID>",
  "description": "<グループの説明>",
  "id": "<グループID>",
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/groups/<グループID>"
  },
  "name": "<グループ名>"
},
...
]
```

2.3.2 グループにユーザーを追加

ユーザーをグループに追加し、追加完了を確認する手順について解説します。

ユーザーをグループに追加することによって、グループの持つロールをユーザーに適用することが出来ます。



注

本APIを実行する際はグローバルトークンを使用する必要があります。

1. 以下で、グループに追加するユーザーと、追加対象のグループをそれぞれIDによって設定します。

```
$ TMP_USER_ID=<追加ユーザーID>
```

```
$ TMP_GROUP_ID=<追加対象のグループID>
```

2. 次のAPIを実行します。

```
$ curl -i -X PUT -s $IDENTITY/v3/groups/$TMP_GROUP_ID/users/$TMP_USER_ID ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

レスポンスは以下のようになります。

```
HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request: EXECUTED_REQ<任意の9つの数字>_204
Vary: X-Auth-Token
Date: Tue, 17 Nov 2015 07:00:02 GMT
```

3. グループにユーザーが追加されたかを確認するため、グループを指定してグループに含まれるユーザー一覧を表示させます。

```
$ TMP_GROUP_ID=<確認対象のグループID>
```

```
$ curl -X GET -s $IDENTITY/v3/groups/$TMP_GROUP_ID/users -H "X-Auth-Token:
$OS_AUTH_TOKEN" ¥
-H "Content-Type:application/json" | jq .
```

以下のように、追加したユーザーを含んだ一覧が返っておれば、作成は完了です。

```
{
  "users": [
    ...
    {
      "domain_id": "<ドメインID>",
      "name": "<ユーザー名>",
      "links": {
        "self": "http://identity.cloud.global.fujitsu.com/v3/users/<ユーザーID>"
      },
      "locale": "ja",
    }
  ]
}
```

```

    "enabled": true,
    "id": "<ユーザーID>",
    "default_project_id": "<デフォルトプロジェクトID>",
    "description": "<ユーザーの説明>"
  },
  ...
],
"links": {
  "self": "http://identity.cloud.global.fujitsu.com/v3/groups/<グループID>/users",
  "next": null,
  "previous": null
}
}

```

4. また、ユーザーをグループから削除することも可能です。その手順は以下になります。
以下のように設定します。

```
$ TMP_USER_ID=<削除ユーザーID>
```

```
$ TMP_GROUP_ID=<ユーザー削除対象のグループID>
```

5. 次のAPIを実行します。

```
$ curl -i -X DELETE -s $IDENTITY/v3/groups/$TMP_GROUP_ID/users/$TMP_USER_ID ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

レスポンスは以下のようになります。

```

HTTP/1.1 204 No Content
X-Fox-Endpoint-Request: EXECUTED_REQ<任意の9つの数字>_204
Vary: X-Auth-Token
Date: Tue, 17 Nov 2015 07:00:02 GMT

```

削除完了の確認方法は追加の場合の確認方法と同じです。

2.3.3 グループ削除

グループを削除し、その削除が完了したことを確認するまでの手順を解説します。



本APIを実行する際はグローバルトークンを使用する必要があります。

注

1. 削除したいグループをグループIDによって指定します。

```
$ TEMP_GROUP_ID=<削除したいグループID>
```

2. 以下のAPIを実行します。

```
$ curl -i -X DELETE -s $IDENTITY/v3/groups/$TEMP_GROUP_ID ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

次のレスポンスがかえってきます。

```

HTTP/1.1 204 No Content
X-Fox-Endpoint-Request: EXECUTED_REQ<任意の9つの数字>_204
Vary: X-Auth-Token
Date: Tue, 17 Nov 2015 07:00:02 GMT

```

3. ユーザーの削除が成功したかどうか確認するために、ユーザー一覧を取得するAPIは次のようになります。

```
$ curl -X GET -s $IDENTITY/v3/groups?domain_id=$DOMAIN_ID ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

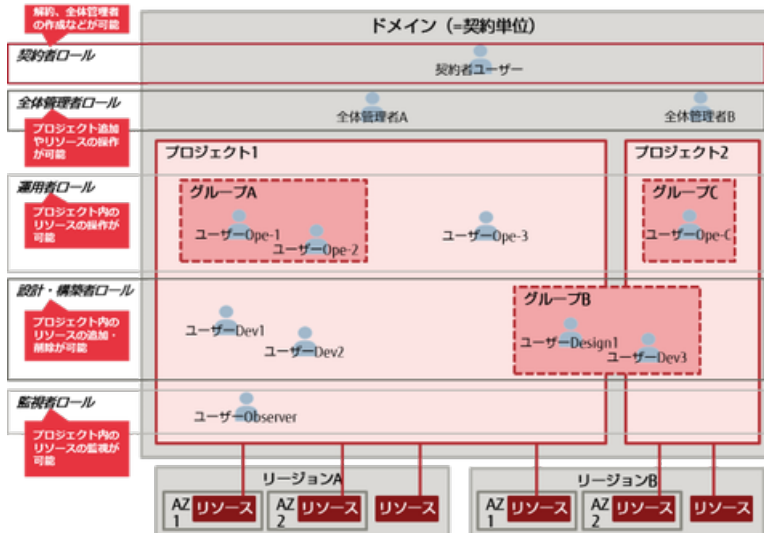
以下のようなフォーマットで同ドメインに含まれるグループが一覧で表示されますが、そこに削除対象のグループがなければ削除成功となります。

```
{
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/groups",
    "previous": null,
    "next": null
  },
  "groups": [
    ...
    {
      "domain_id": "<ドメインID>",
      "description": "<グループの説明>",
      "id": "<グループID>",
      "links": {
        "self": "http://identity.cloud.global.fujitsu.com/v3/groups/<グループID>"
      },
      "name": "<グループ名>"
    },
    ...
  ]
}
```

2.4 ユーザー/グループのロール管理

2.4.1 プリセットロール一覧の確認

ユーザーの操作権限を決定するプリセットロールの一覧する手順を示します。



次のAPIを実行します。

```
$ curl -X GET -Ss $IDENTITY/v3/roles?domain_id=$DOMAIN_ID ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | ¥  
jq '.roles[] |select(.policy_role != true)'
```

以下のレスポンスが得られます。各プリセットロールの権限について詳しくは機能説明書を参照してください。

```
{  
  "id": "0739580a550d4a0f9c78f45a9f038c05",  
  "links": {  
    "self": "http://identity.cloud.global.fujitsu.com/v3/  
roles/0739580a550d4a0f9c78f45a9f038c05"  
  },  
  "name": "cpf_systemowner"  
}  
{  
  "id": "20e572b88c544606863548f8abd4ff53",  
  "links": {  
    "self": "http://identity.cloud.global.fujitsu.com/v3/  
roles/20e572b88c544606863548f8abd4ff53"  
  },  
  "name": "cpf_operator"  
}  
{  
  "id": "3af119c426a742999e7890f6d1f70b36",  
  "links": {  
    "self": "http://identity.cloud.global.fujitsu.com/v3/  
roles/3af119c426a742999e7890f6d1f70b36"  
  },  
  "name": "cpf_admin"  
}  
{  
  "id": "970ea0105b514c16828626fe4dd50960",  
  "links": {  
    "self": "http://identity.cloud.global.fujitsu.com/v3/  
roles/970ea0105b514c16828626fe4dd50960"  
  }  
}
```

```

    },
    "name": "cpf_observer"
  }
  {
    "id": "9fe2ff9ee4384b1894a90878d3e92bab",
    "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/roles/9fe2ff9ee4384b1894a90878d3e92bab"
    },
    "name": "_member_"
  }
  {
    "id": "df7d043a09d34a7c9e2bad15926ee097",
    "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/roles/df7d043a09d34a7c9e2bad15926ee097"
    },
    "name": "cpf_org_manager"
  }
}

```

2.4.2 ロールの付与および参照

ユーザーやグループにロールを付与し、それを参照する手順を解説します。

ユーザーまたはグループをプロジェクトに参加させるには、ロールを選択して付与します。

1. 以下で、必要な設定を行います。

```
$ TMP_PROJECT_ID=<ユーザーやグループを参加させたいプロジェクトID>
```

```
$ TMP_USER_ID=<プロジェクトに参加させたいユーザー>
```

もしくは

```
$ TMP_GROUP_ID=<プロジェクトに参加させたいグループ>
```

```
$ TMP_ROLE_ID=<ロールID>
```

2. 次のAPIを実行します。

```
$ curl -i -X PUT -Ss $IDENTITY/v3/projects/$TMP_PROJECT_ID/users/$TMP_USER_ID/roles/$TMP_ROLE_ID -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

もしくは

```
$ curl -i -X PUT -Ss $IDENTITY/v3/projects/$TMP_PROJECT_ID/groups/$TMP_GROUP_ID/roles/$TMP_ROLE_ID -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

以下のレスポンスが返ります。

```

HTTP/1.1 204 No Content
Vary: X-Auth-Token
Content-Length: 0
Date: Www, DD MMM yyyy hh:mm:ss GMT

```

3. ユーザーやグループのロールを確認します。

確認の方法は以下の4パターンあります。

- a. プロジェクトグループ
- b. プロジェクトユーザー
- c. ドメイングループ
- d. ドメインユーザー

以下の設定をします。

1.

```
$ TMP_PROJECT_ID="<プロジェクトID>"
```

2.

```
$ TMP_DOMAIN_ID="<ドメインID>"
```

3.

```
$ TMP_USER_ID="<ユーザーID>"
```

4.

```
$ TMP_GROUP_ID="<グループID>"
```

4. 次のAPIを実行します。

- ```
$ curl -X GET -Ss $IDENTITY/v3/projects/$TMP_PROJECT_ID/groups/$TMP_GROUP_ID/roles ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```
- ```
$ curl -X GET -Ss $IDENTITY/v3/projects/$TMP_PROJECT_ID/users/$TMP_USER_ID/roles ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```
- ```
$ curl -X GET -Ss $IDENTITY/v3/domains/$DOMAIN_ID/groups/$TMP_GROUP_ID/roles ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```
- ```
$ curl -X GET -Ss $IDENTITY/v3/domains/$DOMAIN_ID/users/$TMP_USER_ID/roles ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

第 3 章: リソースの作成

トピック:

- ・ 想定システム構成
- ・ ネットワーク環境の構築
- ・ セキュリティグループ設定
- ・ ファイアウォール作成
- ・ 簡易構成のSSL-VPN接続 (V2サービス)
- ・ 仮想サーバ作成
- ・ 接続確認
- ・ セキュリティが高いネットワーク構成のSSL-VPN接続 (V2サービス)

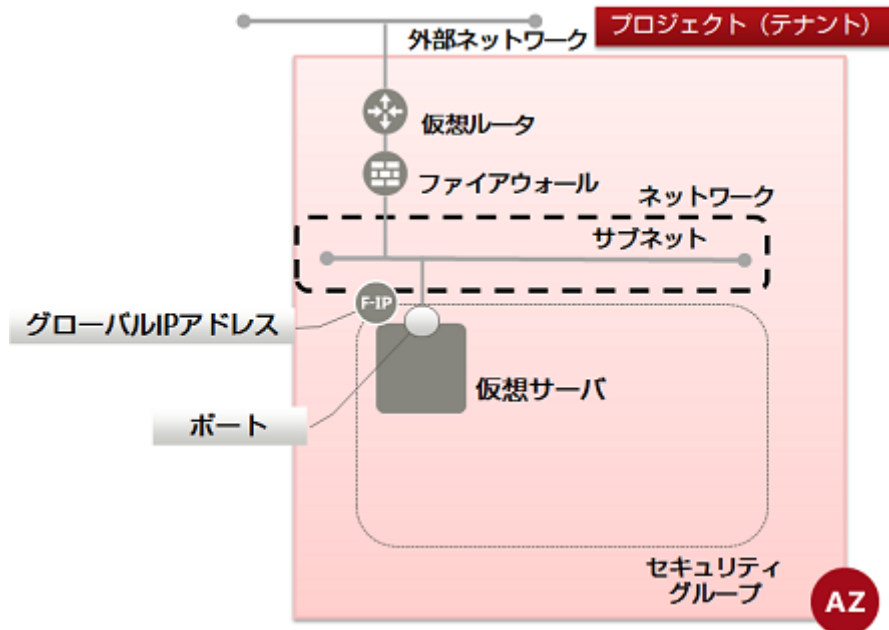
3.1 想定システム構成

3.1.1 システム構成図

本章で作成可能な構成について解説します。

仮想サーバをインターネットに公開する場合

IaaS上でインターネットに公開するシステムの基本的な構築手順を記載します。



本構成を構築するには以下の流れで作業を実施します。

1. ネットワーク作成
2. サブネット作成
3. 仮想ルータ作成
4. 仮想ルータの情報変更(外部ネットワークにアタッチ)
5. 仮想ルータの情報変更(サブネットにアタッチ)
6. セキュリティグループ作成
7. セキュリティグループルール作成
8. ファイアウォールルール作成
 - ファイアウォールルール(IPアドレス指定)作成
 - ファイアウォールルール(ポート番号指定)作成
 - ファイアウォールルール(ICMP許可)作成
 - ファイアウォールルール(拒否ルール)作成
9. ファイアウォールポリシー作成
10. ファイアウォール作成
11. キーペア作成
12. ポート作成 (仮想サーバをポート指定で作成する場合に実施してください。)
13. 仮想サーバ作成
 - 仮想サーバ作成 (CentOS・ポート指定)
 - 仮想サーバ作成 (CentOS・DHCP取得)

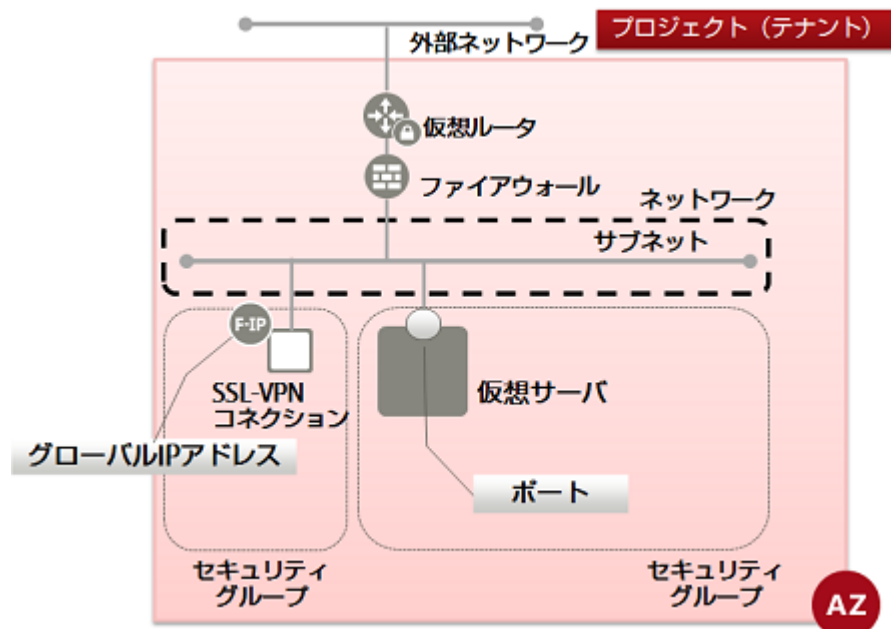
- 仮想サーバ作成 (Windows・ポート指定)
- 仮想サーバ作成 (Windows・DHCP取得)

14. グローバルIPアドレスの取得と仮想サーバへの割り当て

SSL-VPN接続機能を利用する場合

IaaS 上でSSL-VPN接続機能を利用して接続するシステムの基本的な構築手順を記載します。

- V2サービス方式



SSL-VPN接続を仮想インスタンスにて実現する方式です。機能の詳細については「機能説明書」を参照してください。

なお、V2サービスのSSL-VPN接続機能ではユーザー認証の方式として「クライアント証明書利用」「自己署名証明書利用」の2つの認証方式を備えており、本書ではそれぞれの設定手順を説明しています。



注

「クライアント証明書」とは、SSL-VPNサービスを利用する際にユーザーがクライアント認証を行うために、ポータルで発行するクライアント証明書です。

IaaS 上で本構成を構築するには以下の流れで作業を実施します。

1. ネットワーク作成
2. サブネット作成
3. 仮想ルータ作成
4. 仮想ルータの情報変更(外部ネットワークにアタッチ)
5. 仮想ルータの情報変更(サブネットにアタッチ)
6. セキュリティグループ作成
7. セキュリティグループルール作成
8. ファイアウォールルール作成
 - ファイアウォールルール(IPアドレス指定)作成
 - ファイアウォールルール(ポート番号指定)作成
 - ファイアウォールルール(ICMP許可)作成
 - ファイアウォールルール(拒否ルール)作成
9. ファイアウォールポリシー作成
10. ファイアウォール作成
11. SSL-VPN接続

- SSL-VPN接続(V2サービス/クライアント証明書利用)
- SSL-VPN接続(V2サービス/自己署名証明書利用)

12. キーペア作成

13. ポート作成 (仮想サーバをポート指定で作成する場合に実施してください。)

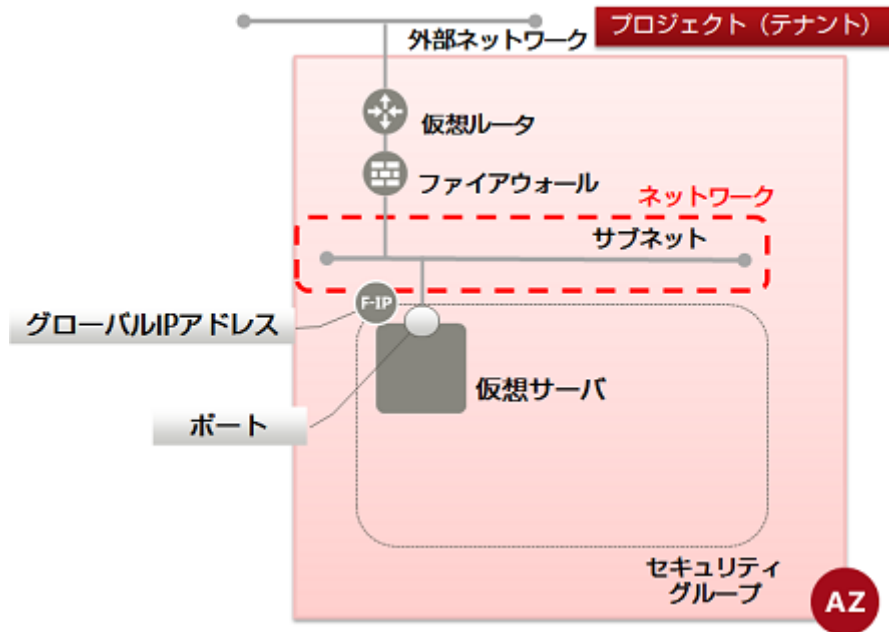
14. 仮想サーバ作成

- 仮想サーバ作成 (CentOS・ポート指定)
- 仮想サーバ作成 (CentOS・DHCP取得)
- 仮想サーバ作成 (Windows・ポート指定)
- 仮想サーバ作成 (Windows・DHCP取得)

3.2 ネットワーク環境の構築

3.2.1 ネットワーク作成

仮想サーバなどのリソースを配備するため、プロジェクト内にネットワークを作成し、確認するまでの手順を解説します。
ネットワークはプロジェクト内に複数作成できます。



1. 以下に必要な設定を行います。

```
$ NW_NAME=<新規ネットワーク名 (任意)>
```

```
$ AZ=<作成先アベイラビリティゾーン名>
```

2. 次のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/networks -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"network":{"name": "$NW_NAME",
"availability_zone": "$AZ"}}' | jq .
```

以下のレスポンスが返ってきます。

```
{
  "network": {
    "status": "ACTIVE",
    "subnets": [],
    "name": "<設定したネットワーク名>",
    "admin_state_up": true,
    "tenant_id": "<プロジェクトID>",
    "shared": false,
    "id": "<新規ネットワークのID>",
    "availability_zone": "<指定したアベイラビリティゾーン>"
  }
}
```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

3. 作成したネットワークを確認するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/networks -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

以下のように、設定したネットワーク名を含んだリストが出力された場合、作成は完了です。

ネットワーク名が、"inf_az1_ext_net01"および"inf_az2_ext_net01"と表記されているものはデフォルトで存在する外部ネットワークです。

```
{
  "networks": [
    {
      "status": "ACTIVE",
      "subnets": [
        "5079f324-5db0-44ee-92ac-3a6b7977b23f",
        "a56b6058-0479-43a1-8b27-01c1c05e96a2",
        "c1da3ee7-51c3-4801-bb97-aa03a4383ef0",
        "e96e55b8-84bb-4777-a782-a5d6e8340039",
        "f5e9ec37-88ec-494b-ac55-dae101a54cc1"
      ],
      "name": "inf_az2_ext-net01",
      "router:external": true,
      "tenant_id": "31ceb599e8ff48aeb66f2fd748988960",
      "admin_state_up": true,
      "shared": true,
      "id": "375c49fa-a706-4676-b55b-2d3554e5db6a",
      "availability_zone": "jp-east-1b"
    },
    ...
    {
      "status": "ACTIVE",
      "subnets": [],
      "name": "<ネットワーク名>",
      "router:external": false,
      "tenant_id": "<プロジェクトID>",
      "admin_state_up": true,
      "shared": false,
      "id": "<ネットワークID>",
      "availability_zone": "<アベイラビリティゾーン>"
    },
    {
      "status": "ACTIVE",
      "subnets": [
        "37ca5225-6ca6-4ee6-a49d-d479400a632b",
        "3c2419a7-7745-452c-bee5-664db03129bb",
        "7c7feecd-082c-415b-865a-82d4e5de97e5",
        "d96cdd6c-6e1e-4331-9c8e-42b52588b767",
        "ff4a9bd6-37be-4c03-9b87-1f693f807b48"
      ],
      "name": "inf_az1_ext-net01",
      "router:external": true,
      "tenant_id": "31ceb599e8ff48aeb66f2fd748988960",
      "admin_state_up": true,
      "shared": true,
      "id": "af4198a9-b392-493d-80ec-a7c6e5a1c22a",
      "availability_zone": "jp-east-1a"
    }
  ]
}
```

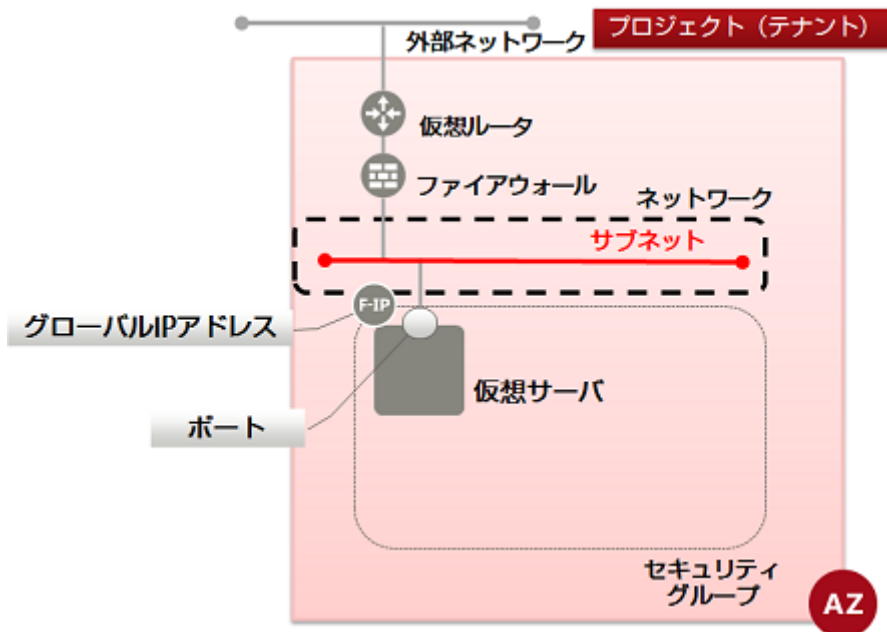
仮想サーバなどのリソースを配備するには、作成したネットワークに対して引き続きサブネットを作成していきます。

3.2.2 サブネット作成

3.2.2.1 サブネット作成

ネットワークに接続するリソースに対するプライベートIPアドレスの管理、DHCPによるIPアドレスの自動設定に必要です。ここではサブネットを作成し、確認するまでの手順を解説します。

ネットワークを指定して、サブネットは作成します。



1. 以下で、必要な設定を行います。



注

ここでDNSサーバを設定しますが、仮想サーバがDNSサーバと通信するためには、インターネットへのアウトバウンド通信を許可する必要があります。セキュリティグループ機能、またはファイアウォールサービスの設定内容を確認し、DNSサーバへの通信(プロトコル: TCP/UDP、ポート番号: 53)を許可するようにしてください。

なお、DNSに指定する値は環境により異なります。本サービスで提供しているDNSを利用する場合は、「機能説明書」の付録「共通ネットワークサービス」を参照してください。

```
$ SUBNET=<新規サブネット名 (任意)>
```

```
$ NETWORK_ID=<指定するネットワークID>
```

```
$ CIDR=<ネットワークアドレス (XXX.XXX.XXX.0/24 (任意))>
```

```
$ GATEWAY_IP=<デフォルトゲートウェイIPアドレス (XXX.XXX.XXX.1 (任意))>
```

```
$ DNS=<¥"133.162.XXX.XXX¥"¥, ¥"133.162.XXX.XXX¥">
```



注

CIDRパラメーターの指定可能なアドレスの範囲については、「機能説明書」の「サブネット管理」を参照して下さい。

GATEWAY_IPパラメーターには、CIDRパラメーターで指定したアドレスレンジ内での任意のIPアドレスを指定して下さい。



警告

作成したサブネットに対し、SSL-VPN接続 (V2サービス) で接続する場合は、上記の変数を以下のように指定する必要があります。

- CIDR
ネットワークアドレスのマスク値を「16bit～29bit」の範囲内で指定する。
例) 192.168.1.0/24
- GATEWAY_IP
VPNサービスで指定する仮想ルータのIPアドレスを指定する。

2. 次のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/subnets -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"subnet": {"name": "$SUBNET",
"network_id": "$NETWORK_ID", "cidr": "$CIDR", "dns_nameservers": ["$DNS"],
"ip_version": 4, "gateway_ip": "$GATEWAY_IP", "availability_zone": "$AZ"}}' | jq .
```

以下のレスポンスが返ります。

```
{
  "subnet": {
    "name": "新規サブネット名",
    "enable_dhcp": true,
    "network_id": "<設定したネットワークID>",
    "tenant_id": "<プロジェクトID>",
    "dns_nameservers": [
      "133.162.XXX.XXX",
      "133.162.XXX.XXX"
    ],
    "allocation_pools": [
      {
        "start": "XXX.XXX.XXX.2",
        "end": "XXX.XXX.XXX.254"
      }
    ],
    "host_routes": [],
    "ip_version": 4,
    "gateway_ip": "<設定したデフォルトゲートウェイIPアドレス>",
    "cidr": "設定したネットワークアドレス",
    "id": "新規サブネットID",
    "availability_zone": "指定したネットワークと同じアベイラビリティゾーン"
  }
}
```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

3. 作成したサブネットを確認するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/subnets -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

以下のようなフォーマットでレスポンスが返ってきます。設定したサブネット名を含んだリストが出力された場合、作成は完了です。

サブネット名が、"inf_az1_fip-pool"および"inf_az2_fip-pool"、"inf_az1_ext-subnet"、"inf_az1_ext-subnet"から始まるサブネットはデフォルトで存在する外部ネットワークに紐づいて作られたサブネットです。

```
{
  "subnets": [
    ...
    {
      "name": "サブネット名",
      "enable_dhcp": true,
      "network_id": "<ネットワークID>",
      "tenant_id": "<プロジェクトID>",

```

```

    "dns_nameservers": [
      "133.162.XXX.XXX",
      "133.162.XXX.XXX"
    ],
    "allocation_pools": [
      {
        "start": "XXX.XXX.XXX.2",
        "end": "XXX.XXX.XXX.254"
      }
    ],
    "host_routes": [],
    "ip_version": 4,
    "gateway_ip": "<デフォルトゲートウェイIPアドレス>",
    "cidr": "ネットワークアドレス",
    "id": "サブネットID",
    "availability_zone": "ネットワークと同じアベイラビリティゾーン"
  },
  ...
]
}

```

3.2.2.2 ルーティングの追加

作成したサブネットに必要なルーティング情報を設定します。この設定は[管理用ネットワーク構築](#)にて必要な設定です。以下に、SSL-VPN接続時に必要となるルーティング設定を例に手順を示します。

1. 以下で、ルーティングに必要な設定を行います。

```
$ SUBNET_ID=<ルーティング (host_routes) を追加するサブネットID>
```

```
$ HOST_ROUTES="{¥nextHop¥":¥サブネットのデフォルトゲートウェイIPアドレス¥",
¥destination¥":¥VPNトンネルの仮想ネットワークcidr¥}"
```

VPNトンネルの仮想ネットワークcidrはXXX.XXX.XXX.XXX/XX形式でネットワークアドレスの指定を行ってください。

例) 192.168.246.0/24



警告

本サービスで使用しているネットワークアドレスやクライアントPCが接続しているローカルネットワークアドレスと競合しないネットワークアドレスを指定してください。

なお、VPNトンネルの仮想ネットワークcidrは、SSL-VPNコネクション作成時([SSL-VPNコネクションの作成 \(V2サービス/クライアント証明書利用\)](#))またはSSL-VPNコネクションの作成([V2サービス/自己署名証明書利用](#))に設定する、CLIENT_ADDRESS_POOL_CIDRと同じである必要があります。

2. 次のAPIを実行します。

```
$ curl -sS $NETWORK/v2.0/subnets/$SUBNET_ID -X PUT -H "X-Auth-Token: $OS_AUTH_TOKEN" ¥
-H "Content-Type: application/json" -d '{"subnet": {"host_routes": ["$HOST_ROUTES"]}}'
| jq .
```

以下のレスポンスが返ります。

```

{
  "subnet": {
    "availability_zone": "<指定したネットワークと同じアベイラビリティゾーン>",
    "id": "<サブネットID>",
    "cidr": "<指定したVPNトンネルの仮想ネットワークcidr>",
    "gateway_ip": "<指定したサブネットのデフォルトゲートウェイIPアドレス>",
    "name": "<サブネット名>",
    "enable_dhcp": true,
    "network_id": "<ネットワークID>",
    "tenant_id": "<プロジェクトID>",
    "dns_nameservers": [

```

```

    "133.162.XXX.XXX",
    "133.162.XXX.XXX"
  ],
  "allocation_pools": [
    {
      "end": "XXX.XXX.XXX.254",
      "start": "XXX.XXX.XXX.2"
    }
  ],
  "host_routes": [
    {
      "destination": "<指定したVPNトンネルの仮想ネットワークcidr>",
      "nextthop": "<指定したサブネットのデフォルトゲートウェイIPアドレス>"
    }
  ],
  "ip_version": 4
}
}

```

3. 設定したルーティングを確認するため、以下のAPIを実行します。

```
$ SUBNET_ID=<ルーティング(host_routes)を追加したサブネットID>
```

```
$ curl -Ss $NETWORK/v2.0/subnets/$SUBNET_ID -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

以下のレスポンスが返ります。設定したルーティングを含んだリストが出力された場合、作成は完了です。

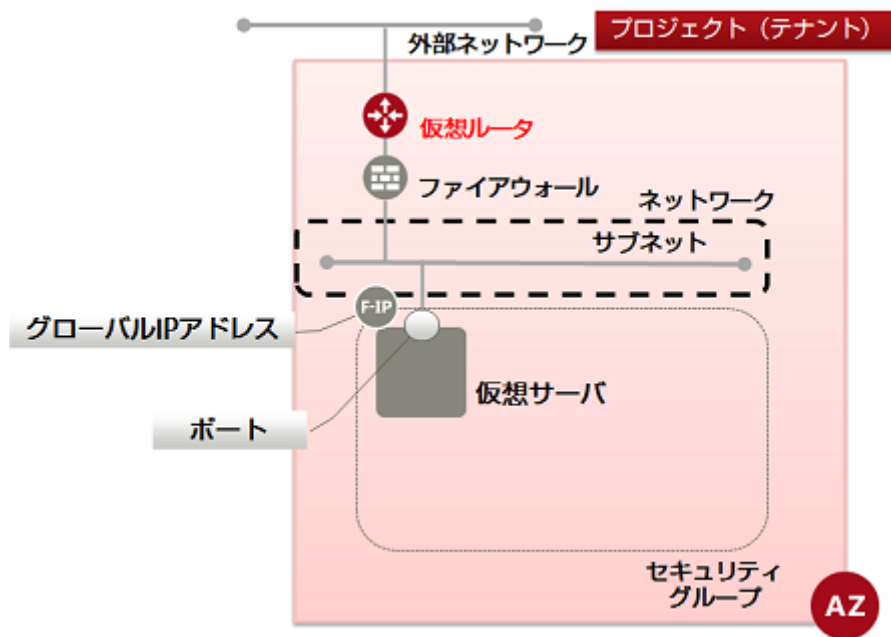
```

{
  "subnet": {
    "availability_zone": "<指定したネットワークと同じアベイラビリティゾーン>",
    "id": "<サブネットID>",
    "cidr": "<指定したVPNトンネルの仮想ネットワークcidr>",
    "gateway_ip": "<指定したサブネットのデフォルトゲートウェイIPアドレス>",
    "name": "<サブネット名>",
    "enable_dhcp": true,
    "network_id": "<ネットワークID>",
    "tenant_id": "<プロジェクトID>",
    "dns_nameservers": [
      "133.162.XXX.XXX",
      "133.162.XXX.XXX"
    ],
    "allocation_pools": [
      {
        "end": "XXX.XXX.XXX.254",
        "start": "XXX.XXX.XXX.2"
      }
    ],
    "host_routes": [
      {
        "destination": "<指定したVPNトンネルの仮想ネットワークcidr>",
        "nextthop": "<指定したサブネットのデフォルトゲートウェイIPアドレス>"
      }
    ],
    "ip_version": 4
  }
}

```

3.2.3 仮想ルータ作成

外部ネットワークとネットワーク、またはネットワーク同士を接続するための仮想ルータを作成し、確認するまでの手順を解説します。



1. 以下で、必要な設定を行います。

```
$ ROUTER_NAME=<新規仮想ルータ名 (任意)>
```

```
$ AZ=<作成先アベイラビリティゾーン名>
```

2. APIを実行します。

```
$ curl -Ss $NETWORK/v2.0/routers -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"router": {"name": "'$ROUTER_NAME'",
"tenant_id": "'$TENANT_ID'", "availability_zone": "'$AZ'"}}' | jq .
```

以下のレスポンスが返ってきます。

```
{
  "router": {
    "status": "ACTIVE",
    "external_gateway_info": null,
    "name": "<新規仮想ルータ名>",
    "admin_state_up": true,
    "tenant_id": "<プロジェクトID>",
    "id": "新規仮想ルータID",
    "availability_zone": "<指定したアベイラビリティゾーン>"
  }
}
```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

3. 作成した仮想ルータを確認するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/routers -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

以下のように、設定した仮想ルータ名を含んだリストが出力された場合、作成は完了です。

```
{
  "routers": [
    ...
    {
      "status": "ACTIVE",
      "external_gateway_info": null,
      "name": "<新規仮想ルータ名>",
      "admin_state_up": true,
      "tenant_id": "<プロジェクトID>",
    }
  ]
}
```

```

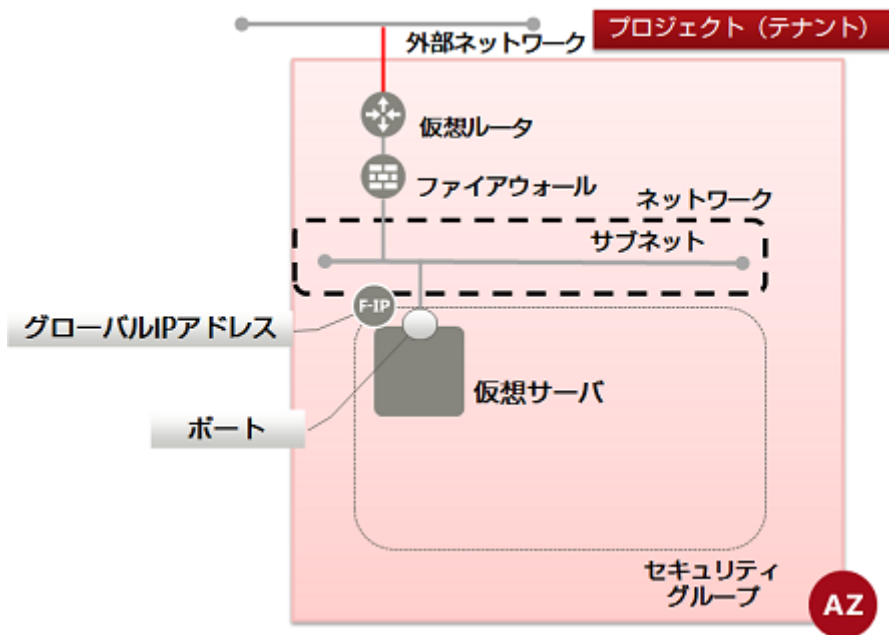
    "id": "新規仮想ルータID",
    "availability_zone": "<指定したアベイラビリティゾーン>"
  },
  ...
]
}

```

仮想ルータの情報変更機能を用いてサブネットを追加する手順に関しては、仮想ルータの情報変更(アタッチ)を参照してください。

3.2.4 仮想ルータの情報変更(外部ネットワークにアタッチ)

既存の仮想ルータに対し、設定情報を更新することによって仮想ルータを外部ネットワークに接続する手順を解説します。



1. 以下で必要な設定を行います。

```
$ ROUTER_ID=<対象の仮想ルータID>
```

```
$ EXT_NET_ID=<接続する外部ネットワークID>
```

外部ネットワークIDは[ネットワーク作成](#)の手順3で実行しているAPIで確認ができます。

```
$ curl -Ss $NETWORK/v2.0/networks -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

ネットワーク名が、"inf_az1_ext_net01"および"inf_az2_ext_net01"と表記されているものはデフォルトで存在する外部ネットワークです。

2. 次のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/routers/$ROUTER_ID -X PUT -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"router": {"external_gateway_info": \
{ "network_id": "'$EXT_NET_ID'"}}}' | jq .
```

以下のレスポンスが返ってきます。

```

{
  "router": {
    "status": "ACTIVE",
    "external_gateway_info": {

```

```

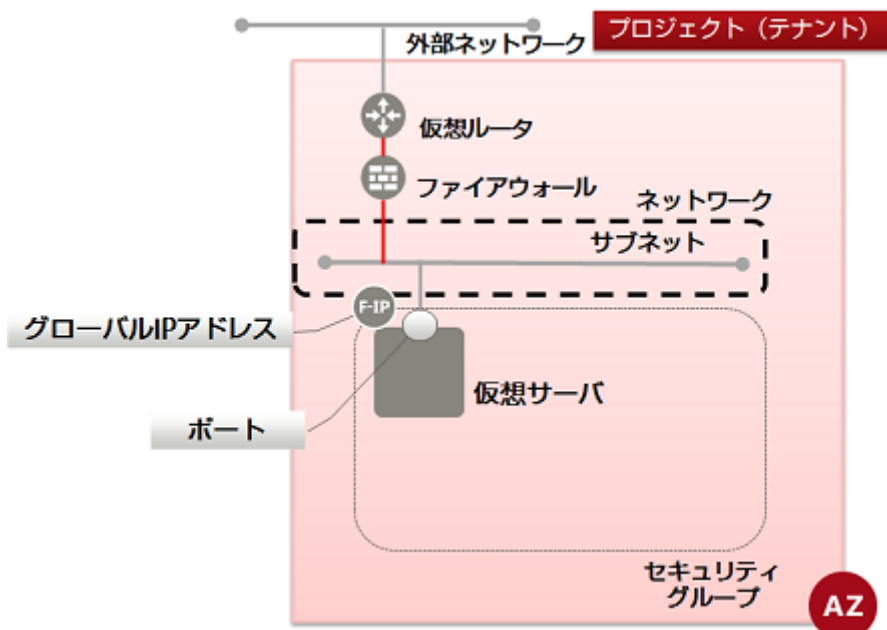
    "network_id": "<指定した外部ネットワークID>",
    "enable_snat": true
  },
  "name": "指定した仮想ルータ名",
  "admin_state_up": true,
  "tenant_id": "プロジェクトID",
  "routes": [],
  "id": "指定した仮想ルータのID",
  "availability_zone": "<アベイラビリティゾーン>"
}
}

```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

3.2.5 仮想ルータの情報変更(サブネットにアタッチ)

既存の仮想ルータに対し、設定情報を更新することによって仮想ルータをIaaS上の既存のサブネットに接続する手順を解説します。



1. 以下で必要な設定を行います。

```
$ ROUTER_ID=<対象の仮想ルータID>
```

```
$ SUBNET_ID=<接続するサブネットID>
```

2. 次のAPIを実行します。

```
$ curl -sS $NETWORK/v2.0/routers/$ROUTER_ID/add_router_interface -X PUT \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" \
-d '{"subnet_id": "$SUBNET_ID"}' | jq .
```

以下のレスポンスが返ってきます。

```

{
  "subnet_id": "<指定したサブネットID>",
  "tenant_id": "<プロジェクトID>",
  "port_id": "<ポートID>",
  "id": "<仮想ルータのID>",
  "availability_zone": "<アベイラビリティゾーン>"
}

```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

3.3 セキュリティグループ設定

3.3.1 セキュリティグループ作成

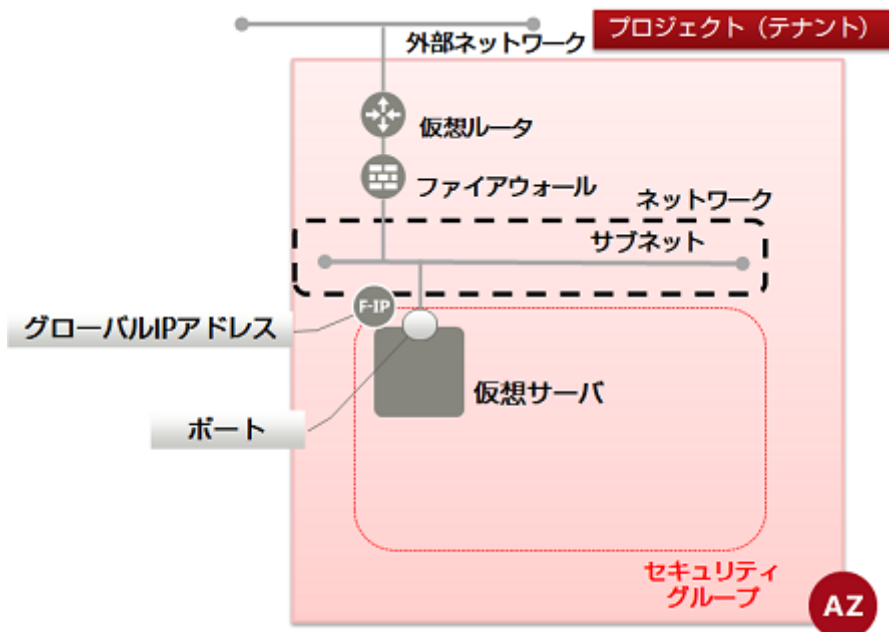
セキュリティグループを用いると、仮想サーバに接続されたポートに対してパケットフィルタリングを行うため、ルール設定をグループ化して定義および設定することができます。ここでは、セキュリティグループを作成し、確認するまでの手順を解説します。

セキュリティグループには複数のルールを設定できます。1つのポートに設定されたセキュリティグループのうち、どれか1つでもルールにマッチしたパケットは通信が許可され、それ以外の通信は遮断されます。(ホワइटリスト方式、OR条件)



注

セキュリティグループは、仮想ルータおよびDHCPサーバが持つポートには設定できません。



ポートには自動的に通信を遮断するデフォルトセキュリティグループが設定されています。そこで、新たにセキュリティグループを作成し、必要に応じて通信を許可するルールを設定します。

1. 以下で、必要な設定を行います。

セキュリティグループはアベイラビリティゾーンに紐づかないため、指定は不要です。

```
$ SG_NAME=<新規セキュリティグループ名 (任意)>
```

2. 次のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/security-groups -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"security_group": {"name": "'$SG_NAME'"}}' | jq .
```

以下のレスポンスが返ります。

```
{
  "security_group": {
    "tenant_id": "<プロジェクトID>"

    "description": "",
    "name": "<新規セキュリティグループ名>",
```

```

"security_group_rules": [
  {
    "remote_group_id": null,
    "direction": "egress",
    "remote_ip_prefix": null,
    "protocol": null,
    "ethertype": "IPv6",
    "port_range_max": null,
    "security_group_id": "<新規セキュリティグループID>",
    "port_range_min": null,
    "tenant_id": "<プロジェクトID>",
    "id": "<デフォルトのセキュリティグループルールID>"
  },
  {
    "remote_group_id": null,
    "direction": "egress",
    "remote_ip_prefix": null,
    "protocol": null,
    "ethertype": "IPv4",
    "port_range_max": null,
    "security_group_id": "<新規セキュリティグループID>",
    "port_range_min": null,
    "tenant_id": "<プロジェクトID>",
    "id": "<デフォルトのセキュリティグループルールID>"
  }
],
"id": "<新規セキュリティグループID>"
}

```

- 作成したセキュリティグループを確認するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/security-groups -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.security_groups[] | .name, .description, .id'
```

以下のように、設定したセキュリティグループ名を含んだリストが出力された場合、作成は完了です。

```

"default"
"default"
"<プロジェクトのデフォルトセキュリティグループID>"
...
"<新規作成したセキュリティグループ名>"
"<新規作成したセキュリティグループID>"
...

```

3.3.2 セキュリティグループルール作成

セキュリティグループにセキュリティグループルールを設定し、確認する手順を解説します。



ヒント

お互いにセキュリティグループが使える仮想サーバ同士の通信の場合、基本的に通信相手をセキュリティグループIDで指定することを推奨します。



ヒント

同じセキュリティグループが設定されている仮想サーバ同士の通信(自らに返ってくる通信も含む)にも明示的にその通信を許可するルールが必要です。自分自身のセキュリティグループIDを指定することで解決します。



ヒント

ingressの場合は送信元、egressの場合は宛先のIPアドレスを指定します。

- 以下で、必要な設定を行います。

```
$ DIRECTION=<通信の方向、ingress、egress で指定>
```

```
$ PROTOCOL=<通信プロトコルtcp、udp、icmp、0~65535で指定>
```

```
$ MIN_PORT_NUM=<最小ポート番号0~65535で指定>
```

```
$ MAX_PORT_NUM=<最大ポート番号 0~65535で指定>
```

```
$ SG_ID=<ルールを追加したいセキュリティグループのセキュリティグループID>
```

```
$ REMOTE_IP=<許可するIPアドレス「XXX.XXX.XXX.0/24」等>
```

もしくは

```
$ REMOTE_GROUP_ID=<許可するセキュリティグループID>
```

2. 次のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/security-group-rules -X POST ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"security_group_rule": {"direction": "'$DIRECTION'",  
"port_range_min": '$MIN_PORT_NUM', "port_range_max": '$MAX_PORT_NUM',  
"protocol": "'$PROTOCOL'", "remote_ip_prefix": "'$REMOTE_IP'",  
"security_group_id": "'$SG_ID'"}}' | jq .
```

もしくは

```
$ curl -Ss $NETWORK/v2.0/security-group-rules -X POST ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"security_group_rule": {"direction": "'$DIRECTION'",  
"port_range_min": '$MIN_PORT_NUM', "port_range_max": '$MAX_PORT_NUM',  
"protocol": "'$PROTOCOL'", "remote_group_id": "'$REMOTE_GROUP_ID'",  
"security_group_id": "'$SG_ID'"}}' | jq .
```

以下のレスポンスが返ります。

```
{  
  "security_group_rule": {  
    "remote_group_id": "<許可するセキュリティグループID>",  
    "direction": "<通信の方向>",  
    "remote_ip_prefix": "<許可するIPアドレス>",  
    "protocol": "<プロトコル>",  
    "tenant_id": "<プロジェクトID>",  
    "port_range_max": <最大ポート番号>,  
    "security_group_id": "<ルールを追加したセキュリティグループID>",  
    "port_range_min": <最小ポート番号>,  
    "ethertype": "IPv4",  
    "id": "<セキュリティグループルールID>",  
    "availability_zone": null  
  }  
}
```

3. 設定したセキュリティグループルールを確認するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/security-group-rules -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" ¥  
| jq .
```

以下のように、設定したセキュリティグループルールを含んだリストが出力された場合、作成は完了です。

```
{  
  "security_group_rules": [  
    ...  
    {  
      "remote_group_id": "<許可するセキュリティグループID>",  
      "direction": "<通信の方向>",  
      "remote_ip_prefix": "<許可するIPアドレス>",  
      "protocol": "<プロトコル>",  
      ...  
    }  
  ]  
}
```

```

    "tenant_id": "<プロジェクトID>",
    "port_range_max": <最大ポート番号>,
    "security_group_id": "<ルールを追加したセキュリティグループID>",
    "port_range_min": <最小ポート番号>,
    "ethertype": "IPv4",
    "id": "<セキュリティグループルールID>",
  }
  ...
]
}

```

3.3.3 セキュリティグループルールの設定例

目的別のセキュリティグループルールの設定例を示します。

以下の設定を行います。

- ingress : PING (ICMP:0 ~255)
- ingress : SSH (TCP:22)
- ingress : HTTP (TCP:80)
- ingress : HTTPS (TCP:443)
- ingress : NTP (UDP:123)
- ingress : KMS (TCP:1688)
- ingress : RDP (TCP:3389)

egressはデフォルト値を使用します。

ingress : PING (ICMP:0 ~255)

```

$ DIRECTION=ingress
$ PROTOCOL=icmp
$ MIN_PORT_NUM=0
$ MAX_PORT_NUM=255
$ REMOTE_IP=<XXX. XXX. XXX. 0/24>
もしくは
$ REMOTE_GROUP_ID=<セキュリティグループID>
$ SG_ID=<設定するセキュリティグループID>

```

ingress : SSH (TCP:22)

```

$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=22
$ MAX_PORT_NUM=22
$ REMOTE_IP=<XXX. XXX. XXX. 0/24>
もしくは
$ REMOTE_GROUP_ID=<セキュリティグループID>
$ SG_ID=<設定するセキュリティグループID>

```

ingress : HTTP (TCP:80)

```

$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=80
$ MAX_PORT_NUM=80
$ REMOTE_IP=<XXX. XXX. XXX. 0/24>
もしくは
$ REMOTE_GROUP_ID=<セキュリティグループID>
$ SG_ID=<設定するセキュリティグループID>

```

ingress : HTTPS (TCP:443)

```

$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=443

```

```
$ MAX_PORT_NUM=443
$ REMOTE_IP=<XXX. XXX. XXX. 0/24>
もしくは
$ REMOTE_GROUP_ID=<セキュリティグループID>
$ SG_ID=<設定するセキュリティグループID>
```

ingress : NTP(UDP:123)

```
$ DIRECTION=ingress
$ PROTOCOL=udp
$ MIN_PORT_NUM=123
$ MAX_PORT_NUM=123
$ REMOTE_IP=<XXX. XXX. XXX. 0/24>
もしくは
$ REMOTE_GROUP_ID=<セキュリティグループID>
$ SG_ID=<設定するセキュリティグループID>
```

ingress : KMS(TCP:1688)

```
$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=1688
$ MAX_PORT_NUM=1688
$ REMOTE_IP=<XXX. XXX. XXX. 0/24>
もしくは
$ REMOTE_GROUP_ID=<セキュリティグループID>
$ SG_ID=<設定するセキュリティグループID>
```

ingress : RDP(TCP:3389)

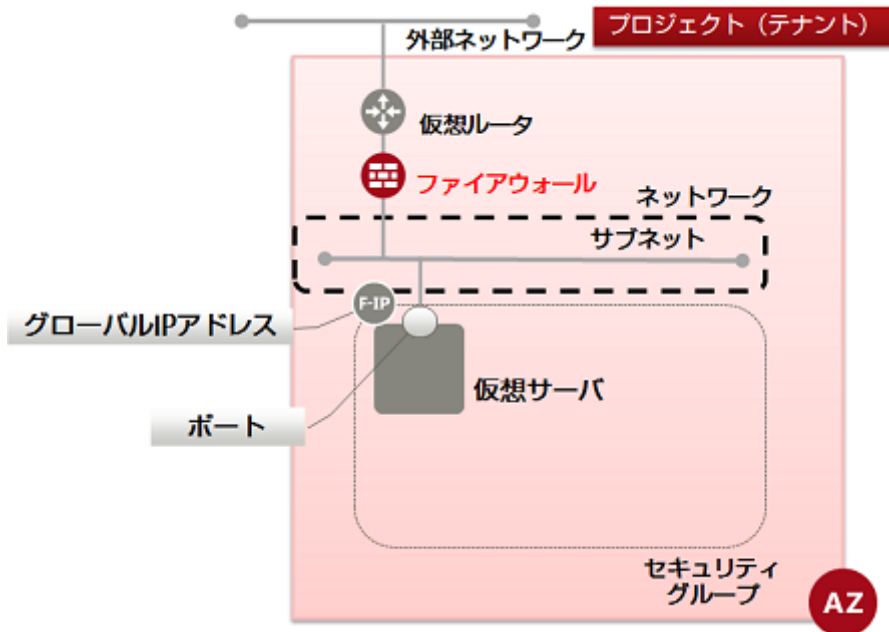
```
$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=3389
$ MAX_PORT_NUM=3389
$ REMOTE_IP=<XXX. XXX. XXX. 0/24>
もしくは
$ REMOTE_GROUP_ID=<セキュリティグループID>
$ SG_ID=<設定するセキュリティグループID>
```

その他の詳しい設定項目は機能説明書をご参照ください。

3.4 ファイアーウォール作成

3.4.1 ファイアーウォールルール(IPアドレス指定)作成

仮想ルータにパケットフィルタを設定するファイアーウォールサービスを利用するために、ファイアーウォールルール(IPアドレス指定)を作成し、確認するまでの手順を解説します。



ヒント

ファイアーウォールサービスの設定は以下の要素から構成され、番号順にフィルタリング情報を構成していきます。ファイアーウォールを仮想ルータに関連付けることで、設定したフィルタリングが行われるようになります。

デフォルトではすべてのトラフィックを遮断するようになっており(ホワイトリスト形式)、許可ルール(AC=allow)によって定義されたもののみFWを通過します。

1. ファイアーウォールルールの作成
2. ファイアーウォールポリシーを作成し、ルール群を登録
3. ポリシーを指定してファイアーウォールを作成し、仮想ルータに関連付け

ファイアーウォールルールを作成するためには、以下を設定します。

- 送信元または送信先IPアドレス
- 通信に使用するプロトコル
- 通信に使用するポート番号
- 通信の許可／禁止

本項目では、具体的に以下にあげたようなルールの作成手順を解説します。

- 許可: 送信元IPアドレス:TCP
- 許可: 送信元IPアドレス:UDP
- 許可: 送信元IPアドレス:ICMP

ファイアーウォールは細かな設定をすることが出来ますが、本ガイドではよくあるルールの作成方法を扱っています。さらに細かな設定項目と設定方法は機能説明書とAPIリファレンスマニュアルを参照してください。

1. 以下の設定をします。

```
$ FWR_NAME=<FWルール名 (任意)>
```

```
$ AC=allow
```

```
$ SOURCE_IP=<許可するIPアドレス「XXX.XXX.XXX.0/24」等>
```

```
$ PROTOCOL=<プロトコル（tcp、udp、icmpで指定）>
```

```
$ AVAILABILITY_ZONE=<作成先アベイラビリティゾーン名>
```

2. 次のAPIを実行します。

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewall_rules -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"firewall_rule": {"name": "$FWR_NAME",
"action": "$AC", "source_ip_address": "$SOURCE_IP", "protocol": "$PROTOCOL",
"availability_zone": "$AVAILABILITY_ZONE"}}' | jq .
```

以下のレスポンスが返ってきます。

```
{
  "firewall_rule": {
    "protocol": "<プロトコル>",
    "description": "",
    "ip_version": 4,
    "tenant_id": "<プロジェクトID>",
    "enabled": true,
    "source_ip_address": "<設定したIPアドレス>",
    "destination_ip_address": null,
    "firewall_policy_id": null,
    "action": "allow",
    "shared": false,
    "source_port": null,
    "position": null,
    "destination_port": null,
    "id": "<新規ファイアーウォールルールID>",
    "name": "<新規ファイアーウォールルール名>",
    "availability_zone": "<設定したアベイラビリティゾーン>"
  }
}
```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

3. 作成したファイアーウォールルールを確認するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/fw/firewall_rules -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | \
jq '.firewall_rules[] | .name, .id'
```

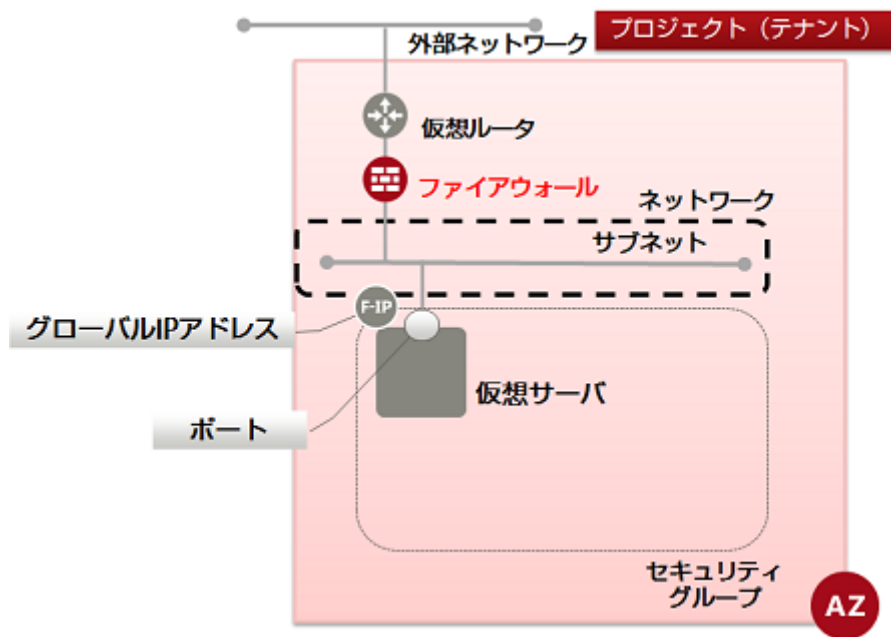
以下のように、設定したファイアーウォールルール名を含んだリストが出力された場合、作成は完了です。

```
...
"<新規ファイアーウォールルール名>"
"<新規ファイアーウォールルールID>"
...
```

ファイアーウォールルールを作成が完了すると、それらを束ねるファイアーウォールポリシーの作成に進みます。

3.4.2 ファイアーウォールルール(ポート番号指定)作成

仮想ルータにパケットフィルタを設定するファイアーウォールサービスを利用するために、ファイアーウォールルール(ポート番号指定)を作成し、確認するまでの手順を解説します。



ファイアウォールサービスの設定は以下の要素から構成され、番号順にフィルタリング情報を構成していきます。ファイアウォールを仮想ルータに関連付けることで、設定したフィルタリングが行われるようになります。

デフォルトではすべてのトラフィックを遮断するようになっており(ホワイトリスト形式)、許可ルール(AC=allow)によって定義されたもののみFWを通過します。

1. ファイアウォールルールの作成
2. ファイアウォールポリシーを作成し、ルール群を登録
3. ポリシーを指定してファイアウォールを作成し、仮想ルータに関連付け

本ガイドでは、具体的に以下にあげたルールの作成手順を解説します。

- 許可 : SSH(TCP:22)
- 許可 : HTTP(TCP:80)
- 許可 : HTTPS(TCP:443)
- 許可 : NTP(UDP:123)
- 許可 : KMS(TCP:1688)
- 許可 : RDP(TCP:3389)

ファイアウォールは細かな設定をすることが出来ますが、本ガイドではよくあるルールの作成方法を扱っています。さらに細かな設定項目と設定方法は機能説明書とAPIリファレンスマニュアル参照してください。

1. 以下の設定をします。

```
$ FWR_NAME=<FWルール名 (任意) >
```

```
$ AC=allow
```

```
$ DESTINATION_PORT=<宛先ポート番号、または範囲、例 : "0 - 255"、22、等>
```

```
$ PROTOCOL=<プロトコル (tcp、udp、icmpで指定) >
```

```
$ AVAILABILITY_ZONE=<作成先アベイラビリティゾーン名>
```

2. 次のAPIを実行します。

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewall_rules -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"firewall_rule": {"name": "'$FWR_NAME'",
"action": "'$AC'", "destination_port": "'$DESTINATION_PORT'", "protocol":
"'$PROTOCOL'",
"availability_zone": "'$AVAILABILITY_ZONE'" }}' | jq .
```

以下のレスポンスが返ってきます。

```
{
  "firewall_rule": {
    "protocol": "<プロトコル>",
    "description": "",
    "ip_version": 4,
    "tenant_id": "<プロジェクトID>",
    "enabled": true,
    "source_ip_address": "null",
    "destination_ip_address": null,
    "firewall_policy_id": null,
    "action": "allow",
    "shared": false,
    "source_port": null,
    "position": null,
    "destination_port": "<設定したポート番号>",
    "id": "<新規ファイアーウォールルールID>",
    "name": "<新規ファイアーウォールルール名>",
    "availability_zone": "<設定したアベイラビリティゾーン>"
  }
}
```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

- 作成したファイアーウォールルールを確認するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/fw/firewall_rules -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.firewall_rules[] | .name, .id'
```

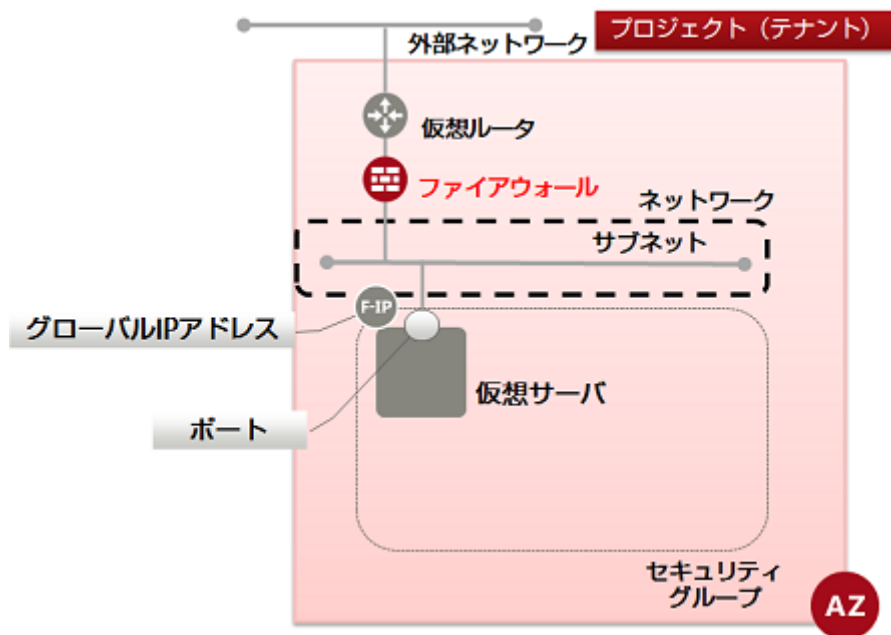
以下のように、設定したファイアーウォールルール名を含んだリストが出力された場合、作成は完了です。

```
...
"<新規ファイアーウォールルール名>"
"<新規ファイアーウォールルールID>"
...
```

ファイアーウォールルールを作成が完了すると、それらを束ねるファイアーウォールポリシーの作成に進みます。

3.4.3 ファイアーウォールルール(ICMP許可)作成

仮想ルータにパケットフィルタを設定するファイアーウォールサービスを利用するために、ファイアーウォールルール(ICMP許可)を作成し、確認するまでの手順を解説します。



ヒント

ファイアウォールサービスの設定は以下の要素から構成され、番号順にフィルタリング情報を構成していきます。ファイアウォールを仮想ルータに関連付けることで、設定したフィルタリングが行われるようになります。

デフォルトではすべてのトラフィックを遮断するようになっており(ホワイトリスト形式)、許可ルール(AC=allow)によって定義されたもののみFWを通過します。

1. ファイアウォールルールの作成
2. ファイアウォールポリシーを作成し、ルール群を登録
3. ポリシーを指定してファイアウォールを作成し、仮想ルータに関連付け

本ガイドでは、具体的に以下にあげたルールの作成手順を解説します。

- 許可 : PING (ICMP:0~255)

ファイアウォールは細かな設定をすることが出来ますが、本ガイドではよくあるルールの作成方法を扱っています。さらに細かな設定項目と設定方法は機能説明書とAPIリファレンスマニュアルを参照してください。

1. 以下の設定をします。

```
$ FWR_NAME=<FWルール名 (任意) >
```

```
$ AC=allow
```

```
$ PROTOCOL=icmp
```

```
$ AVAILABILITY_ZONE=<作成先アベイラビリティゾーン名>
```

2. 次のAPIを実行します。

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewall_rules -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"firewall_rule": {"name": "'$FWR_NAME'",
"action": "'$AC'", "protocol": "'$PROTOCOL'",
"availability_zone": "'$AVAILABILITY_ZONE'"}}' | jq .
```

以下のレスポンスが返ってきます。

```
{
  "firewall_rule": {
    "protocol": "icmp",
    "description": "",
    "ip_version": 4,
    "tenant_id": "<プロジェクトID>",
    "enabled": true,
```

```

"source_ip_address": "null",
"destination_ip_address": null,
"firewall_policy_id": null,
"action": "allow",
"shared": false,
"source_port": null,
"position": null,
"destination_port": null,
"id": "<新規ファイアウォールルールID>",
"name": "<新規ファイアウォールルール名>",
"availability_zone": "<設定したアベイラビリティゾーン>"
}
}

```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

- 作成したファイアウォールルールを確認するため、以下のAPIを実行します。

```

$ curl -Ss $NETWORK/v2.0/fw/firewall_rules -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | \
jq '.firewall_rules[] | .name, .id'

```

以下のように、設定したファイアウォールルール名を含んだリストが出力された場合、作成は完了です。

```

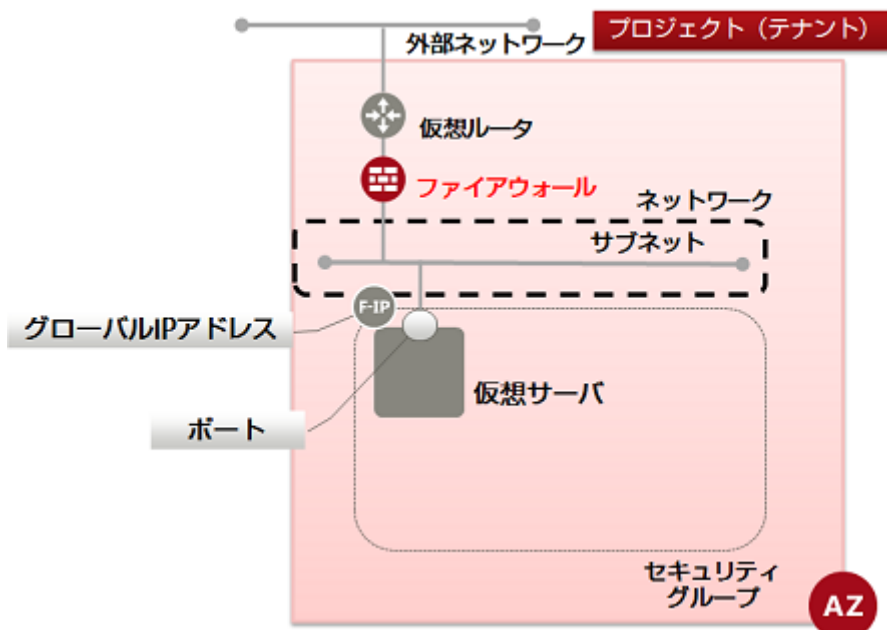
...
"<新規ファイアウォールルール名>"
"<新規ファイアウォールルールID>"
...

```

ファイアウォールルールを作成が完了すると、それらを束ねるファイアウォールポリシーの作成に進みます。

3.4.4 ファイアウォールルール(拒否ルール)作成

仮想ルータにパケットフィルタを設定するファイアウォールサービスを利用するために、ファイアウォールルール(設定済みのルール以外すべて拒否)を作成し、確認するまでの手順を解説します。



ヒント

ファイアウォールサービスの設定は以下の要素から構成され、番号順にフィルタリング情報を構成していきます。ファイアウォールを仮想ルータに関連付けることで、設定したフィルタリングが行われるようにします。

デフォルトではすべてのトラフィックを遮断するようになっており(ホワイトリスト形式)、許可ルール(AC=allow)によって定義されたもののみFWを通過します。

1. ファイアウォールルールの作成
2. ファイアウォールポリシーを作成し、ルール群を登録
3. ポリシーを指定してファイアウォールを作成し、仮想ルータに関連付け

本ガイドでは、具体的に以下にあげたルールの作成手順を解説します。

- 設定されていないポートを全て閉じる

ファイアウォールは細かな設定をすることが出来ますが、本ガイドではよくあるルールの作成方法を扱っています。さらに細かな設定項目と設定方法は機能説明書とAPIリファレンスマニュアルを参照してください。

1. 以下の設定をします。

```
$ FWR_NAME=<FWルール名 (任意) >
```

```
$ AC=deny
```

```
$ AVAILABILITY_ZONE=<作成先アベイラビリティゾーン名>
```

2. 次のAPIを実行します。

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewall_rules -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"firewall_rule": {"name": "'$FWR_NAME'",
"action": "'$AC'", "availability_zone": "'$AVAILABILITY_ZONE'"}}' | jq .
```

以下のレスポンスが返ってきます。

```
{
  "firewall_rule": {
    "protocol": null,
    "description": "",
    "ip_version": 4,
    "tenant_id": "<プロジェクトID>",
    "enabled": true,
    "source_ip_address": "null",
    "destination_ip_address": null,
    "firewall_policy_id": null,
    "action": "deny",
    "shared": false,
    "source_port": null,
    "position": null,
    "destination_port": null,
    "id": "<新規ファイアウォールルールID>",
    "name": "<新規ファイアウォールルール名>",
    "availability_zone": "<設定したアベイラビリティゾーン>"
  }
}
```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

3. 作成したファイアウォールルールを確認するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/fw/firewall_rules -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | \
jq '.firewall_rules[] | .name, .id'
```

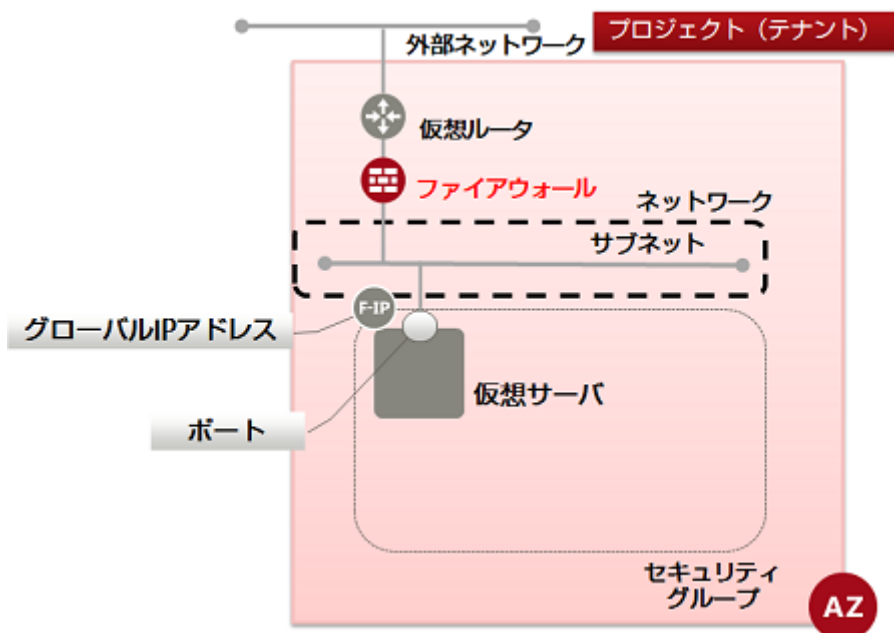
以下のように、設定したファイアウォールルール名を含んだリストが出力された場合、作成は完了です。

```
...
"<新規ファイアウォールルール名>"
"<新規ファイアウォールルールID>"
...
```

ファイアウォールルールの作成が完了すると、それらを束ねるファイアウォールポリシーの作成に進みます。

3.4.5 ファイアウォールポリシー作成

複数のファイアウォールルールを束ねるファイアウォールポリシーを作成し、確認するまでの手順を解説します。



リスト内のルールの順番によってルールに優先順位がつけられており、上から順に順次検証され、通信の可否を制御します。



ヒント

ポリシー内で自動的に「DENY ALL」のルールが最後尾に追加されます。これにより、許可ルールの定義されないトラフィックはデフォルトで遮断されます。(ホワイトリスト方式)ただし、この自動的に追加される「DENY ALL」ルールは暗黙ルールとなっており、ポリシーに表示されません。

ファイアウォールサービスの設定は以下の要素から構成され、番号順にフィルタリング情報を構成していきます。ファイアウォールを仮想ルータに関連付けることで、設定したフィルタリングが行われるようにします。

1. ファイアウォールルールの作成
2. ファイアウォールポリシーを作成し、ルール群を登録
3. ポリシーを指定してファイアウォールを作成し、仮想ルータに関連付け
1. ファイアウォールルールの一覧を取得するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/fw/firewall_rules -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | \
jq '.firewall_rules[] | .name, .id'
```

以下のようなフォーマットでファイアウォールルールのリストが表示されます。ここからポリシーに必要なルールを選択します。

```
...
"<ファイアウォールルール名>"
"<ファイアウォールルールID>"
...
```

2. 以下の設定をします。

```
$ NAME=<ファイアウォールポリシー名 (任意) >
```



```
$ FWR1=<ファイアーウォールルールID (1) >
```

```
$ FWR2=<ファイアーウォールルールID (2) >
```

```
$ FWR3=<ファイアーウォールルールID (3) >
```

※ファイアーウォールポリシーに設定する分のファイアーウォールルールIDを記載します。

```
$ AVAILABILITY_ZONE=<ファイアーウォールルールで設定されているものと同じアベイラビリティゾーン>
```

3. 次のAPIを実行します。

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewall_policies ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"firewall_policy":{"firewall_rules": [ "'$FWR1'", "'$FWR2'", "'$FWR3'",  
    "'<ファイアーウォールポリシーに設定する数分の領域を追加します>'"],  
    "name": "'$NAME'", "availability_zone": "'$AVAILABILITY_ZONE'"}}' | jq .
```

以下のレスポンスが返ってきます。ファイアーウォールルールIDの表示される順番がそのまま優先順位となります。

```
{  
  "firewall_policy": {  
    "name": "<新規ファイアーウォールポリシー名>",  
    "firewall_rules": [  
      "<設定したファイアーウォールルールID (1) >",  
      "<設定したファイアーウォールルールID (2) >",  
      ..  
      "<設定したファイアーウォールルールID (n) >"  
    ],  
    "tenant_id": "<プロジェクトID>",  
    "audited": false,  
    "shared": false,  
    "id": "<新規ファイアーウォールポリシーID>",  
    "description": "",  
    "availability_zone": "<設定したアベイラビリティゾーン>"  
  }  
}
```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

4. 作成したファイアーウォールポリシーを確認するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/fw/firewall_policies -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" ¥  
| jq .
```

以下のように、設定したファイアーウォールポリシー名を含んだリストが出力された場合、作成は完了です。

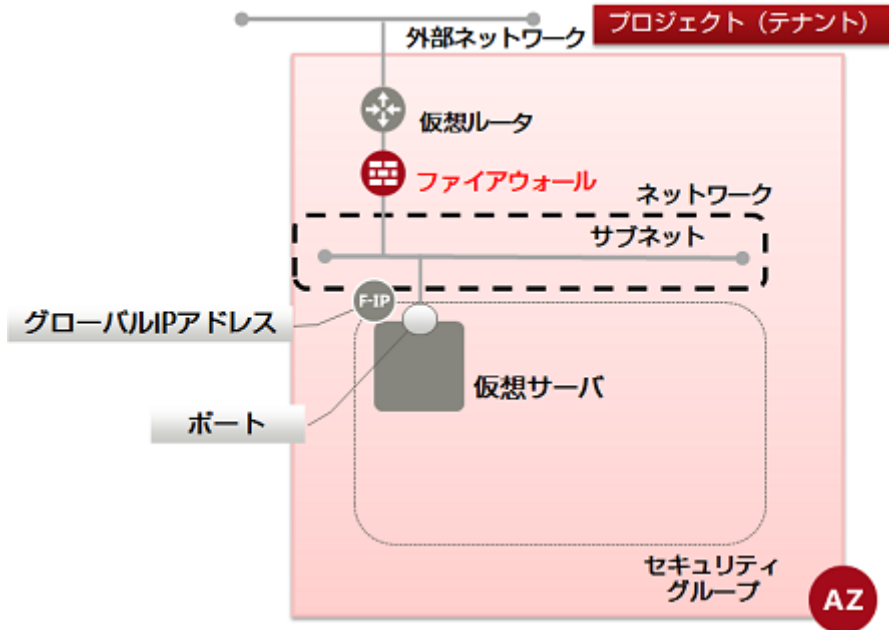
```
{  
  "firewall_policies": [  
    ..  
    {  
      "name": "<新規ファイアーウォールポリシー名>",  
      "firewall_rules": [  
        "<設定したファイアーウォールルールID (1) >",  
        "<設定したファイアーウォールルールID (2) >",  
        ..  
        "<設定したファイアーウォールルールID (n) >"  
      ],  
      "tenant_id": "<プロジェクトID>",  
      "audited": false,  
      "shared": false,  
      "id": "<新規ファイアーウォールポリシーID>",  
      "description": "",  
      "availability_zone": "<設定したアベイラビリティゾーン>"  
    }  
  ]  
}
```

```
},  
...  
]  
}
```

ファイアウォールポリシーの作成が完了すると、それを仮想ルータに紐づけるファイアウォールの作成に進みます。

3.4.6 ファイアウォール作成

ルールを登録したファイアウォールポリシーを指定して、仮想ルータにファイアウォールを作成し、確認する手順を解説します。



ファイアウォールサービスの設定は以下の要素から構成され、番号順にフィルタリング情報を構成していきます。ファイアウォールを仮想ルータに関連付けることで、設定したフィルタリングが行われるようになります。

1. ファイアウォールルールの作成
2. ファイアウォールポリシーを作成し、ルール群を登録
3. ポリシーを指定してファイアウォールを作成し、仮想ルータに関連付け
1. 以下の設定をします。

```
$ FW_NAME=<新規ファイアウォール名 (任意)>
```

```
$ FWP_ID=<設定するファイアウォールポリシーID>
```

```
$ ROUTER_ID=<適用する仮想ルータID>
```

```
$ AVAILABILITY_ZONE=<ファイアウォールポリシーで設定されているものと同じアベイラビリティゾーン>
```

2. 次のAPIを実行します。

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewalls -H "X-Auth-Token: $OS_AUTH_TOKEN" \\  
-H "Content-Type: application/json" -d '{"firewall": {"name": "'$FW_NAME'", \\  
"firewall_policy_id": "'$FWP_ID'", "router_id": "'$ROUTER_ID'", \\  
"availability_zone": "'$AVAILABILITY_ZONE'"}}' | jq .
```

以下のレスポンスが返ってきます。

```
{
```

```

"firewall": {
  "status": "PENDING_CREATE",
  "router_id": "<設定した仮想ルータID>",
  "name": "<新規ファイアーウォール名>",
  "admin_state_up": true,
  "tenant_id": "<プロジェクトID>",
  "firewall_policy_id": "<設定したファイアーウォールポリシーID>",
  "id": "新規ファイアーウォールID",
  "description": "",
  "availability_zone": "アベイラビリティゾーン"
}
}

```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

- 作成したファイアーウォールを確認するため、以下のAPIを実行します。

```
$ curl -sS $NETWORK/v2.0/fw/firewalls -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

以下のように、設定したファイアーウォールポリシー名を含んだリストが出力された場合、作成は完了です。

```

{
  "firewalls": [
    ...
    {
      "status": "PENDING_CREATE",
      "router_id": "<設定した仮想ルータID>",
      "name": "<新規ファイアーウォール名>",
      "admin_state_up": true,
      "tenant_id": "<プロジェクトID>",
      "firewall_policy_id": "<設定したファイアーウォールポリシーID>",
      "id": "新規ファイアーウォールID",
      "description": "",
      "availability_zone": "アベイラビリティゾーン"
    },
    ...
  ]
}

```



警告

自プロジェクト以外のファイアーウォールポリシー、またはルールが設定されると、作成されたファイアーウォールは異常な状態となります。その状態になった場合、ユーザーでの削除ができなくなり、本サービス基盤の管理者による削除が必要になります。ファイアーウォールおよびそのポリシー/ルールは【必ず同一のプロジェクトID】となるよう作成してください。

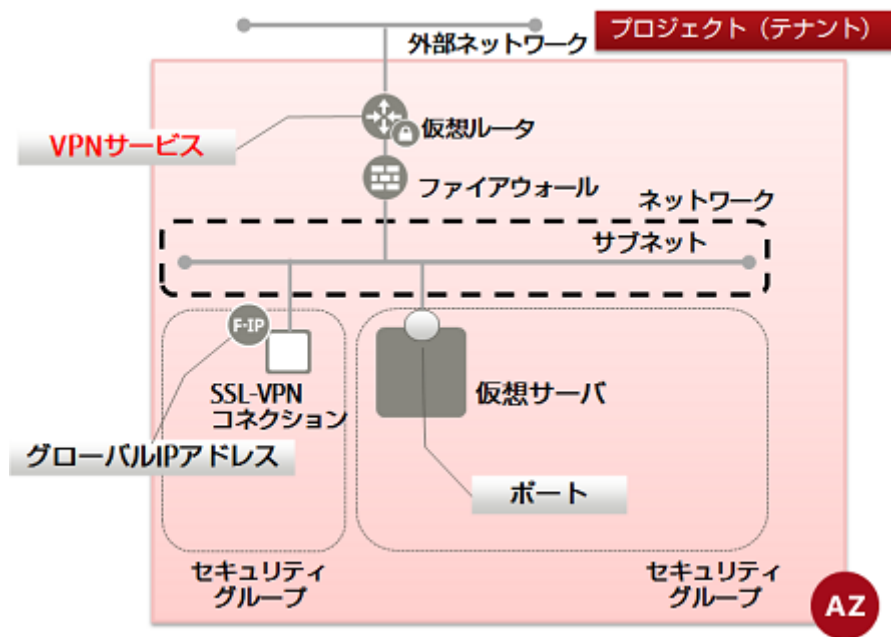
3.5 簡易構成のSSL-VPN接続 (V2サービス)

3.5.1 SSL-VPN接続 (V2サービス/クライアント証明書利用)

この章では、クライアント証明書を利用したSSL-VPN接続の手順を解説します。自己署名証明書を利用したSSL-VPN接続の手順は、[SSL-VPN接続 \(V2サービス/自己署名証明書利用\)](#)をご参照ください。

3.5.1.1 VPNサービスの作成 (V2サービス/クライアント証明書利用)

VPNサービスを作成する手順を解説します。



1. 以下で必要な設定を行います。

```
$ SUBNET_ID=<SSL-VPNで接続したいサブネットID>
```

```
$ ROUTER_ID=<SSL-VPNで利用するルータID>
```

```
$ VPN_SERVICE_NAME=<VPNサービス名 (任意) >
```

```
$ ADMIN_STATE_UP=true
```

```
$ AZ=<アベイラビリティゾーン>
```



警告

SSL-VPN接続 (V2サービス)で接続するサブネットのパラメーター定義が、以下のように指定されている必要があります。

- cidr
ネットワークアドレスのマスク値が「16bit～29bit」の範囲内で作成されている。
例) 192.168.1.0/24
- gateway_ip
VPNサービスで指定する仮想ルータのIPアドレスが指定されている。

2. 次のAPIを実行します。

```
$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" ¥
```

```
-H "Content-Type: application/json" -d '{"vpnservice": {"subnet_id": "' $SUBNET_ID' ",
"router_id": "' $ROUTER_ID' ", "name": "' $VPN_SERVICE_NAME' ",
"admin_state_up": "' $ADMIN_STATE_UP' ", "availability_zone": "' $AZ' " }}' | jq .
```

以下のレスポンスが返ってきます。

```
{
  "vpnservice": {
    "availability_zone": "<アベイラビリティゾーン>",
    "router_id": "<SSL-VPNで接続するルータID>",
    "status": "PENDING_CREATE",
    "name": "<VPNサービス名>",
    "admin_state_up": true,
    "subnet_id": "<SSL-VPNで接続するサブネットID>",
    "tenant_id": "<SSL-VPNで接続するプロジェクトID>",
    "id": "<VPNサービスID>",
    "description": ""
  }
}
```

3. 作成したVPNサービスを確認するため、以下のAPIを実行します。

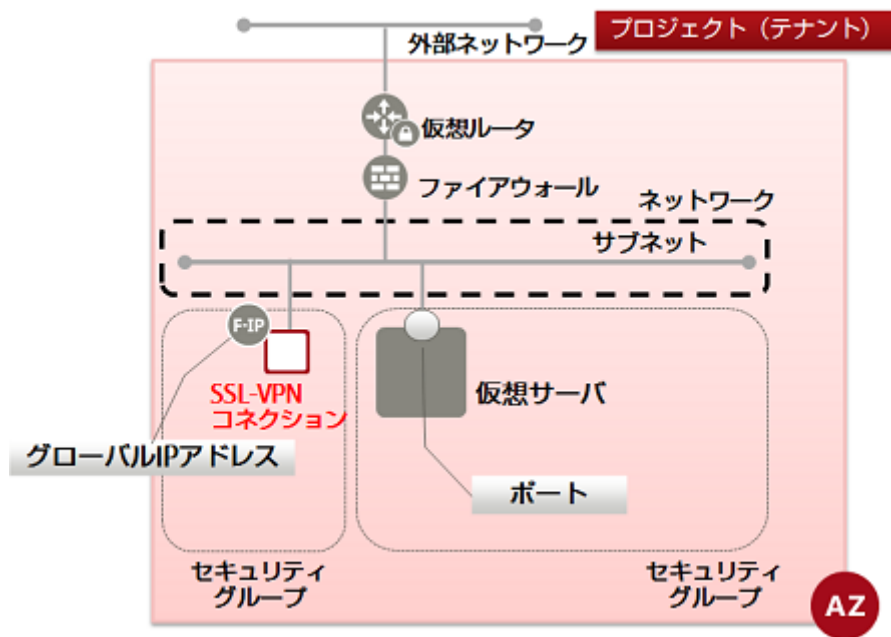
```
$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" ¥
-H "Content-Type: application/json" | jq .
```

以下のように、作成したVPNサービスのstatusがPENDING_CREATEになっていることを確認できた場合、作成は完了です。

```
{
  "vpnservices": [
    {
      "availability_zone": "<アベイラビリティゾーン>",
      "router_id": "<SSL-VPNで接続するルータID>",
      "status": "PENDING_CREATE",
      "name": "<VPNサービス名>",
      "admin_state_up": true,
      "subnet_id": "<SSL-VPNで接続するサブネットID>",
      "tenant_id": "<SSL-VPNで接続するプロジェクトID>",
      "id": "<VPNサービスID>",
      "description": ""
    }
  ]
}
```

3.5.1.2 SSL-VPNコネクションの作成 (V2サービス/クライアント証明書利用)

SSL-VPNコネクション (V2サービス) を作成する手順を解説します。



1. 以下で必要な設定を行います。

```
$ NAME=<コネクション名 (任意)>
```

```
$ VPNSERVICE_ID=<VPNサービスID>
```

```
$ PROTOCOL=tcp
```

```
$ AZ=<アベイラビリティゾーン>
```

```
$ CLIENT_ADDRESS_POOL_CIDR=<VPNトンネルの仮想ネットワークcidr (任意)>
```

CLIENT_ADDRESS_POOL_CIDRはXXX.XXX.XXX.XXX/XX形式でネットワークアドレスの指定を行ってください。

例) 192.168.246.0/24



本サービスで使用しているネットワークアドレスやクライアントPCが接続しているローカルネットワークアドレスと競合しないネットワークアドレスを指定してください。



本手順を利用してSSL-VPNコネクションを作成した場合、以下の設定が自動的に追加されます。

注

a. ファイアウォールルール

SSL-VPN接続専用SSL-VPNコネクションへのアクセスを許可するファイアウォールルール(443/TCP)が自動的に追加されます。

自動的に追加されるファイアウォールルールを変更したい場合は、SSL-VPNコネクション作成後に変更してください。

b. スタティックルーティング

SSL-VPN接続専用SSL-VPNコネクションに設定した「client_address_pool_cidr」へのルーティングが仮想ルータに追加されます。

c. セキュリティグループ

SSL-VPN接続専用全ての通信許可が設定されたセキュリティグループが作成されます。

SSL-VPNコネクションを作成する際に「security_groups」パラメーターを指定することでユーザーが作成したセキュリティグループを利用することも可能です。

詳細については IaaS APIリファレンス(Network編)を参照してください。

d. グローバルIPアドレス

「floatingips」パラメーターを省略するとSSL-VPNコネクションに割り当てられるグローバルIPアドレスが作成されます。

事前に作成したグローバルIPを利用する場合は「floatingips」パラメーターを指定してください。

詳細については IaaS APIリファレンス(Network編)を参照してください。



自動的に追加された設定の他に、SSL-VPNコネクションの「client_address_pool_cidrs」パラメーターに設定した仮想ネットワークから仮想サーバへのアクセス許可をセキュリティグループとファイアウォールルールに定義する必要があります。

注

2. 次のAPIを実行します。

```
$ curl -sS $NETWORK/v2.0/vpn/ssl-vpn-v2-connections -X POST ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{"ssl_vpn_v2_connection": {"name": "$NAME",
"client_address_pool_cidrs": [ "$CLIENT_ADDRESS_POOL_CIDR" ],
"admin_state_up": true, "vpnservice_id": "$VPNSERVICE_ID",
"availability_zone": "$AZ", "protocol": "$PROTOCOL"} }' | jq .
```

以下のレスポンスが返ってきます。

```
{
  "ssl_vpn_v2_connections": [
    {
      "access_points": [
        {
          "floatingip": null,
          "client_address_pool_cidr": "<VPNトンネルの仮想ネットワークcidr>",
          "internal_gateway": "<SSL-VPNコネクションのプライベートIPアドレス>",
          "external_address": "<SSL-VPNのグローバルIPアドレス>"
        }
      ],
      "security_groups": [
        "<SSL-VPN接続専用のセキュリティグループ>"
      ],
      "protocol": "tcp",
      "availability_zone": "<アベイラビリティゾーン>",
      "tenant_id": "<プロジェクトID>",
      "name": "<SSL-VPNコネクション名>",
      "admin_state_up": true,
      "client_address_pool_cidrs": [
        "<VPNトンネルの仮想ネットワークcidr>"
      ]
    }
  ]
}
```

```

    ],
    "credential_id": null,
    "vpnservice_id": "<VPNサービス名>",
    "id": "<SSL-VPNコネクションID>",
    "status": "PENDING_CREATE"
  }
]
}

```

3. 作成したSSL-VPNコネクションの状態を確認するため、以下のAPIを実行します。

```

$ curl -sS $NETWORK/v2.0/vpn/ssl-vpn-v2-connections -X GET ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .

```

以下のように、SSL-VPNコネクションのstatusがACTIVEになっていることを確認します。

```

{
  "ssl_vpn_v2_connections": [
    {
      "access_points": [
        {
          "floatingip": null,
          "client_address_pool_cidr": "<VPNトンネルの仮想ネットワークcidr>",
          "internal_gateway": "<SSL-VPNコネクションのプライベートIPアドレス>",
          "external_address": "<SSL-VPNのグローバルIPアドレス>"
        }
      ],
      "security_groups": [
        "<SSL-VPN接続専用のセキュリティグループ>"
      ],
      "protocol": "tcp",
      "availability_zone": "<アベイラビリティゾーン>",
      "tenant_id": "<プロジェクトID>",
      "name": "<SSL-VPNコネクション名>",
      "admin_state_up": true,
      "client_address_pool_cidrs": [
        "<VPNトンネルの仮想ネットワークcidr>"
      ],
      "credential_id": null,
      "vpnservice_id": "<VPNサービス名>",
      "id": "<SSL-VPNコネクションID>",
      "status": "ACTIVE"
    }
  ]
}

```

作成直後は、statusがPENDING_CREATEのままとなっている場合があります。その場合は時間をおいてから、再度、確認APIを実行してください。

4. VPNサービスの状態を確認するため、以下のAPIを実行します。

```

$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" ¥
-H "Content-Type:application/json" | jq .

```

以下のように、VPNサービスのstatusがACTIVEになっていることを確認できた場合、作成は完了です。

```

{
  "vpnservices": [
    {
      "availability_zone": "<アベイラビリティゾーン>",
      "router_id": "<SSL-VPNで接続されたルータID>",
      "status": "ACTIVE",
      "name": "<VPNサービス名>",
      "admin_state_up": true,
      "subnet_id": "<SSL-VPNで接続するサブネットID>",
      "tenant_id": "<SSL-VPNで接続するプロジェクトID>",
      "id": "<VPNサービスID>",
    }
  ]
}

```



```
    "description": ""
  }
]
```

3.5.1.3 VPNクライアントとの接続設定 (V2サービス/クライアント証明書利用)

VPNクライアントとの接続設定 (V2サービス) を行う手順を解説します。



注

本手順ではSSL-VPNクライアントPC環境として、以下の環境を使用して確認しています。

- OS : Windows 7 Professional 64bit 日本語版
- VPN クライアント : OpenVPN 2.3.12



注

本手順ではOpenSSL1.1.0を使用して証明書の形式変換を実施しています。

1. CA証明書を作成するための中間証明書とroot証明書をダウンロードします。

<http://rms-digicert.ne.jp/howto/basis/digicert-root-certificates.html> から、中間証明書とroot証明書をダウンロードします。

中間証明書 : [DigiCert Global G2 TLS RSA SHA256 2020 CA1]

root証明書 : [DigiCert Global Root G2]

2. 次のコマンドを実行し、ダウンロードした証明書の形式を変換します。

```
openssl x509 -in <中間証明書名> -inform DER -out <中間証明書名 (任意)>
openssl x509 -in <root証明書名> -inform DER -out <root証明書名 (任意)>
```

3. 変換したことを確認するため、以下のコマンドを実行します。

```
# ls -l
```

以下のように、証明書が表示されることを確認します。

```
<中間証明書名>
```

```
<root証明書名>
```

4. 中間証明書とroot証明書を結合します。

証明書をテキストエディタに貼り付けます。



注

- 作成した2つのPEM形式の証明書のファイルをテキストエディタで開きます。
- 表示した情報の「-----BEGIN CERTIFICATE-----」から「-----END CERTIFICATE-----」までをコピーし、新規テキストファイルに貼り付けます。
- 作成されたテキストファイルが以下のような内容になっていることを確認します。

```
-----BEGIN CERTIFICATE-----
中間証明書ファイル (PEM形式) の記述内容
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
root証明書ファイル (PEM形式) の記述内容
-----END CERTIFICATE-----
```

テキストファイルを任意の名前で保存します。ファイルの拡張子は.crtとしてください。

5. クライアント証明書を発行します。

クライアント証明書の発行については「ポータルユーザズガイド」の「3.3.5.1 認証用の証明書の発行」を参照してください。



警告

SSL-VPN接続で使用するクライアント証明書は必ずユーザー毎に一意の証明書を作成してください。
複数のユーザーが1つの証明書を利用して同時にSSL-VPN接続を行うことはできません。
複数ユーザーで利用する場合、ポータルからユーザー数分証明書を取得してください。

6. クライアント証明書の形式変換

クライアント証明書の形式変換については[証明書](#)の形式変換を参照してください。

本章で説明しているSSL-VPNクライアント(OpenVPN)で使用する秘密鍵は[証明書](#)の形式変換に記載の「暗号化されている秘密鍵」と「暗号化されていない秘密鍵」に対応しています。

7. OpenVPNのインストールを行います。

<https://www.openvpn.jp/download/> から、インストーラをダウンロードして、インストールを実行します。

詳細なインストール方法については「機能説明書」の付録「[SSL-VPNクライアントのセットアップ\(Windows編\)](#)」を参照してください。

8. OpenVPNクライアント設定ファイルの編集を行います。

インストール先のsample-configフォルダにあるclient.ovpnをconfigフォルダにコピーします。

本手順で作成したCA証明書とクライアント証明書/秘密鍵をconfigフォルダに保存しておく必要があります。

configフォルダのclient.ovpnをテキストエディタで開き、以下の通り編集します。

- protoで始まる行

```
proto tcp
```

- remoteで始まる行

```
remote [接続先サーバアドレス (SSL-VPN ConnectionリソースのグローバルIPアドレス)] [接続先ポート ( 443 )]
```

- caで始まる行

```
ca <CA証明書名>
```

- certで始まる行

```
cert <クライアント証明書名>
```

- keyで始まる行

```
key <クライアント秘密鍵名>
```

- comp-lzoで始まる行

```
#comp-lzo
```

- cipherで始まる行

```
cipher AES-128-CBC
```

- http-proxyで始まる行 (HTTPプロキシサーバ経由で接続する場合に指定します)

```
http-proxy <HTTPプロキシサーバアドレス> <HTTPプロキシのポート番号> stdin basic
```

stdin: HTTPプロキシサーバ接続時に、ユーザー名とパスワードの入力を要求されます。

basic: 認証方式は基本認証となります。



例:

注

```
proto tcp
remote xxx.xxx.xxx.xxx 443
ca ca.crt
cert client.crt
key client.key
#comp-lzo
cipher AES-128-CBC
http-proxy xxx.xxx.xxx.xxx 8080 stdin basic
```

9. OpenVPNクライアントを起動します。

OpenVPNクライアントを右クリックし、[管理者として実行]を選択し、管理者権限で起動します。

10. SSL-VPN接続を行います。



SSL-VPN接続実施後に仮想サーバに接続するには、以下にVPNトンネルのネットワークアドレスから接続する仮想サーバへのアクセス許可の設定を実施する必要があります。

注

- SSL-VPN機能を設定した仮想ルータのファイアーウォール
- 仮想サーバに割り当てられたセキュリティグループ

PC端末の起動タスクトレイ上のOpenVPNアイコンを右クリックし、[接続]メニューをクリックします。

タスクトレイ上のOpenVPNアイコンがグリーンに切り替わることが確認できたら、SSL-VPNの接続は完了です。

SSL-VPNクライアントPCから仮想サーバのプライベートIPアドレスを指定することで接続が可能となります。

※SSL-VPN接続を切断したい場合

切断するときはタスクトレイ上のOpenVPNアイコンを右クリックし、[切断]をクリックします。

タスクトレイ上のOpenVPNアイコンがグレーに切り替わることを確認できたら、切断完了です。

3.5.2 SSL-VPN接続 (V2サービス/自己署名証明書利用)

この章では、自己署名証明書を利用したSSL-VPN接続の手順を解説します。クライアント証明書を利用したSSL-VPN接続の手順は、[SSL-VPN接続 \(V2サービス/クライアント証明書利用\)](#)をご参照ください。

3.5.2.1 SSL-VPN用証明書作成 (V2サービス/自己署名証明書利用)

SSL-VPN用証明書を作成する手順を解説します。

本手順はLinux環境 (OS: CentOS6.8、ツール: EasyRSA3.0.1) を使用した手順で作成しています。

実行ユーザーはスーパーユーザーで実施しています。

※本手順では以下ディレクトリで作業することを前提に記載します。

作業ディレクトリ: /root/EasyRSA-3.0.1/

1. CA局を初期化します

```
# ./easyrsa init-pki
```

証明書作成環境が作成されます。

以下のディレクトリに証明書が生成されます。

- CA証明書、DH鍵: ./pki/.
- サーバ・クライアント証明書: ./pki/issued/.
- 秘密鍵: ./pki/private/.

2. CA証明書を作成します。

```
# ./easysrsa build-ca
```

以下のようなメッセージが表示されるため、指示に従い対話式に処理を進めます。

```
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to '/root/EasyRSA-3.0.1/pki/private/ca.key.0MjyMXpDsw'
Enter PEM pass phrase: <CA証明書用パスワード (入力)>
Verifying - Enter PEM pass phrase: <CA証明書用パスワード (再入力)>
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [EasyRSA CA]: <コモンネーム (任意)>
CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
/root/EasyRSA-3.0.1/pki/ca.crt
```

3. CA 証明書が作成されたことを確認するため以下のコマンドを実行します。

```
# ls -l /root/easysrsa/pki/
```

以下のように、作成したCA証明書が出力された場合、作成は完了です。

```
-rw-----. 1 root root 1180 MM DD hh:mm ca.crt
```

4. サーバ証明書と秘密鍵を作成します。

```
# ./easysrsa build-server-full <サーバ証明書と秘密鍵名 (任意)> nopass
```

以下のようなメッセージが表示されるため、指示に従い対話式に処理を進めます

```
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to '/root/EasyRSA-3.0.1/pki/private/server1.key.LCqVP0rjSV'
-----
Using configuration from /root/EasyRSA-3.0.1/openssl-1.0.cnf
Enter pass phrase for /root/EasyRSA-3.0.1/pki/private/ca.key: <CA証明書用パスワード (再入力)>
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName          :PRINTABLE:'<サーバ証明書と秘密鍵名>'
Certificate is to be certified until  MM DD hh:mm:ss YYYY GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
```

5. サーバ証明書と秘密鍵が作成されたことを確認するため、以下のコマンドを実行します。

サーバ証明書が作成されたことを確認するには以下のコマンドを実行します。

```
# ls -l /root/easysrsa/pki/issued/
```

以下のように、作成したサーバ証明書が出力された場合、作成は完了です。

```
-rw-r--r--. 1 root root 4391 MM DD hh:mm <サーバ証明書と秘密鍵名>.crt
```

6. サーバ証明書用秘密鍵が作成されたことを確認するため、以下のコマンドを実行します。

サーバ証明書用秘密鍵が作成されたことを確認するには以下のコマンドを実行します。

```
# ls -l /root/easyrsa/pki/private/
```

以下のように、作成した秘密鍵が出力された場合、作成は完了です。

```
-rw-r--r--. 1 root root 1704 MM DD hh:mm <サーバ証明書と秘密鍵名>.key
```

7. DH鍵を作成します。

```
# ./easyrsa gen-dh
```

以下のようなメッセージが表示されます。

```
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
.....
+.....
+.....
+...+.....+.....+.....+.....+.....
+.....+.....
          .
          .
.....+++++
```

```
DH parameters of size 2048 created at /root/EasyRSA-3.0.1/pki/dh.pem
```

8. DH鍵が作成されたことを確認します。

```
# ls -l ./root/easyrsa/pki/
```

以下のように、作成したDH鍵が出力された場合、作成は完了です。

```
-rw-r--r--. 1 root root 424 MM DD hh:mm dh.pem
```

9. クライアント秘密鍵と証明書を作成します。



警告

SSL-VPN接続で使用するクライアント証明書は必ずユーザー毎に一意的な証明書を作成してください。
複数のユーザーが1つの証明書を利用して同時にSSL-VPN接続を行うことはできません。

```
# ./easyrsa build-client-full <クライアント秘密鍵と証明書名 (任意)> nopass
```

以下のようなメッセージが表示されるため、指示に従い対話式に進めます。

```
Generating a 2048 bit RSA private key
.....
++++.++++
writing new private key to '/root/EasyRSA-3.0.1/pki/private/bps021.key.a5TcULwXAN'
-----
Using configuration from /root/EasyRSA-3.0.1/openssl-1.0.cnf
Enter pass phrase for /root/EasyRSA-3.0.1/pki/private/ca.key: <CA証明書用パスフレーズ (再入力)>
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName          :PRINTABLE:'<クライアント秘密鍵と証明書名 (任意)>'
Certificate is to be certified until MM DD hh:mm:ss YYYY GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
```

10. 以下でクライアント証明書が作成されたことを確認します。

```
# ls -l ./root/easyrsa/pki/issued/
```

以下のように、作成したクライアント証明書が出力された場合、作成は完了です。

```
-rw-----. 1 root root 1704 MM DD hh:mm <クライアント秘密鍵と証明書名>.crt
```

11. 以下でクライアント秘密鍵が作成されたことを確認します。

```
# ls -l ./root/easyrsa/pki/private/
```

以下のように、作成したクライアント秘密鍵が出力された場合、作成は完了です。

```
-rw-----. 1 root root 1704 MM DD hh:mm <クライアント秘密鍵と証明書名>.key
```



作成した証明書を必要に応じて必要なクライアントPCに転送します。

注

- CA証明書
API実行環境クライアントPC、SSL-VPNクライアントPC
- サーバ証明書と秘密鍵
API実行環境クライアントPC
- DH鍵
API実行環境クライアントPC
- クライアント秘密鍵と証明書
SSL-VPNクライアントPC

3.5.2.2 SSL-VPN用証明書登録(V2サービス/自己署名証明書利用)

SSL-VPN用証明書をIaaS 鍵管理機能に登録する手順を解説します。

1. CA証明書を登録します。

```
$ CA_NAME=ca (固定値)
```



CA_NAMEは上記の固定値を指定してください。

注

固定値が指定されていない場合、SSL-VPN接続の作成時にエラーになります。

```
$ EXPIRATION="<CA証明書の有効期限 (任意)>"
```



<CA証明書の有効期限 (任意)>はYYYY-MM-DDThh:mm:ssの形式で指定します。

注

メタ文字を含むため、ダブルクォーテーションをつけてください。

```
$ CONTENT_TYPE=text/plain
```

2. 次のAPIを実行します。

```
$ curl -X POST -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"name": "$CA_NAME", "expiration": "$EXPIRATION", "payload": "<CA証明書の内容>",  
"payload_content_type": "$CONTENT_TYPE"}'
```



<CA証明書の内容>は以下の手順で設定します。

注

- catでCA証明書の情報を表示します。
- 表示した情報の「-----BEGIN CERTIFICATE-----」から「-----END CERTIFICATE-----」までをコピーします。
- テキストエディタなどで改行コードをLF(\n)に置換します。
- 置換した情報を<CA証明書の内容>として使用します。

※置換した場合の証明書の内容については付録の「4.1.1 証明書登録におけるpayloadの指定方法」を参照して下さい。

以下のように、レスポンスコード201が出力された場合、登録は完了です

```
HTTP/1.1 100 Continue
```

```
HTTP/1.1 201 Created
```

```
X-Fcx-Endpoint-Request: EXECUTED_REQ000256040_201
```

```
Location: http://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/<プロジェクトID>/secrets/<CA証明書ID>
```

```
Content-Type: application/json;charset=UTF-8
```

```
Content-Length: 155
```

```
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

```
{"secret_ref": "https://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/v1/<プロジェクトID>/secrets/<CA証明書ID>"}
```

3. サーバ証明書を登録します。

```
$ SV_CERT_NAME=server_certificate (固定値)
```



SV_CERT_NAMEは上記の固定値を指定してください。

注

固定値が指定されていない場合、SSL-VPN接続の作成時にエラーになります。

```
$ EXPIRATION="<サーバ証明書の有効期限 (任意)>"
```



<サーバ証明書の有効期限(任意)>はYYYY-MM-DDThh:mm:ssの形式で指定します。

注

サーバ証明書の有効期限はデフォルトで証明書作成時の10年後の日付が設定されるため、有効期限は10年以内の日付を指定してください。

メタ文字を含むため、ダブルクォーテーションをつけてください。

```
$ CONTENT_TYPE=text/plain
```

4. 次のAPIを実行します。

```
$ curl -X POST -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets ¥
```

```
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
```

```
-d '{"name": "$SV_CERT_NAME", "expiration": "$EXPIRATION", "payload": "<サーバ証明書の内容>"
```

```
"payload_content_type": "$CONTENT_TYPE" }'
```



<サーバ証明書の内容>は以下の手順で設定します。

注

- catでサーバ証明書の情報を表示します。
- 取得した情報の「-----BEGIN CERTIFICATE-----」から「-----END CERTIFICATE-----」までをコピーします。
- テキストエディタなどで改行コードをLF(\n)に置換します。
- 置換した情報を<サーバ証明書の内容>として使用します。

※置換した場合の証明書の内容については付録の「4.1.1 証明書登録におけるpayloadの指定方法」を参照して下さい。

以下のように、レスポンスコード201が出力された場合、登録は完了です。

```
HTTP/1.1 100 Continue
```

```
HTTP/1.1 201 Created
```

```
X-Fcx-Endpoint-Request: EXECUTED_REQ000225866_201
```

```
Location: http://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/<プロジェクトID>/secrets/<サーバ証明書ID>
```

```
Content-Type: application/json;charset=UTF-8
```

```
Content-Length: 155
```

```
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

```
{ "secret_ref": "https://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/v1/<プロジェクトID>/secrets/<サーバ証明書ID>" }
```

5. サーバ証明書用秘密鍵を登録します。

```
$ SV_KEY_NAME=server_key (固定値)
```



SV_KEY_NAMEは上記の固定値を指定してください。

注

固定値が指定されていない場合、SSL-VPNコネクションの作成時にエラーになります。

```
$ EXPIRATION="<サーバ証明書用秘密鍵の有効期限 (任意)>"
```



<サーバ証明書用秘密鍵の有効期限(任意)>はYYYY-MM-DDThh:mm:ssの形式で指定します。

注

サーバ証明書用秘密鍵の有効期限はデフォルトで秘密鍵作成時の10年後の日付が設定されるため、有効期限は10年以内の日付を指定してください。

メタ文字を含むため、ダブルクォーテーションをつけてください。

```
$ CONTENT_TYPE=text/plain
```

6. 次のAPIを実行します。

```
$ curl -X POST -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets ¥
```

```
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
```

```
-d '{ "name": "'$SV_KEY_NAME'", "expiration": "'$EXPIRATION'",  
"payload": "<サーバ証明書用秘密鍵の内容>", "payload_content_type": "'$CONTENT_TYPE'" }'
```




<サーバ証明書用秘密鍵の内容>は以下の手順で設定します。

注

- catでサーバ証明書用秘密鍵の情報を表示します。
- 取得した情報の「-----BEGIN PRIVATE KEY-----」から「-----END PRIVATE KEY-----」までをコピーします。
- テキストエディタなどで改行コードをLF(\n)に置換します。
- 置換した情報を<サーバ用秘密鍵の内容>として使用します。

※置換した場合の証明書の内容については付録の「4.1.1 証明書登録におけるpayloadの指定方法」を参照して下さい。

以下のように、レスポンスコード201が出力された場合、登録は完了です。

```
HTTP/1.1 100 Continue
```

```
HTTP/1.1 201 Created
```

```
X-Fcx-Endpoint-Request: EXECUTED_REQ000257600_201
```

```
Location: http://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/<プロジェクトID>/secrets/<サーバ証明書用秘密鍵ID>
```

```
Content-Type: application/json;charset=UTF-8
```

```
Content-Length: 155
```

```
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

```
{"secret_ref": "https://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/v1/<プロジェクトID>/secrets/<サーバ証明書用秘密鍵ID>"}
```

7. DH鍵を登録します。

```
$ DH_NAME=dh (固定値)
```



DH_NAMEは上記の固定値を指定してください。

注

固定値が指定されていない場合、SSL-VPN接続の作成時にエラーになります。

```
$ EXPIRATION="<DH鍵の有効期限 (任意)>"
```



<DH鍵の有効期限(任意)>はYYYY-MM-DDThh:mm:ssの形式で指定します。

注

メタ文字を含むため、ダブルクォーテーションをつけてください。

```
$ CONTENT_TYPE=text/plain
```

8. 次のAPIを実行します。

```
$ curl -X POST -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets ¥
```

```
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
```

```
-d '{"name": "$DH_NAME", "expiration": "$EXPIRATION", "payload": "<DH鍵の内容>", "payload_content_type": "$CONTENT_TYPE"}'
```



<DH鍵の内容>は以下の手順で設定します。

注

- catでDH鍵の情報を表示します。
- 取得した情報の「-----BEGIN DH PARAMETERS-----」から「-----END DH PARAMETERS-----」までをコピーします。
- テキストエディタなどで改行コードをLF(\n)に置換します。
- 置換した情報を<DH鍵の内容>として使用します。

※置換した場合の証明書の内容については付録の「4.1.1 証明書登録におけるpayloadの指定方法」を参照して下さい。

以下のように、レスポンスコード201が出力された場合、登録は完了です。

```
HTTP/1.1 100 Continue
```

```
HTTP/1.1 201 Created
```

```
X-Fox-Endpoint-Request: EXECUTED_REQ000229580_201
```

```
Location: http://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/<プロジェクトID>/secrets/<DH鍵ID>
```

```
Content-Type: application/json;charset=UTF-8
```

```
Content-Length: 155
```

```
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

```
{"secret_ref": "https://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/v1/<プロジェクトID>/secrets/<DH鍵ID>"}
```

9. 本手順で登録した証明書を確認するため、以下のAPIを実行します。

```
$ curl -X GET -sS $KEYMANAGEMENT/v1/$PROJECT_ID/secrets ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

以下のように、登録した証明書が出力された場合、作成は完了です。

```
{  
  "total": 4,  
  "secrets": [  
    {  
      "expiration": "<CA証明書の有効期限>",  
      "bit_length": null,  
      "status": "ACTIVE",  
      "secret_ref": "https://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/  
v1/<プロジェクトID>/secrets/<CA証明書ID>",  
      "updated": "YYYY-MM-DDThh:mm:ss.SSS",  
      "name": "ca",  
      "algorithm": null,  
      "created": "YYYY-MM-DDThh:mm:ss.SSS",  
      "content_types": {  
        "default": "text/plain"  
      },  
      "mode": null  
    },  
    {  
      "expiration": "<サーバ証明書の有効期限>",  
      "bit_length": null,  
      "status": "ACTIVE",  
      "secret_ref": "https://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/  
v1/<プロジェクトID>/secrets/<サーバ証明書ID>",  
      "updated": "YYYY-MM-DDThh:mm:ss.SSS",  
      "name": "server_certificate",  
      "algorithm": null,  
      "created": "YYYY-MM-DDThh:mm:ss.SSS",  
      "content_types": {  
        "default": "text/plain"  
      },  
      "mode": null  
    },  
    {  
      "expiration": "<サーバ証明書用秘密鍵の有効期限（任意）>",  
      "bit_length": null,  
      "status": "ACTIVE",  
      "secret_ref": "https://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/  
v1/<プロジェクトID>/secrets/<サーバ証明書用秘密鍵ID>",  
      "updated": "YYYY-MM-DDThh:mm:ss.SSS",  
      "name": "server_key",  
      "algorithm": null,  
      "created": "YYYY-MM-DDThh:mm:ss.SSS",  
      "content_types": {  
        "default": "text/plain"  
      },  
      "mode": null  
    }  
  ]  
}
```

```

{
  "expiration": "<DH鍵の有効期限（任意）>",
  "bit_length": null,
  "status": "ACTIVE",
  "secret_ref": "https://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/v1/<プロジェクトID>/secrets/<DH鍵ID>",
  "updated": "YYYY-MM-DDThh:mm:ss.SSS",
  "name": "dh",
  "algorithm": null,
  "created": "YYYY-MM-DDThh:mm:ss.SSS",
  "content_types": {
    "default": "text/plain"
  },
  "mode": null
}
]
}

```



本手順で登録した証明書の"secret_ref"の値は「SSL-VPNコネクシヨンの作成(V2サービス/自己署名証明書利用)」に使用するため、控えておいてください。

注

3.5.2.3 SSL-VPN用鍵コンテナ作成 (V2サービス/自己署名証明書利用)

SSL-VPN用鍵コンテナを作成する手順を解説します。



鍵コンテナとはCA証明書、サーバ証明書、秘密鍵及びDH鍵をまとめた集合体であり、この後のSSL-VPNコネクシヨン作成時に使用します。この鍵コンテナ作成時には、以下の情報を固定値として指定する必要があります。

注

- type: "generic"
- name: "ca", "server_certificate", "server_key", "dh" の4つ

1. 以下で必要な設定を行います。

```
$ CONTAINER_NAME=<鍵コンテナ名（任意）>
```

```
$ TYPE=generic（固定値）
```

```
$ CA_NAME=ca（固定値）
```

```
$ CA_URL=<CA証明書のsecret_ref>
```

```
$ SV_CERT_NAME=server_certificate（固定値）
```

```
$ SV_CERT_URL=<サーバ証明書のsecret_ref>
```

```
$ SV_KEY_NAME=server_key（固定値）
```

```
$ SV_KEY_URL=<サーバ証明書用秘密鍵のsecret_ref>
```

```
$ DH_NAME=dh（固定値）
```

```
$ DH_URL=<DH鍵のsecret_ref>
```

2. 次のAPIを実行します。

```

$ curl -X POST -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/containers ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{"name": "$CONTAINER_NAME", "type": "$TYPE",
"secret_refs": [{"name": "$CA_NAME", "secret_ref": "$CA_URL"},
{"name": "$SV_CERT_NAME", "secret_ref": "$SV_CERT_URL"},
{"name": "$SV_KEY_NAME", "secret_ref": "$SV_KEY_URL"},
{"name": "$DH_NAME", "secret_ref": "$DH_URL"}]}'

```

以下のようなレスポンスが返ります。

```
HTTP/1.1 201 Created
X-Fcx-Endpoint-Request: EXECUTED_REQ000257862_201
Location: http://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/<プロジェクトID>/containers/<鍵コンテナID>
Content-Type: application/json;charset=UTF-8
Content-Length: 161
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

3. 作成した鍵コンテナを確認するため、以下のAPIを実行します。

```
$ curl -X GET -sS $KEYMANAGEMENT/v1/$PROJECT_ID/containers ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" | jq .
```

以下のように、鍵コンテナに登録した証明書が出力された場合、作成は完了です。

```
{
  "containers": [
    {
      "type": "generic",
      "container_ref": "https://keymanagement.<リージョン識別子>.cloud.global.fujitsu.com/v1/<プロジェクトID>/containers/<鍵コンテナID>",
      "created": "YYYY-MM-DDThh:mm:ss.SSS",
      "secret_refs": [
        {
          "name": "ca",
          "secret_id": "<CA証明書ID>"
        },
        {
          "name": "server_certificate",
          "secret_id": "<サーバ証明書ID>"
        },
        {
          "name": "server_key",
          "secret_id": "<サーバ証明書用秘密鍵ID>"
        },
        {
          "name": "dh",
          "secret_id": "<DH鍵ID>"
        }
      ],
      "name": "<鍵コンテナ名>",
      "updated": "YYYY-MM-DDThh:mm:ss.SSS",
      "status": "ACTIVE"
    }
  ],
  "total": 1
}
```

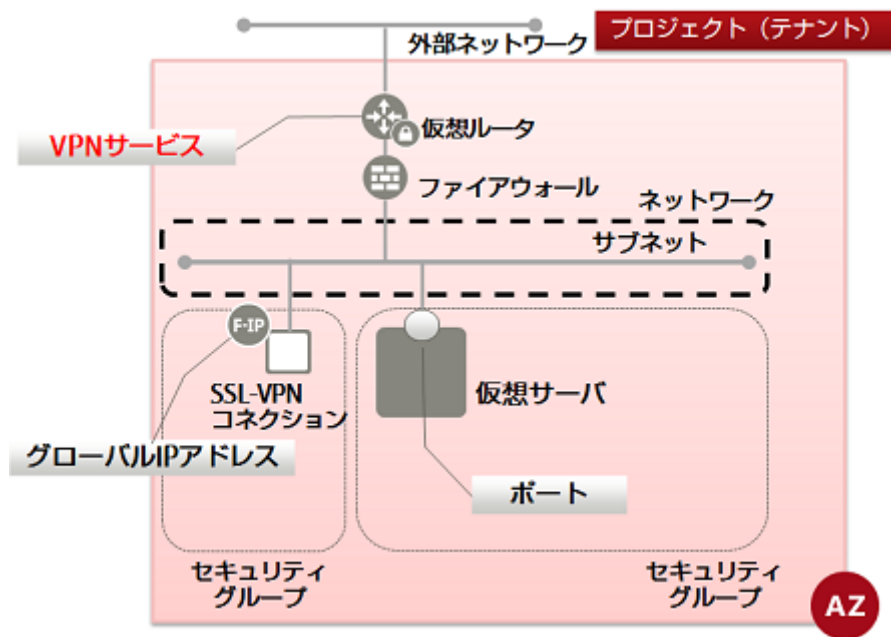


注

鍵コンテナの「container_ref」パラメーターの末尾に記載されている<鍵コンテナID>は「[SSL-VPNコネクションの作成 \(V2サービス/自己署名証明書利用\)](#)」に使用するため控えておいてください。

3.5.2.4 VPNサービスの作成 (V2サービス/自己署名証明書利用)

VPNサービスを作成する手順を解説します。



1. 以下で必要な設定を行います。

```
$ SUBNET_ID=<SSL-VPNで接続したいサブネットID>
```

```
$ ROUTER_ID=<SSL-VPNで利用するルータID>
```

```
$ VPN_SERVICE_NAME=<VPNサービス名 (任意) >
```

```
$ ADMIN_STATE_UP=true
```

```
$ AZ=<アベイラビリティゾーン>
```



警告

SSL-VPN接続 (V2サービス) で接続するサブネットのパラメータ定義が、以下のように指定されている必要があります。

- cidr
ネットワークアドレスのマスク値が「16bit～29bit」の範囲内で作成されている。
例) 192.168.1.0/24
- gateway_ip
VPNサービスで指定する仮想ルータのIPアドレスが指定されている。

2. 次のAPIを実行します。

```
$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"vpnservice": {"subnet_id": "$SUBNET_ID", \
"router_id": "$ROUTER_ID", "name": "$VPN_SERVICE_NAME", \
"admin_state_up": "$ADMIN_STATE_UP", "availability_zone": "$AZ" }}' | jq .
```

以下のレスポンスが返ってきます。

```
{
  "vpnservice": {
    "availability_zone": "<アベイラビリティゾーン>",
    "router_id": "<SSL-VPNで接続するルータID>",
    "status": "PENDING_CREATE",
    "name": "<VPNサービス名>",
    "admin_state_up": true,
    "subnet_id": "<SSL-VPNで接続するサブネットID>",
    "tenant_id": "<SSL-VPNで接続するプロジェクトID>",
    "id": "<VPNサービスID>",
    "description": ""
  }
}
```

```
}  
}
```

3. 作成したVPNサービスを確認するため、以下のAPIを実行します。

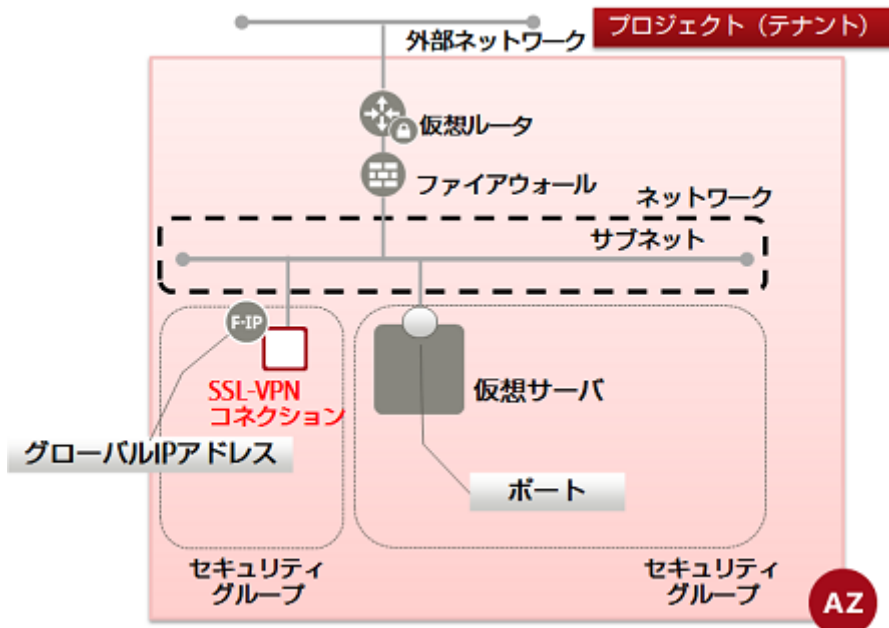
```
$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" \  
-H "Content-Type:application/json" | jq .
```

以下のように、作成したVPNサービスのstatusがPENDING_CREATEになっていることを確認できた場合、作成は完了です。

```
{  
  "vpnservices": [  
    {  
      "availability_zone": "<アベイラビリティゾーン>",  
      "router_id": "<SSL-VPNで接続するルータID>",  
      "status": "PENDING_CREATE",  
      "name": "<VPNサービス名>",  
      "admin_state_up": true,  
      "subnet_id": "<SSL-VPNで接続するサブネットID>",  
      "tenant_id": "<SSL-VPNで接続するプロジェクトID>",  
      "id": "<VPNサービスID>",  
      "description": ""  
    }  
  ]  
}
```

3.5.2.5 SSL-VPNコネクションの作成 (V2サービス/自己署名証明書利用)

SSL-VPNコネクション (V2サービス) を作成する手順を解説します。



1. 以下で必要な設定を行います。

```
$ NAME=<コネクション名 (任意) >
```

```
$ CREDENTIAL_ID=<鍵コンテナID>
```

```
$ VPNSERVICE_ID=<VPNサービスID>
```

```
$ PROTOCOL=tcp
```

```
$ AZ=<アベイラビリティゾーン>
```

```
$ CLIENT_ADDRESS_POOL_CIDR=<VPNトンネルの仮想ネットワークcidr (任意) >
```

CLIENT_ADDRESS_POOL_CIDRはXXX.XXX.XXX.XXX/XX形式でネットワークアドレスの指定を行ってください。

例) 192.168.246.0/24



警告

本サービスで使用するネットワークアドレスやクライアントPCが接続しているローカルネットワークアドレスと競合しないネットワークアドレスを指定してください。



注

本手順を利用してSSL-VPNコネクションを作成した場合、以下の設定が自動的に追加されます。

a. ファイアーウォールルール

SSL-VPN接続専用としてSSL-VPNコネクションへのアクセスを許可するファイアーウォールルール(443/TCP)が自動的に追加されます。

自動的に追加されるファイアーウォールルールを変更したい場合は、SSL-VPNコネクション作成後に変更してください。

b. スタティックルーティング

SSL-VPN接続専用としてSSL-VPNコネクションに設定した「client_address_pool_cidr」へのルーティングが仮想ルータに追加されます。

c. セキュリティグループ

SSL-VPN接続専用として全ての通信許可が設定されたセキュリティグループが作成されます。

SSL-VPNコネクションを作成する際に「security_groups」パラメーターを指定することでユーザーが作成したセキュリティグループを利用することも可能です。

詳細については IaaS APIリファレンス(Network編)を参照してください。

d. グローバルIPアドレス

「floatingips」パラメーターを省略するとSSL-VPNコネクションに割り当てられるグローバルIPアドレスが作成されます。

事前に作成したグローバルIPを利用する場合は「floatingips」パラメーターを指定してください。

詳細については IaaS APIリファレンス(Network編)を参照してください。



注

自動的に追加された設定の他に、SSL-VPNコネクションの「client_address_pool_cidrs」パラメーターに設定した仮想ネットワークから仮想サーバへのアクセス許可をセキュリティグループとファイアーウォールルールに定義する必要があります。

2. 次のAPIを実行します。

```
$ curl -sS $NETWORK/v2.0/vpn/ssl-vpn-v2-connections -X POST ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"ssl_vpn_v2_connection": {"name": "'$NAME'",  
"client_address_pool_cidrs": [ "'$CLIENT_ADDRESS_POOL_CIDR' " ],  
"admin_state_up": true, "credential_id": "'$CREDENTIAL_ID'",  
"vpnservice_id": "'$VPNSERVICE_ID'", "availability_zone": "'$AZ'",  
"protocol": "'$PROTOCOL'" }}' | jq .
```

以下のレスポンスが返ってきます。

```
{  
  "ssl_vpn_v2_connections": [  
    {  
      "access_points": [  
        {  
          "floatingip": null,  
          "client_address_pool_cidr": "<VPNトンネルの仮想ネットワークcidr>",  
          "internal_gateway": "<SSL-VPNコネクションのプライベートIPアドレス>",  
          "external_address": "<SSL-VPNのグローバルIPアドレス>"  
        }  
      ]  
    }  
  ]  
}
```

```

    ],
    "security_groups": [
      "＜SSL-VPN接続専用のセキュリティグループ＞"
    ],
    "protocol": "tcp",
    "availability_zone": "＜アベイラビリティゾーン＞",
    "tenant_id": "＜プロジェクトID＞",
    "name": "＜SSL-VPNコネクション名＞",
    "admin_state_up": true,
    "client_address_pool_cidrs": [
      "＜VPNトンネルの仮想ネットワークcidr＞"
    ],
    "credential_id": "＜鍵コンテナID＞",
    "vpnservice_id": "＜VPNサービス名＞",
    "id": "＜SSL-VPNコネクションID＞",
    "status": "PENDING_CREATE"
  }
]
}

```

3. 作成したSSL-VPNコネクションの状態を確認するため、以下のAPIを実行します。

```

$ curl -sS $NETWORK/v2.0/vpn/ssl-vpn-v2-connections -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .

```

以下のように、SSL-VPNコネクションのstatusがACTIVEになっていることを確認します。

```

{
  "ssl_vpn_v2_connections": [
    {
      "access_points": [
        {
          "floatingip": null,
          "client_address_pool_cidr": "＜VPNトンネルの仮想ネットワークcidr＞",
          "internal_gateway": "＜SSL-VPNコネクションのプライベートIPアドレス＞",
          "external_address": "＜SSL-VPNのグローバルIPアドレス＞"
        }
      ],
      "security_groups": [
        "＜SSL-VPN接続専用のセキュリティグループ＞"
      ],
      "protocol": "tcp",
      "availability_zone": "＜アベイラビリティゾーン＞",
      "tenant_id": "＜プロジェクトID＞",
      "name": "＜SSL-VPNコネクション名＞",
      "admin_state_up": true,
      "client_address_pool_cidrs": [
        "＜VPNトンネルの仮想ネットワークcidr＞"
      ],
      "credential_id": "＜鍵コンテナID＞",
      "vpnservice_id": "＜VPNサービス名＞",
      "id": "＜SSL-VPNコネクションID＞",
      "status": "ACTIVE"
    }
  ]
}

```

作成直後は、statusがPENDING_CREATEのままとなっている場合があります。その場合は時間を置いてから、再度、確認APIを実行してください。

4. VPNサービスの状態を確認するため、以下のAPIを実行します。

```

$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" ¥
-H "Content-Type:application/json" | jq .

```

以下のように、VPNサービスのstatusがACTIVEになっていることを確認できた場合、作成は完了です。


```
{
  "vpnservices": [
    {
      "availability_zone": "<アベイラビリティゾーン>",
      "router_id": "<SSL-VPNで接続されたルータID>",
      "status": "ACTIVE",
      "name": "<VPNサービス名>",
      "admin_state_up": true,
      "subnet_id": "<SSL-VPNで接続するサブネットID>",
      "tenant_id": "<SSL-VPNで接続するプロジェクトID>",
      "id": "<VPNサービスID>",
      "description": ""
    }
  ]
}
```

3.5.2.6 VPNクライアントとの接続設定 (V2サービス/自己署名証明書利用)

VPNクライアントとの接続設定 (V2サービス) を行う手順を解説します。



注

SSL-VPN接続 (V2サービス/自己署名証明書利用) の場合、CA証明書、サーバ証明書、クライアント証明書、秘密鍵の発行と管理は、お客様ご自身で行ってください。本サービスでは、自己署名証明書の失効を確認しません。



注

本手順ではSSL-VPNクライアントPC環境として、以下の環境を使用して確認しています。

- OS : Windows 7 Professional 64bit 日本語版
- VPN クライアント : OpenVPN 2.3.12

1. OpenVPNのインストールを行います。

<https://www.openvpn.jp/download/> から、インストーラをダウンロードして、インストールを実行します。

詳細なインストール方法については「機能説明書」の付録「[SSL-VPNクライアントのセットアップ \(Windows編\)](#)」を参照してください。

2. クライアント設定ファイルの編集を行います。

インストール先のsample-configフォルダにあるclient.ovpnをconfigフォルダにコピーします。

[SSL-VPN用証明書作成 \(V2サービス/自己署名証明書利用\)](#) で作成したCA証明書とクライアント証明書/秘密鍵をconfig フォルダに保存しておく必要があります。

configフォルダのclient.ovpnをテキストエディタで開き、以下の通り編集します。

- protoで始まる行

```
proto top
```

- remoteで始まる行

```
remote [接続先サーバアドレス (SSL-VPN ConnectionリソースのグローバルIPアドレス)] [接続先ポート ( 443 )]
```

- caで始まる行

```
ca <CA証明書名>
```

- certで始まる行

```
cert <クライアント証明書名>
```

- keyで始まる行

```
key <クライアント秘密鍵名>
```

- comp-lzoで始まる行

```
#comp-lzo
```

- cipherで始まる行

```
cipher AES-128-CBC
```

- http-proxyで始まる行 (HTTPプロキシサーバ経由で接続する場合に指定します)

```
http-proxy <HTTPプロキシサーバアドレス> <HTTPプロキシのポート番号> stdin basic
```

stdin: HTTPプロキシサーバ接続時に、ユーザー名とパスワードの入力を要求されます。

basic: 認証方式は基本認証となります。



例:

注

```
proto tcp
remote xxx.xxx.xxx.xxx 443
ca ca.crt
cert client.crt
key client.key
#comp-lzo
cipher AES-128-CBC
http-proxy xxx.xxx.xxx.xxx 8080 stdin basic
```

3. OpenVPNクライアントを起動します。
OpenVPNクライアントを右クリックし、[管理者として実行]を選択し、管理者権限で起動します。
4. SSL-VPN接続を行います。



注

SSL-VPN接続実施後に仮想サーバに接続するには、以下にVPNトンネルのネットワークアドレスから接続する仮想サーバへのアクセス許可の設定を実施する必要があります。

- SSL-VPN機能を設定した仮想ルータのファイアーウォール
- 仮想サーバに割り当てられたセキュリティグループ

PC端末の起動タスクトレイ上のOpenVPNアイコンを右クリックし、[接続]メニューをクリックします。

タスクトレイ上のOpenVPNアイコンがグリーンに切り替わることが確認できたら、SSL-VPNの接続は完了です。

SSL-VPNクライアントPCから仮想サーバのプライベートIPアドレスを指定することで接続が可能となります。

※SSL-VPN接続を切断したい場合

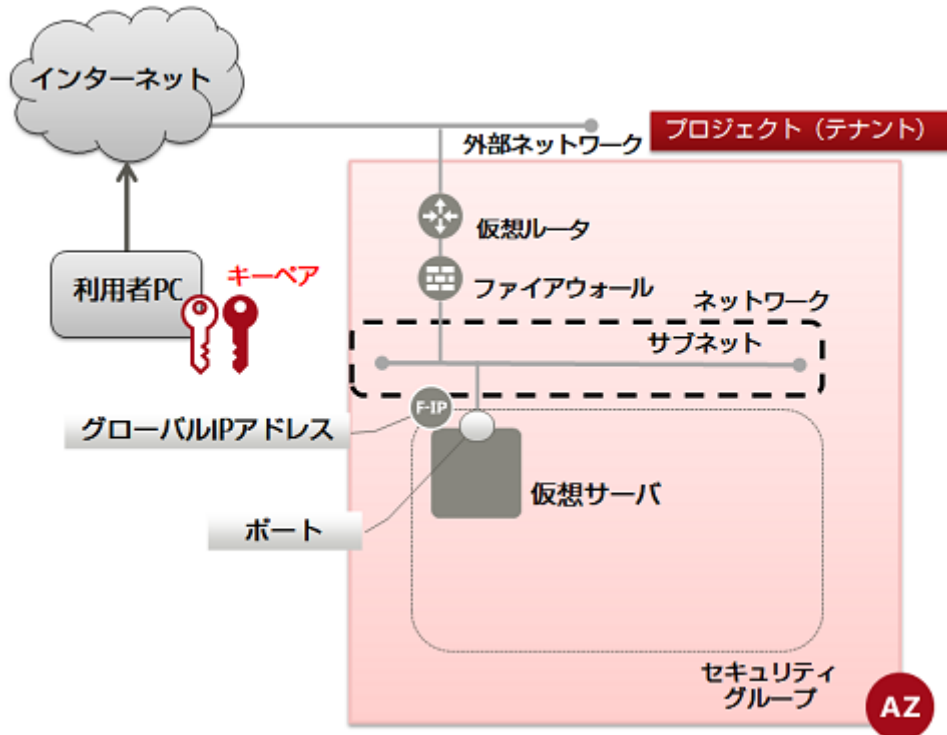
切断するときはタスクトレイ上のOpenVPNアイコンを右クリックし、[切断]をクリックします。

タスクトレイ上のOpenVPNアイコンがグレーに切り替わることを確認できたら、切断完了です。

3.6 仮想サーバ作成

3.6.1 キーペア作成

仮想サーバへSSHを使用してログインする場合に使用するキーペアを作成し、確認する手順を解説します。



キーペアを登録すると、SSH認証用の鍵ファイル(*.pem)を取得することができます。このキーペアを用いて仮想サーバにログインできます。



鍵ファイルの管理には十分ご注意ください。

重要

1. 以下の設定をします。

```
$ KEYPAIR=<キーペア名 (任意)>
```

```
$ AZ=<作成先アベイラビリティゾーン名>
```

2. 次のAPIを実行します。

```
$ curl -sS $COMPUTE/v2/$PROJECT_ID/os-keypairs -X POST ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" ¥  
-d '{"keypair": {"name": "'$KEYPAIR'", "availability_zone": "'$AZ'"}}' | ¥  
jq -r .keypair.private_key > $KEYPAIR.pem
```

3. 作成した鍵ファイルを有効にするために次のコマンドを実行します。

```
$ chmod 600 $KEYPAIR.pem
```

4. 鍵ファイルの中身を確認するために次のコマンドを実行します。

```
cat $KEYPAIR.pem
```

以下のレスポンスが返ってきます。

```
-----BEGIN RSA PRIVATE KEY-----  
~~省略~~  
-----END RSA PRIVATE KEY-----
```

5. 作成したキーペアを確認するため、以下のAPIを実行します。

```
$ curl -X GET -Ss $COMPUTE/v2/$PROJECT_ID/os-keypairs ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

以下のように、設定したキーペア名を含んだリストが出力された場合、作成は完了です。

```
{  
  "keypairs": [  
    ...  
    {  
      "keypair": {  
        "public_key": "ssh-rsa  
        ~~内容省略~~  
        == Generated by Nova¥n",  
        "name": "キーペア名",  
        "fingerprint": "64:96:18:f9:f2:96:e9:7d:f2:b3:dd:ee:bc:45:eb:a0"  
      }  
    },  
    ...  
  ]  
}
```

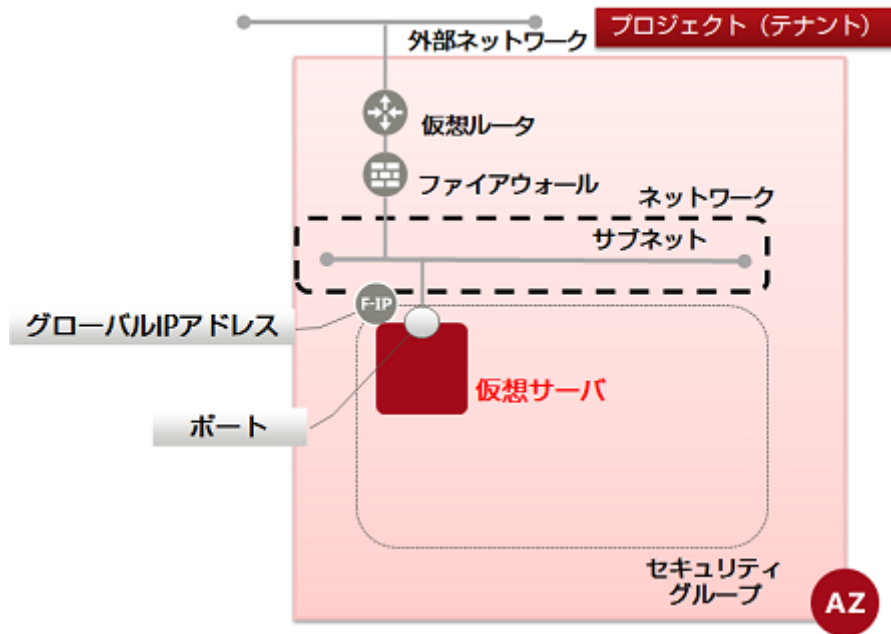


キーペアは生成するだけでなく、外部で作成したキーペアをインポートすることもできます。

ヒント

3.6.2 仮想サーバ作成準備(仮想サーバイメージの取得)

OS提供サービスで提供している仮想サーバのOSイメージの一覧を取得する手順を解説します。



仮想サーバのOSとして以下のラインナップから選択します。仮想サーバ配備時に、使用したいOSイメージを選択してください。

IaaSが提供しているOSイメージについては「機能説明書」の「OS提供サービス」を参照してください。

一覧を表示するには、次のAPIを実行します。

```
$ curl -sS $COMPUTE/v2/$PROJECT_ID/images/detail -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.images[] | .name, .id, .status'
```

以下のようなフォーマットでレスポンスが返ってきます。

```
"<イメージ名>"
"イメージID"
"イメージのステータス"
```

レスポンスの例は以下のようになります。

```
"Ubuntu Server 14.04 LTS (English) 01"
"ffa17298-537d-40b2-a848-0a4d22b49df5"
"ACTIVE"
"FJK5-TemplateBuilder-V02"
"14117885-4104-45a1-9bcd-6dc572e9ee5f"
"ACTIVE"
"Windows Server 2012 SE 64bit (Japanese) 01"
"30718484-1002-43bf-9cf5-f1777c6ed4cb"
"ACTIVE"
"Windows Server 2012 R2 SE 64bit (Japanese) 03"
"5ab16551-c229-4611-834b-a16e074c187e"
"ACTIVE"
"Red Hat Enterprise Linux 6.5 64bit (English) 02"
"db9766f0-c95c-4f1c-bb29-304a90405e3e"
"ACTIVE"
"Windows Server 2008 R2 EE 64bit SP1 (Japanese) 02"
"dcbd4261-e5d1-4d49-9e4f-7434a14dbf4e"
"ACTIVE"
"CentOS 6.5 64bit (English) 03"
"839c1db6-738c-4e2b-9a1d-c14977564203"
"ACTIVE"
"Windows Server 2008 R2 SE 64bit SP1 (Japanese) 02"
"0e2bd896-0ede-4e00-bf71-248ef92c2202"
"ACTIVE"
```

仮想サーバを作成する場合、イメージIDを指定します。

3.6.3 仮想サーバ作成準備(仮想サーバタイプ・フレーバー一覧の取得)

提供している仮想サーバタイプ(フレーバー)の一覧を取得する手順を解説します。

提供している仮想サーバタイプ(フレーバー)は「機能説明書」の「[仮想サーバ](#)」を参照してください。
一覧を表示するには、次のAPIを実行します。

```
$ curl -sS $COMPUTE/v2/$PROJECT_ID/flavors/detail -X GET ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.flavors[] | {name: .name, id: .id}'
```

以下のようなフォーマットでレスポンスが返ってきます。

```
{  
  "name": "<フレーバー名>",&br/>  "id": "<フレーバーId>"  
}
```

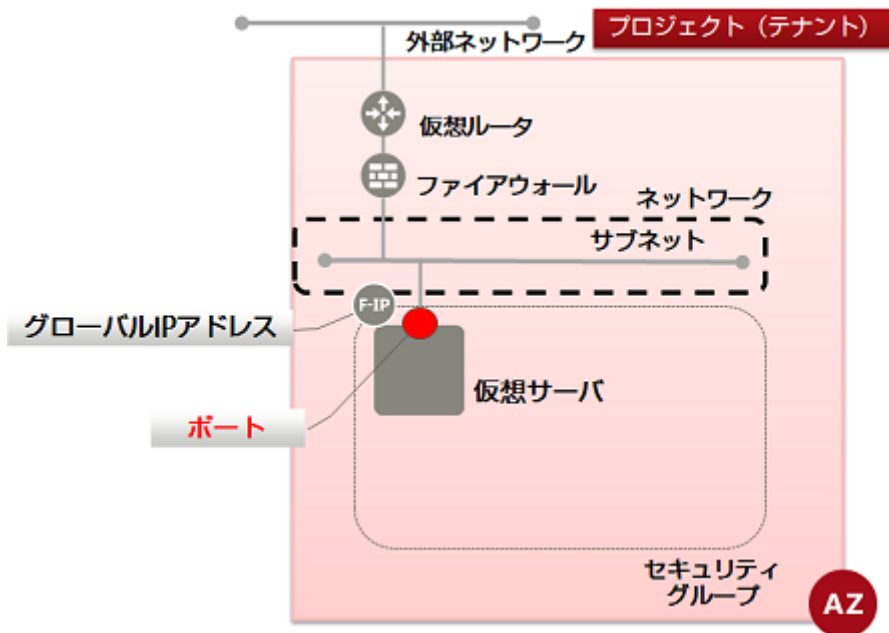
レスポンスの例は以下のようになります。

```
{  
  "name": "S-1",  
  "id": "1101"  
}  
{  
  "name": "S-2",  
  "id": "1102"  
}  
{  
  "name": "S-4",  
  "id": "1103"  
}  
{  
  "name": "S-8",  
  "id": "1104"  
}  
{  
  "name": "S-16",  
  "id": "1105"  
}  
{  
  "name": "M-1",  
  "id": "1201"  
}  
{  
  "name": "M-2",  
  "id": "1202"  
}  
{  
  "name": "M-4",  
  "id": "1203"  
}  
{  
  "name": "M-8",  
  "id": "1204"  
}  
{  
  "name": "M-16",  
  "id": "1205"  
}
```

仮想サーバを作成する場合、フレーバーIDを指定します。

3.6.4 ポート作成

仮想サーバなどのリソースをネットワークに接続するため、IPアドレスとの関連付けを行うポート(ネットワークインターフェース)を作成し、確認するまでの手順を解説します。



以下のリソースを作成する際にサブネットだけを指定した場合、システムが自動的にポートを作成して割り当てます。

• 仮想サーバ



注

DHCPによる自動割当てではなくIPアドレスを指定して配備したい場合は、そのIPアドレスでポートを事前に作成したあと、仮想サーバに割り当ててください。



ヒント

仮想サーバにはポートを複数追加できます。

• 仮想ルータ



注

デフォルトゲートウェイ(x.x.x.1)に作成する場合だけ、自動でポートを割り当てます。すでにx.x.x.1のアドレスで仮想ルータが接続されているネットワークに仮想ルータを追加する場合は、手動でポートを構成する必要があります。

1. 以下で必要な設定を行います。

```
$ PORT_NAME=<ポート名 (任意) >
```

```
$ NETWORK_ID=<ネットワークID>
```

```
$ SUBNET_ID=<サブネットID>
```

```
$ FIXED_IP_ADDRESS=<指定するIPアドレス「XXX.XXX.XXX.XXX」 (サブネット作成時に指定した範囲内) で指定>
```

```
$ SG_ID=<セキュリティグループID>
```

2. 次のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/ports -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" ¥  
-H "Content-Type: application/json" -d '{"port": {"network_id": "$NETWORK_ID",  
"name": "$PORT_NAME", "availability_zone": "$AZ",
```

```
"fixed_ips": [{"subnet_id": "$SUBNET_ID", "ip_address": "$FIXED_IP_ADDRESS"}],  
"security_groups": ["$SG_ID"]}] | jq .
```

以下のレスポンスが返ってきます。

```
{  
  "port": {  
    "status": "<ステータス>",  
    "name": "<ポート名>",  
    "allowed_address_pairs": [],  
    "admin_state_up": true,  
    "network_id": "<ネットワークID>",  
    "tenant_id": "<プロジェクトID>",  
    "binding:vnic_type": "normal",  
    "device_owner": "",  
    "mac_address": "<MACアドレス>",  
    "fixed_ips": [  
      {  
        "subnet_id": "<サブネットID>",  
        "ip_address": "<指定するIPアドレス「XXX.XXX.XXX.XXX」>"  
      }  
    ],  
    "id": "<ポートID>",  
    "security_groups": [  
      null  
    ],  
    "device_id": "",  
    "availability_zone": "<アベイラビリティゾーン>"  
  }  
}
```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

3. 作成したポートを確認するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/ports -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.ports[] | .name, .status, .id, .fixed_ips[]'
```

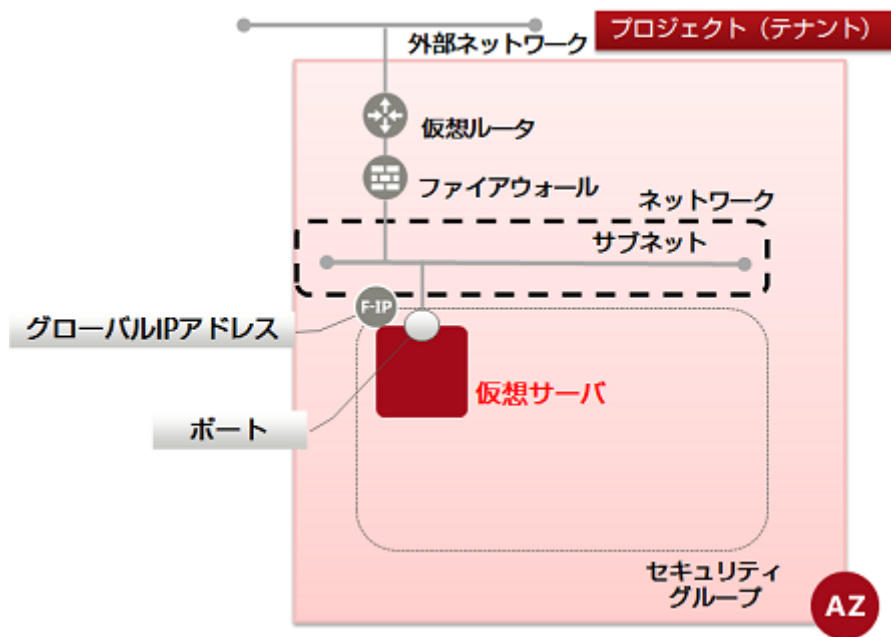
以下のように、設定したポート名を含んだリストが出力された場合、作成は完了です。

```
...  
""  
"<ステータス>"  
"<ポートID>"  
{  
  "subnet_id": "<サブネットID>",  
  "ip_address": "<プライベートIPアドレス>"  
}  
...
```

3.6.5 仮想サーバ作成 (CentOS・ポート指定)

ポート(プライベートIPアドレス)を指定して、仮想サーバ(CentOS)を作成し、確認するまでの手順を解説します。

ポート(プライベートIPアドレス)をDHCP取得する場合、「仮想サーバ作成 (CentOS・DHCP取得)」を参照してください。



1. 以下の設定を行います。

```
$ VM_NAME=<仮想サーバ名 (任意)>
```

```
$ FLAVOR_ID=<仮想サーバのスペック (flavorID一覧表示で確認)>
```

```
$ VOL_SIZE=30
```

```
$ DEVICE_NAME=<ブロックストレージパス (「/dev/vda」の形式で指定)>
```

```
$ IMAGE_REF_ID=839c1db6-738c-4e2b-9a1d-c14977564203<CentOSのイメージID>
```

```
$ SOURCE=<ブロックストレージタイプ>
```

```
$ DESTINATION=<接続先ボリューム (volume指定)>
```

```
$ ISDELETE=<0:仮想サーバ作成時に作成したボリュームを、仮想サーバ削除時に一緒に削除しない、  
1:仮想サーバ作成時に作成したボリュームを仮想サーバ削除時に一緒に削除する>
```

```
$ KEYNAME=<キーペア名>
```

```
$ INSTANCE_MAX=<サーバの最大数>
```

```
$ INSTANCE_MIN=<サーバの最小数>
```

```
$ PORT_ID=<指定するポートID>
```

2. 次のAPIを実行します。

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"server": {"name": "'$VM_NAME'",
"imageRef": "", "flavorRef": "'$FLAVOR_ID'",
"block_device_mapping_v2": [ {"boot_index": "0", "uuid": "'$IMAGE_REF_ID'",
"volume_size": "'$VOL_SIZE'", "device_name": "'$DEVICE_NAME'", "source_type":
"'$SOURCE'",
"destination_type": "'$DESTINATION'", "delete_on_termination": '$ISDELETE'} ] },
"key_name": "'$KEYNAME'", "max_count": '$INSTANCE_MAX', "min_count": '$INSTANCE_MIN',
"networks": [{"port": "'$PORT_ID'"}]}' | jq .
```

以下のレスポンスが返ります。

```
{
```

```

"server": {
  "security_groups": [
    {
      "name": "default"
    }
  ],
  "OS-DCF:diskConfig": "MANUAL",
  "id": "<新規仮想サーバID>",
  "links": [
    {
      "href": "http://10.3.0.201/v2/<プロジェクトID>/servers/<新規仮想サーバID>",
      "rel": "self"
    },
    {
      "href": "http://10.3.0.201/<プロジェクトID>/servers/<新規仮想サーバID>",
      "rel": "bookmark"
    }
  ]
}
}

```

3. 作成した仮想サーバを確認するため、以下のAPIを実行します。

```

$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.servers[] |
{status: .status, network_name: .addresses |
keys, key_name: .key_name, flavor_id: .flavor |
.id, vm_id: .id, security_group: .security_groups[] | .name, name: .name, }'

```

以下のように、設定した仮想サーバ名を含んだリストが出力された場合、作成は完了です。

```

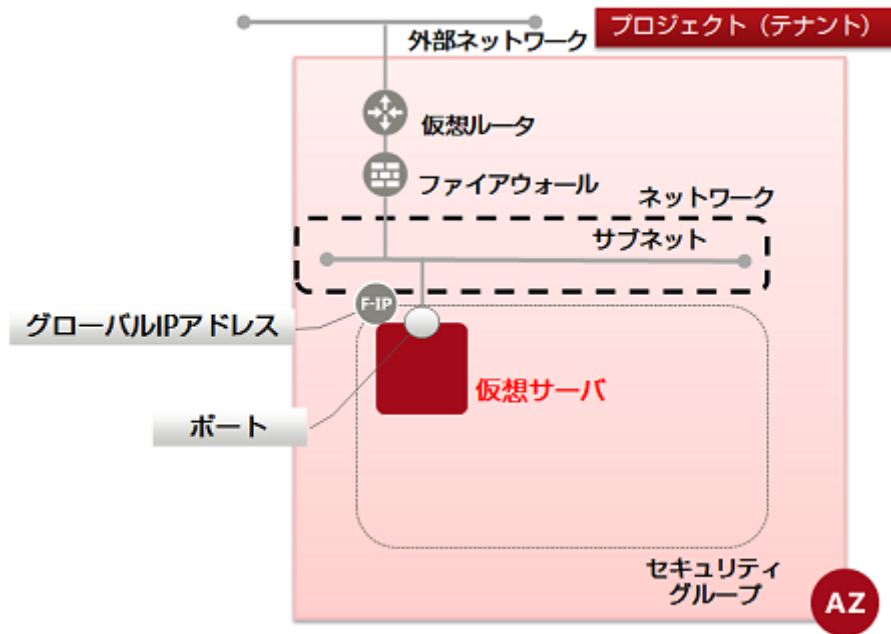
...
{
  "status": "<仮想サーバのステータス>",
  "network_name": [
    "<接続されているネットワーク名>"
  ],
  "key_name": "<キーペア名>",
  "flavor_id": "<フレーバーID>",
  "vm_id": "<仮想サーバID>",
  "security_group": "<セキュリティグループ名>",
  "name": "<仮想サーバ名>"
}
...

```

3.6.6 仮想サーバ作成 (CentOS・DHCP取得)

ポート(プライベートIPアドレス)をDHCP取得して、仮想サーバ(CentOS)を作成し、確認するまでの手順を解説します。

ポート(プライベートIPアドレス)を指定する場合、「仮想サーバ作成 (CentOS・ポート指定)」を参照してください。



1. 以下の設定を行います。

```
$ VM_NAME=<仮想サーバ名 (任意)>
```

```
$ FLAVOR_ID=<仮想サーバのスペック (flavorID一覧表示で確認)>
```

```
$ VOL_SIZE=30
```

```
$ DEVICE_NAME=<ブロックストレージパス (「/dev/vda」の形式で指定)>
```

```
$ SOURCE=<ブロックストレージタイプ>
```

```
$ IMAGE_REF_ID=839c1db6-738c-4e2b-9a1d-c14977564203<CentOSのイメージID>
```

```
$ DESTINATION=<接続先ボリューム (volume指定)>
```

```
$ ISDELETE=<0:仮想サーバ作成時に作成したボリュームを、仮想サーバ削除時に一緒に削除しない、  
1:仮想サーバ作成時に作成したボリュームを仮想サーバ削除時に一緒に削除する>
```

```
$ KEYNAME=<キーペア名>
```

```
$ INSTANCE_MAX=<サーバの最大数>
```

```
$ INSTANCE_MIN=<サーバの最小数>
```

```
$ NETWORK_ID=<接続したいネットワークID>
```

```
$ SG_NAME=<指定するセキュリティグループ名>
```

2. 次のAPIを実行します。

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers -X POST ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"server": {"name": "'$VM_NAME'", "availability_zone": "'$AZ'",  
"imageRef": "", "flavorRef": "'$FLAVOR_ID'",  
"block_device_mapping_v2": [ {"boot_index": "0", "uuid": "'$IMAGE_REF_ID'",  
"volume_size": "'$VOL_SIZE'", "device_name": "'$DEVICE_NAME'", "source_type":  
"'$SOURCE'",  
"destination_type": "'$DESTINATION'", "delete_on_termination": '$ISDELETE'} ] ,  
"key_name": "'$KEYNAME'", "max_count": '$INSTANCE_MAX', "min_count": '$INSTANCE_MIN',  
"networks": [ {"uuid": "'$NETWORK_ID'"} ],  
"security_groups": [ {"name": "'$SG_NAME'"} ] }' | jq .
```

以下のレスポンスが返ります。

```
{
  "server": {
    "security_groups": [
      {
        "name": "<指定したセキュリティグループ名>"
      }
    ],
    "OS-DCF:diskConfig": "MANUAL",
    "id": "<新規仮想サーバID>",
    "links": [
      {
        "href": "http://10.3.0.201/v2/<プロジェクトID>/servers/<新規仮想サーバID>",
        "rel": "self"
      },
      {
        "href": "http://10.3.0.201/<プロジェクトID>/servers/<新規仮想サーバID>",
        "rel": "bookmark"
      }
    ]
  }
}
```

3. 作成した仮想サーバを確認するため、以下のAPIを実行します。

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.servers[] |
{status: .status, network_name: .addresses |
keys, key_name: .key_name, flavor_id: .flavor |
.id, vm_id: .id, security_group: .security_groups[] | .name, name: .name, }'
```

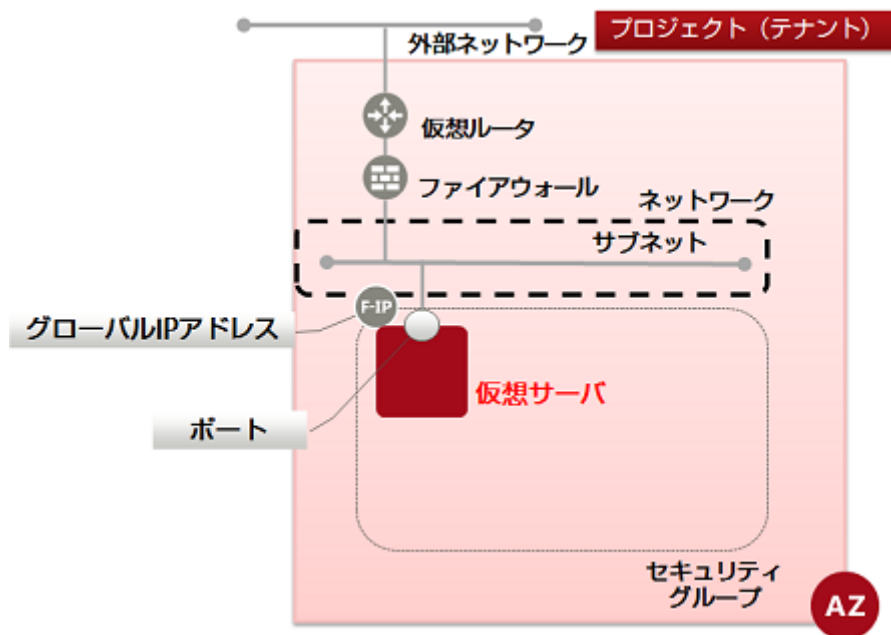
以下のように、設定した仮想サーバ名を含んだリストが出力された場合、作成は完了です。

```
...
{
  "status": "<仮想サーバのステータス>",
  "network_name": [
    "<接続されているネットワーク名>"
  ],
  "key_name": "<キーペア名>",
  "flavor_id": "<フレーバーID>",
  "vm_id": "<仮想サーバID>",
  "security_group": "<設定したセキュリティグループ名>",
  "name": "<仮想サーバ名>"
}
...
```

3.6.7 仮想サーバ作成 (Windows・ポート指定)

ポート(プライベートIPアドレス)を指定して、仮想サーバ(Windows)を作成し、確認するまでの手順を解説します。

ポート(プライベートIPアドレス)をDHCP取得する場合、「仮想サーバ作成 (Windows・DHCP取得)」を参照してください。



次の手順で仮想サーバを作成し、パスワードを入手します。パスワードはリモートデスクトップログインする際に必要になります。

- パスワード暗号化、復号化用キーペアを設定して、仮想サーバを作成
- 暗号化されたパスワードを取得
- キーペアで復号化してパスワードを取得



危険

admin_passをメタデータに指定する形で配備したWindowsOSの仮想サーバは、他のユーザーがパスワードを参照することが出来るため、配備後にパスワードを変更してください。

1. パスワード暗号化、復号化用キーペアを設定して、仮想サーバを作成します。
以下の設定を行います。

```
$ VM_NAME=<仮想サーバ名（任意）>
```

```
$ IMAGE_REF_ID=<WindowsのイメージID>
```

```
$ FLAVOR_ID=<仮想サーバのスペック（flavorID一覧表示で確認）>
```

```
$ VOL_SIZE=80
```

```
$ DEVICE_NAME=<ブロックストレージパス（「/dev/vda」の形式で指定）>
```

```
$ SOURCE=<ブロックストレージタイプ>
```

```
$ DESTINATION=<接続先ボリューム（volume指定）>
```

```
$ ISDELETE=<0:仮想サーバ作成時に作成したボリュームを、仮想サーバ削除時に一緒に削除しない、  
1:仮想サーバ作成時に作成したボリュームを仮想サーバ削除時に一緒に削除する>
```

```
$ KEYNAME=<キーペア名>
```

```
$ INSTANCE_MAX=<サーバの最大数>
```

```
$ INSTANCE_MIN=<サーバの最小数>
```

```
$ PORT_ID=<指定するポートID>
```

2. 次のAPIを実行します。

```
$ curl -sS $COMPUTE/v2/$PROJECT_ID/servers -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"server": {"name": "$VM_NAME",
"imageRef": "", "flavorRef": "$FLAVOR_ID",
"block_device_mapping_v2": [{"boot_index": "0", "uuid": "$IMAGE_REF_ID",
"volume_size": "$VOL_SIZE", "source_type": "$SOURCE",
"destination_type": "$DESTINATION", "delete_on_termination": '$ISDELETE'}]},
"max_count": '$INSTANCE_MAX', "min_count": '$INSTANCE_MIN',
"key_name": "$KEYNAME", "networks": [{"port": "$PORT_ID"}]}' | jq .
```

以下のレスポンスが返ります。

```
{
  "server": {
    "security_groups": [
      {
        "name": "<指定したセキュリティグループ名>"
      }
    ],
    "OS-DCF:diskConfig": "MANUAL",
    "id": "<新規仮想サーバID>",
    "links": [
      {
        "href": "http://10.3.0.201/v2/<プロジェクトID>/servers/<新規仮想サーバID>",
        "rel": "self"
      },
      {
        "href": "http://10.3.0.201/<プロジェクトID>/servers/<新規仮想サーバID>",
        "rel": "bookmark"
      }
    ]
  }
}
```

3. 作成した仮想サーバを確認するため、以下のAPIを実行します。

```
$ curl -sS $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET \
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.servers[] |
{status: .status, network_name: .addresses |
keys, key_name: .key_name, flavor_id: .flavor |
.id, vm_id: .id, security_group: .security_groups[] | .name, name: .name, }'
```

以下のように、設定した仮想サーバ名を含んだリストが出力された場合、作成は完了です。

```
...
{
  "status": "<仮想サーバのステータス>",
  "network_name": [
    "<接続されているネットワーク名>"
  ],
  "key_name": "<キーペア名>",
  "flavor_id": "<フレーバーID>",
  "vm_id": "<仮想サーバID>",
  "security_group": "<設定したセキュリティグループ名>",
  "name": "<仮想サーバ名>"
}
...
```

4. 暗号化されたパスワードを取得します。

仮想サーバを作成してから、パスワード取得ができるようになる間隔は10分です。

以下の設定を行います。

```
$ SERVER_ID=<仮想サーバのID>
```

```
$ PROJECT_ID=<プロジェクトID>
```

5. 次のAPIを実行します。

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-server-password -X GET ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

以下のようなレスポンスになります。

```
{  
  "password": "~~~~~ <暗号化されたパスワード> ~~~~=="  
}
```

6. キーペアで復号化してパスワードを取得します。

暗号化されたパスワードを環境変数に設定します。

```
$ PASSWORD=<取得したパスワード>
```

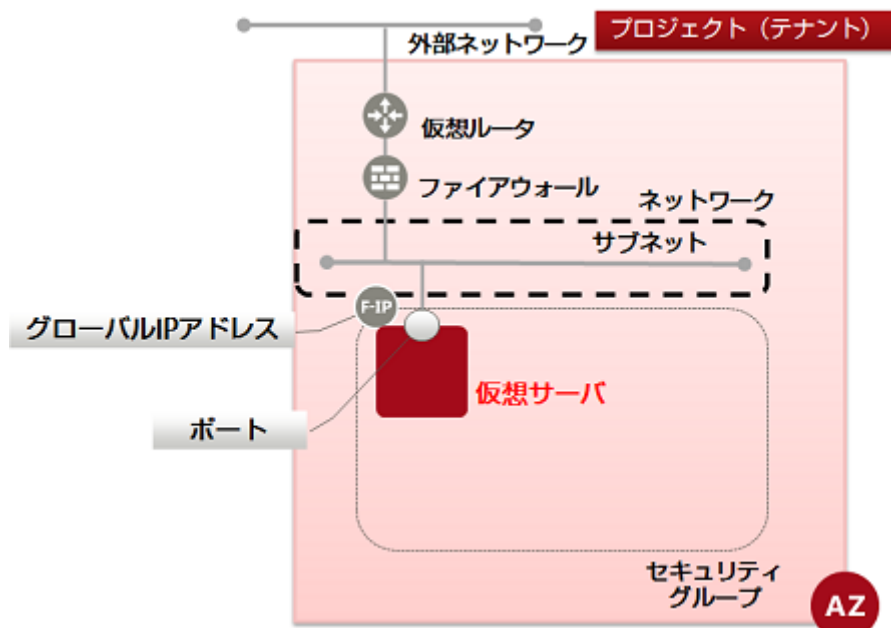
7. 次のコマンドを実行します。

```
$ echo $PASSWORD | openssl base64 -d -A | openssl rsautl -decrypt ¥  
-inkey <指定した鍵ファイルへのパス>
```

3.6.8 仮想サーバ作成 (Windows・DHCP取得)

ポート(プライベートIPアドレス)をDHCP取得して、仮想サーバ(Windows)を作成し、確認するまでの手順を解説します。

ポート(プライベートIPアドレス)を指定する場合、「仮想サーバ作成 (Windows・ポート指定)」を参照してください。



次の手順で仮想サーバを作成し、パスワードを入手します。パスワードはリモートデスクトップログインする際に必要になります。

- パスワード暗号化、復号化用キーペアを設定して、仮想サーバを作成
- 暗号化されたパスワードを取得
- キーペアで復号化してパスワードを取得



admin_passをメタデータに指定する形で配備したWindowsOSの仮想サーバは、他のユーザーがパスワードを参照することが出来るため、配備後にパスワードを変更してください。

1. パスワード暗号化、復号化用キーペアを設定して、仮想サーバを作成します。

以下の設定を行います。

```
$ VM_NAME=<仮想サーバ名（任意）>
```

```
$ IMAGE_REF_ID=<WindowsのイメージID>
```

```
$ FLAVOR_ID=<仮想サーバのスペック（flavor ID一覧表示で確認）>
```

```
$ VOL_SIZE=80
```

```
$ DEVICE_NAME=<ブロックストレージパス（「/dev/vda」の形式で指定）>
```

```
$ SOURCE=<ブロックストレージタイプ>
```

```
$ DESTINATION=<接続先ボリューム（volume指定）>
```

```
$ ISDELETE=<0:仮想サーバ作成時に作成したボリュームを、仮想サーバ削除時に一緒に削除しない、
```

```
1:仮想サーバ作成時に作成したボリュームを仮想サーバ削除時に一緒に削除する>
```

```
$ KEYNAME=<キーペア名>
```

```
$ INSTANCE_MAX=<サーバの最大数>
```

```
$ INSTANCE_MIN=<サーバの最小数>
```

```
$ NETWORK_ID=<接続したいネットワークID>
```

```
$ SG_NAME=<指定するセキュリティグループ名>
```

2. 次のAPIを実行します。

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers -X POST -H "X-Auth-Token: $OS_AUTH_TOKEN" \
-H "Content-Type: application/json" -d '{"server": {"name": "'$VM_NAME'",
"imageRef": "", "flavorRef": "'$FLAVOR_ID'",
"block_device_mapping_v2": [ {"boot_index": "0", "uuid": "'$IMAGE_REF_ID'",
"volume_size": "'$VOL_SIZE'", "source_type": "'$SOURCE'",
"destination_type": "'$DESTINATION'", "delete_on_termination": '$ISDELETE'} ] ,
"max_count": '$INSTANCE_MAX', "min_count": '$INSTANCE_MIN',
"key_name": "'$KEYNAME'", "networks": [{"uuid": "'$NETWORK_ID'"}],
"security_groups": [{"name": "'$SG_NAME'"}]}' | jq .
```

以下のレスポンスが返ります。

```
{
  "server": {
    "security_groups": [
      {
        "name": "<指定したセキュリティグループ名>"
      }
    ],
    "OS-DCF:diskConfig": "MANUAL",
    "id": "<新規仮想サーバID>",
    "links": [
      {
        "href": "http://10.3.0.201/v2/<プロジェクトID>/servers/<新規仮想サーバID>",
        "rel": "self"
      }
    ],
    {
      "href": "http://10.3.0.201/<プロジェクトID>/servers/<新規仮想サーバID>",

```



```

    "rel": "bookmark"
  }
]
}
}

```

3. 作成した仮想サーバを確認するため、以下のAPIを実行します。

```

$ curl -sS $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.servers[] |
{status: .status, network_name: .addresses |
keys, key_name: .key_name, flavor_id: .flavor |
.id, vm_id: .id, security_group: .security_groups[] | .name, name: .name, }'

```

以下のように、設定した仮想サーバ名を含んだリストが出力された場合、作成は完了です。

```

...
{
  "status": "<仮想サーバのステータス>",
  "network_name": [
    "<接続されているネットワーク名>"
  ],
  "key_name": "<キーペア名>",
  "flavor_id": "<フレーバーID>",
  "vm_id": "<仮想サーバID>",
  "security_group": "<設定したセキュリティグループ名>",
  "name": "<仮想サーバ名>"
}
...

```

4. 暗号化されたパスワードを取得します。

仮想サーバを作成してから、パスワード取得ができるようになる間隔は10分です。

以下の設定を行います。

```
$ SERVER_ID=<仮想サーバのID>
```

```
$ PROJECT_ID=<プロジェクトID>
```

5. 次のAPIを実行します。

```

$ curl -sS $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-server-password ¥
-X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .

```

以下のようなレスポンスになります。

```

{
  "password": "~~~~~ <暗号化されたパスワード> ~~~~=="
}

```

6. キーペアで復号化してパスワードを取得します。

暗号化されたパスワードを環境変数に設定します。

```
$ PASSWORD=<取得したパスワード>
```

7. 次のコマンドを実行します。

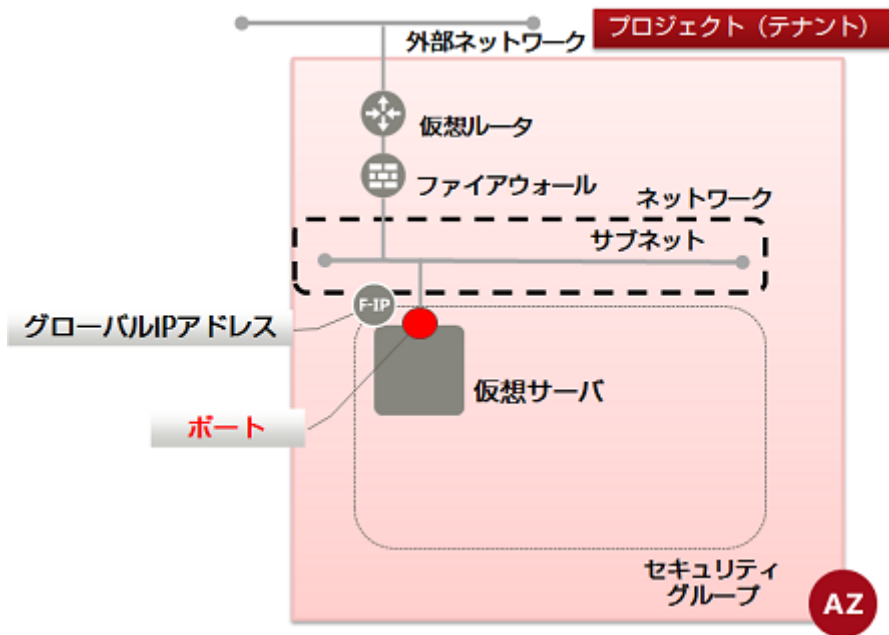
```

$ echo $PASSWORD | openssl base64 -d -A | openssl rsautl -decrypt ¥
-inkey <指定した鍵ファイルへのパス>

```

3.6.9 仮想サーバポートの取得

仮想サーバにグローバルIPアドレスを割り当てるために必要な仮想サーバのポートIDを取得する手順について解説します。



1. ポートを取得したい仮想サーバを指定するために、以下の設定を行います。

```
$ SERVER_ID=<仮想サーバID>
```

2. ポート取得には次のAPIを実行します。

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-interface ¥  
-X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

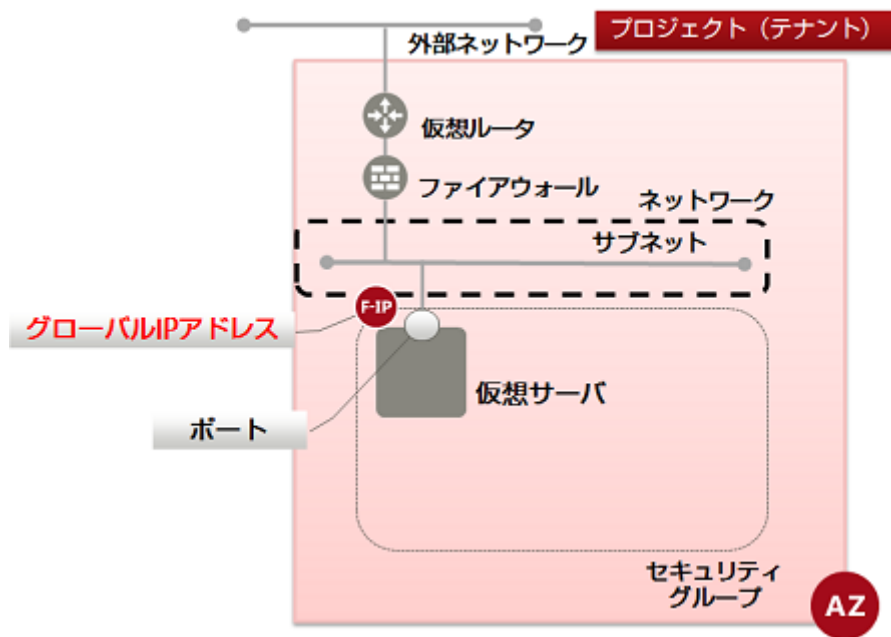
以下のようなフォーマットでレスポンスが返ってきます。

```
{  
  "interfaceAttachments": [  
    {  
      "port_state": "<ステータス>",  
      "fixed_ips": [  
        {  
          "subnet_id": "<仮想サーバID>",  
          "ip_address": "<プライベートIPアドレス>"  
        }  
      ],  
      "port_id": "<ポートID>",  
      "net_id": "<仮想サーバが接続しているネットワークID>",  
      "mac_addr": "<MACアドレス>"  
    }  
  ]  
}
```

3.6.10 グローバルIPアドレスの取得と仮想サーバへの割り当て

仮想リソースにインターネットからアクセスするためのグローバルIPアドレスを取得し仮想サーバに割り当て、そして確認するまでの手順を解説します。

グローバルIPアドレスは取得と仮想リソース割り当ての2ステップに分かれますが、本項目では1つのAPIで実現します。



1. 以下で必要な設定を行います。

```
$ NETWORK_ID=<グローバルIPアドレスを割り当てたい仮想サーバが存在するアベイラビリティゾーンの外部ネットワークID>
```

```
$ VM_PORT_ID=<仮想サーバID>
```

2. 次のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/floatingips -X POST -H "X-Auth-Token:$OS_AUTH_TOKEN" \
-H "Content-Type:application/json" -d '{"floatingip":\n{"floating_network_id":"$NETWORK_ID", "port_id":"$VM_PORT_ID",\n"availability_zone":"$AZ"}}' | jq .
```

以下のレスポンスが返ってきます。

```
{
  "floatingip": {
    "router_id": "<デフォルトゲートウェイの仮想ルータID>",
    "status": "DOWN",
    "tenant_id": "<プロジェクトID>",
    "floating_network_id": "<外部ネットワークアドレス>",
    "fixed_ip_address": "<プライベートIPアドレス>",
    "floating_ip_address": "<グローバルIPアドレス>",
    "port_id": "<設定したポートのID>",
    "id": "<グローバルIPアドレスのID>",
    "availability_zone": "<アベイラビリティゾーン>"
  }
}
```

アベイラビリティゾーンは、AZ1がjp-east-1a、AZ2がjp-east-1bで表現されます。

3. 作成したグローバルIPアドレスを確認するため、以下のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/floatingips -X GET -H "X-Auth-Token:$OS_AUTH_TOKEN" \
-H "Content-Type:application/json" | jq .
```

以下のように、新規割り当てしたグローバルIPアドレスを含んだリストが出力された場合、作成は完了です。

```
{
  "floatingips": [
    ...
    {

```

```
"router_id": "<デフォルトゲートウェイの仮想ルータID>",
"status": "<ステータス>",
"tenant_id": "<プロジェクトID>",
"floating_network_id": "<外部ネットワークアドレス>",
"fixed_ip_address": "<プライベートIPアドレス>",
"floating_ip_address": "<グローバルIPアドレス>",
"port_id": "<ポートのID>",
"id": "<グローバルIPアドレスのID>",
"availability_zone": "<アベイラビリティゾーン>"
}

...
]
```

3.7 接続確認

3.7.1 仮想サーバ(CentOS)へのログイン

仮想サーバ(CentOS)にログインする手順を解説します。

ログインする前に以下を確認してください。

- ログイン対象仮想サーバに割り当てられたグローバルIPアドレスのステータスが"ACTIVE"であること
- セキュリティグループルールが利用環境に対して正しく設定されていること
- ファイアウォールルール、ポリシーが利用環境に対して正しく設定されていること
- 鍵ファイル(.pem)が作業ディレクトリに存在すること

1. 以下のコマンドを実行します。

```
$ ssh -i <Keypairを指定> k5user@<対象仮想サーバに割り当てられたグローバルIPアドレス>
```

```
The authenticity of host ' <対象仮想サーバに割り当てられたグローバルIPアドレス (対象仮想サーバに割り当てられたグローバルIPアドレス)>' can't be established.  
RSA key fingerprint is <省略>.  
Are you sure you want to continue connecting (yes/no)? <>  
Warning: Permanently added ' <対象仮想サーバに割り当てられたグローバルIPアドレス>' (RSA)  
to the list of known hosts.
```

2. root権限の取得。

本サービスが提供するCentOSにおいて、ログイン後rootの権限を取得するためには、特殊なコマンドが必要です。

```
$ sudo su -
```

3.7.2 仮想サーバ(Windows)へのログイン

仮想サーバ(Windows)にログインする手順を解説します。

ログインする前に以下を確認してください。

- ログイン対象仮想サーバに割り当てられたグローバルIPアドレスのステータスが"ACTIVE"であること
- セキュリティグループルールが利用環境に対して正しく設定されていること
- ファイアウォールルール、ポリシーが利用環境に対して正しく設定されていること

クライアントPCからリモートデスクトップを起動し、ログイン接続を行います。



警告

仮想サーバログイン後、セキュリティの観点から仮想サーバ側で「ファイアウォール」の設定が必要となります。

設定されたことを確認の上、仮想サーバを利用してください。

3.8 セキュリティが高いネットワーク構成のSSL-VPN接続 (V2サービス)

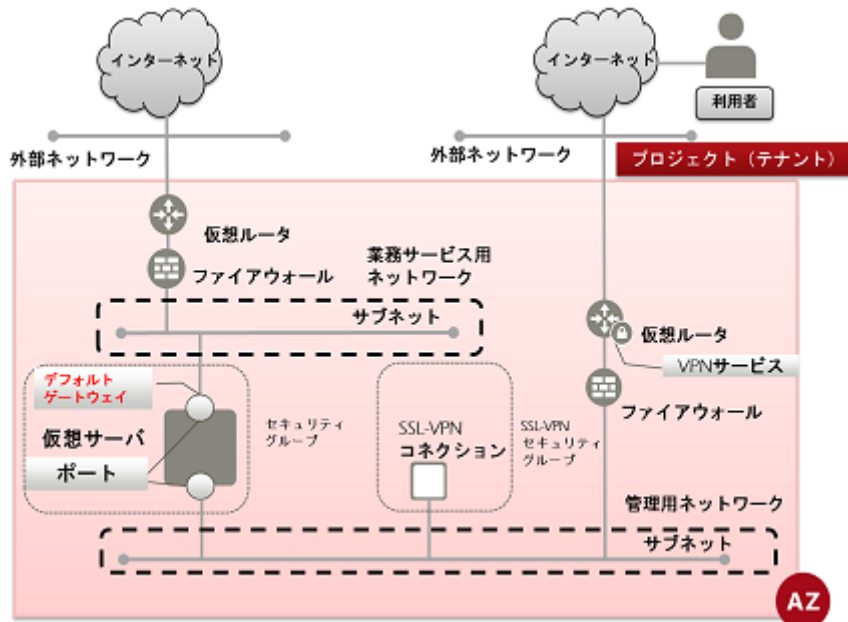
3.8.1 システム構成図

ここでは、業務サービス用と管理用でネットワークを分離する構成(以下、セキュリティが高いネットワーク構成)にてSSL-VPN接続機能を利用する際の構成例について解説します。

セキュリティが高いネットワーク構成にてSSL-VPN接続機能を利用する際の構成例

本構成では仮想サーバに対して管理用ネットワークと業務サービス用ネットワークからアクセスが可能となります。

- 管理用ネットワークはSSL-VPNでのアクセスを行います。
- 業務サービス用ネットワークをデフォルトゲートウェイ(デフォルトルート)として使用します。
- 仮想サーバはLinuxを想定しています。この場合インターフェースファイルの作成と、2つのネットワークのうち、OSから見てどちらをデフォルトゲートウェイ(デフォルトルート)として使用するかを設定する必要があります。

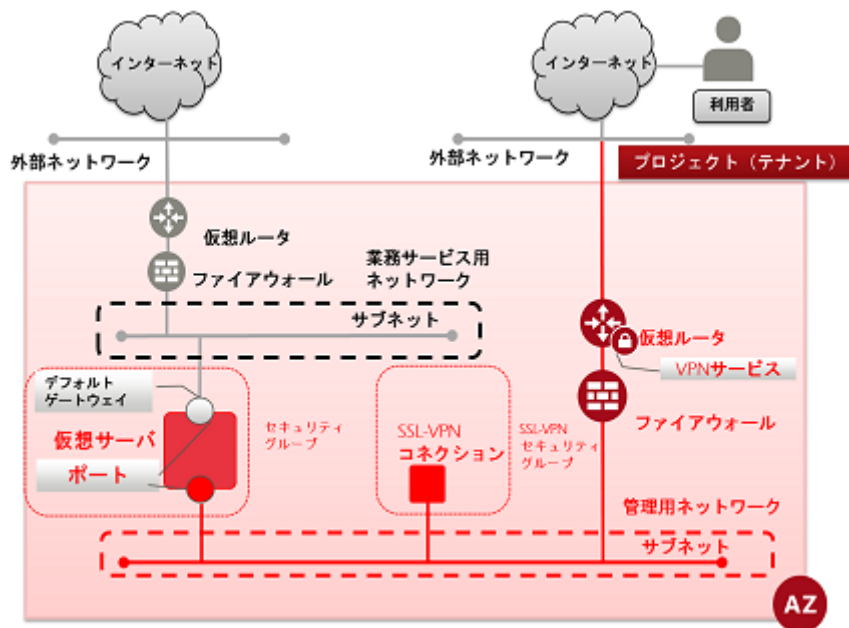


3.8.2 ネットワーク環境の構築

ここでは、管理用、業務サービス用2つのネットワーク環境を構成する手順を解説します。

3.8.2.1 管理用ネットワーク構築

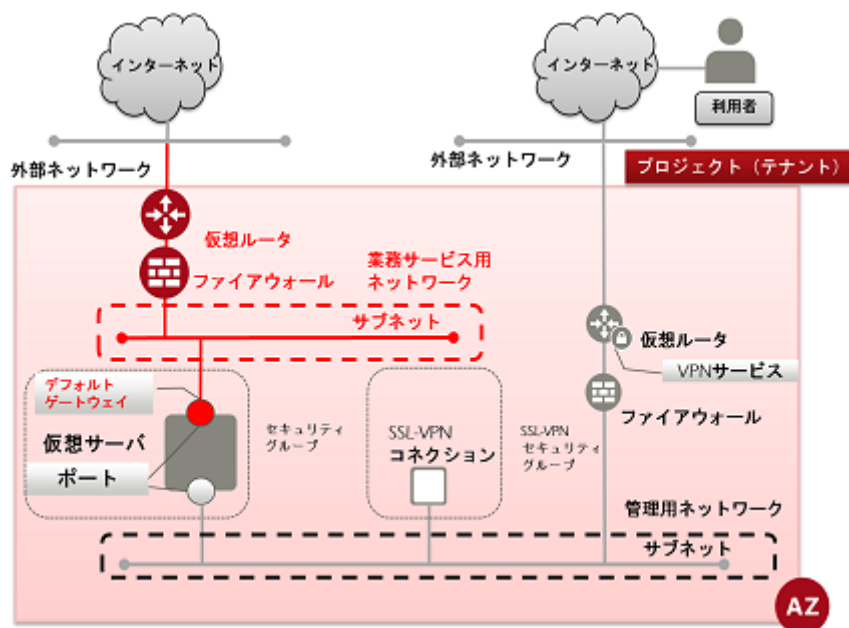
管理用のネットワーク環境の構築は以下の流れで作業を実施します。



1. ネットワーク作成
2. サブネット作成
 - サブネット作成
 - ルーティングの追加
3. 仮想ルータ作成
4. 仮想ルータの情報変更(外部ネットワークにアタッチ)
5. 仮想ルータの情報変更(サブネットにアタッチ)
6. セキュリティグループ作成
7. セキュリティグループルール作成
8. ファイアウォールルール作成
 - ファイアウォールルール(IPアドレス指定)作成
 - ファイアウォールルール(ポート番号指定)作成
 - ファイアウォールルール(ICMP許可)作成
 - ファイアウォールルール(拒否ルール)作成
9. ファイアウォールポリシー作成
10. ファイアウォール作成
11. SSL-VPN接続
 - SSL-VPN接続(V2サービス/クライアント証明書利用)
 - SSL-VPN接続(V2サービス/自己署名証明書利用)
12. キーペア作成
13. ポート作成(仮想サーバをポート指定で作成する場合に実施してください。)
14. 仮想サーバ作成
 - 仮想サーバ作成(CentOS・ポート指定)
 - 仮想サーバ作成(CentOS・DHCP取得)
 - 仮想サーバ作成(Windows・ポート指定)
 - 仮想サーバ作成(Windows・DHCP取得)

3.8.2.2 業務サービス用ネットワーク構築

業務サービス用のネットワーク環境の構築は以下の流れで作業を実施します。



1. ネットワーク作成
2. サブネット作成
 - ・ サブネット作成
3. 仮想ルータ作成
4. 仮想ルータの情報変更(外部ネットワークにアタッチ)
5. 仮想ルータの情報変更(サブネットにアタッチ)
6. セキュリティグループ作成
7. セキュリティグループルール作成
8. ファイアウォールルール作成
 - ・ ファイアウォールルール(IPアドレス指定)作成
 - ・ ファイアウォールルール(ポート番号指定)作成
 - ・ ファイアウォールルール(ICMP許可)作成
 - ・ ファイアウォールルール(拒否ルール)作成
9. ファイアウォールポリシー作成
10. ファイアウォール作成
11. ポート作成
12. ポートのアタッチ
13. インターフェイスファイルの設定 (LinuxOSのみ)

3.8.2.2.1 ポートのアタッチ

事前に業務サービス用ネットワークに作成したポートを仮想サーバにアタッチする手順を解説します。ポートの作成手順については[ポート作成](#)を参照してください。

1. 以下で、必要な設定を行います。

```
$SERVER_ID=<ポートをアタッチする仮想サーバID>
```

```
$PORT_ID=<仮想サーバにアタッチする業務サービス用ネットワークのポートID>
```

2. APIを実行します。

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-interface -X POST ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{"interfaceAttachment": {"port_id": "'$PORT_ID'"}}' | jq .
```


以下のレスポンスが返ってきます。

```
{
  "interfaceAttachment": {
    "mac_addr": "<業務サービス用ネットワークのインターフェースのMACアドレス>",
    "net_id": "<ネットワークID>",
    "port_id": "<仮想サーバにアタッチする業務サービス用ネットワークポートID>",
    "fixed_ips": [
      {
        "ip_address": "<業務サービス用ネットワークのポートのIPアドレス>",
        "subnet_id": "<業務サービス用ネットワークのサブネットID>"
      }
    ],
    "port_state": "DOWN"
  }
}
```

3. 仮想サーバのインターフェースとポートがアタッチされたことを確認するため、以下のAPIを実行します。

```
$ SERVER_ID=<ポートをアタッチした仮想サーバID>
```

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-interface -X GET \
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

以下のレスポンスが返ってくれば、アタッチは完了です。

```
{
  "interfaceAttachments": [
    {
      "mac_addr": "<業務サービス用ネットワークのインターフェースのMACアドレス>",
      "net_id": "<ネットワークID>",
      "port_id": "<仮想サーバにアタッチした業務サービス用ネットワークのポートID>",
      "fixed_ips": [
        {
          "ip_address": "<業務サービス用ネットワークのポートのIPアドレス>",
          "subnet_id": "<業務サービス用ネットワークのサブネットID>"
        }
      ],
      "port_state": "ACTIVE"
    },
    {
      "mac_addr": "<インターフェースのMACアドレス>",
      "net_id": "<ネットワークID>",
      "port_id": "<仮想サーバにアタッチされている管理用ネットワークのポートID>",
      "fixed_ips": [
        {
          "ip_address": "<管理用ネットワークのポートのIPアドレス>",
          "subnet_id": "<管理用ネットワークのサブネットID>"
        }
      ],
      "port_state": "ACTIVE"
    }
  ]
}
```

3.8.2.2.2 インターフェースファイルの設定 (LinuxOSのみ)

LinuxOSの仮想サーバにポートを追加した際に、インターフェースファイルの設定を行います。ここではその手順について解説します。



注

本手順は以下の環境で作成しています。

- OS: CentOS6.8
- 仮想サーバのデフォルトゲートウェイ: 業務サービス用ネットワーク側のポート
- eth0: 管理用ネットワークのポート
- eth1: 業務サービス用ネットワークのポート

1. 管理用ネットワークポートのインターフェースファイル (ifcfg-eth0) をコピーして、仮想サーバに追加した業務サービス用ネットワークポートのインターフェースファイル (ifcfg-eth1) を作成します。また、インターフェースファイルにはどちらをデフォルトゲートウェイとして使用するかを記述します。

- /etc/sysconfig/network-scripts/ifcfg-eth0

DEFROUTE=noを追記します。

```
DEVICE="eth0"  
BOOTPROTO="dhcp"  
IPV6INIT="yes"  
MTU="1500"  
NM_CONTROLLED="yes"  
ONBOOT="yes"  
TYPE="Ethernet"  
UUID="733285fa-1efe-4d9c-a70b-668922168a3f"  
DEFROUTE=no
```

- /etc/sysconfig/network-scripts/ifcfg-eth1

UUIDをコメントアウトし、DEFROUTE=yesを追記します。

```
DEVICE="eth1"  
BOOTPROTO="dhcp"  
IPV6INIT="yes"  
MTU="1500"  
NM_CONTROLLED="yes"  
ONBOOT="yes"  
TYPE="Ethernet"  
#UUID="733285fa-1efe-4d9c-a70b-668922168a3f"  
DEFROUTE=yes
```



重要

DEFROUTEにてインターフェースをデフォルトゲートウェイとして利用するかを指定します。

2. ネットワークの再構成を行います

以下のコマンドを実行し、OSのネットワーク設定を再構成します。

```
/etc/init.d/network restart
```

第 4 章: リソースの削除

トピック:

- ・ 基本構成の削除
- ・ SSL-VPN接続削除 (V2サービス)
- ・ ファイアーウォール削除
- ・ 仮想サーバ削除
- ・ セキュリティグループ削除
- ・ ネットワーク環境の削除

4.1 基本構成の削除

4.1.1 削除の順番について

基本構成の削除の順番について解説します。



セキュリティの観点から外部につながる部分から削除してください。

注

基本的には作成した手順の逆順で、削除を実行します。

4.2 SSL-VPN接続削除 (V2サービス)

4.2.1 SSL-VPN接続の削除 (V2サービス)

SSL-VPN接続 (V2サービス) を削除する手順を解説します。



注

SSL-VPN接続が削除された場合、SSL-VPN接続作成時に自動作成された以下の設定も削除されます。

1. ファイアーウォールルール
SSL-VPN接続専用で作成されたファイアーウォールルールが削除されます。
2. スタティックルーティング
SSL-VPN接続専用で設定されたスタティックルーティングが削除されます。
3. セキュリティグループ
SSL-VPN接続専用で作成されたセキュリティグループが削除されます。
4. グローバルIPアドレス
SSL-VPN接続用に割り当てられたグローバルIPアドレスは削除されます。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_SSLCONNECT_ID=<SSL-VPN接続ID>
```

2. 次のAPIを実行します。

```
$ curl -X DELETE -sS -i $NETWORK/v2.0/vpn/ssl-vpn-v2-connections/$TMP_SSLCONNECT_ID \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json"
```

```
HTTP/1.1 204 No Content
Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT
x-compute-request-id: req-<ID>
Cache-Control: no-cache
X-Request-Id: <ID>
X-Runtime: <runtime>
Keep-Alive: timeout=5, max=100
Content-Type: text/html; charset=UTF-8
Content-Length: 0
```

4.2.2 VPNサービスの削除 (V2サービス)

VPNサービスを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_VPN_ID=<VPNサービスID>
```

2. 次のAPIを実行します。

```
$ curl -X DELETE -sS -i $NETWORK/v2.0/vpn/vpnservices/$TMP_VPN_ID \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json"
```

```
HTTP/1.1 204 No Content
Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

```
x-compute-request-id: req-<ID>
Cache-Control: no-cache
X-Request-Id: <ID>
X-Runtime: <runtime>
Keep-Alive: timeout=5, max=100
Content-Type: text/html; charset=UTF-8
Content-Length: 0
```

4.2.3 SSL-VPN用鍵コンテナの削除 (V2サービス)

SSL-VPN用鍵コンテナを削除する手順を解説します。

[SSL-VPN接続 \(V2サービス/自己署名証明書利用\)](#)を参照してSSL-VPN接続を作成した場合のみ実行する手順です。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_CON_NAME=<鍵コンテナID>
```

2. 次のAPIを実行します。

```
$ curl -X DELETE -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/containers/$TMP_CON_NAME ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request: EXECUTED_REQ<ID>
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

4.2.4 SSL-VPN用証明書の削除 (V2サービス)

SSL-VPN用証明書を削除する手順を解説します。

[SSL-VPN接続 \(V2サービス/自己署名証明書利用\)](#)を参照してSSL-VPN接続を作成した場合のみ実行する手順です。

削除は、基本的に作成した手順の逆順で、削除を実行します。



警告

削除する際は、CA証明書・サーバ証明書用秘密鍵・サーバ証明書・DH鍵で削除したい証明書を個別に削除する必要があります。

1. 以下の設定を行います。

```
$ TMP_CRTKEY_NAME=<DH鍵ID>
```

2. 次のAPIを実行します。

```
$ curl -X DELETE -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets/$TMP_CRTKEY_NAME ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request: EXECUTED_REQ<ID>
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

3. 以下の設定を行います。

```
$ TMP_CRTKEY_NAME=<サーバ証明書用秘密鍵ID>
```

4. 次のAPIを実行します。

```
$ curl -X DELETE -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets/$TMP_CRTKEY_NAME ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request: EXECUTED_REQ<ID>
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

5. 以下の設定を行います。

```
$ TMP_CERTKEY_NAME=<サーバ証明書ID>
```

6. 次のAPIを実行します。

```
$ curl -X DELETE -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets/$TMP_CERTKEY_NAME ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request:EXECUTED_REQ<ID>
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

7. 以下の設定を行います。

```
$ TMP_CERTKEY_NAME=<CA証明書ID>
```

8. 次のAPIを実行します。

```
$ curl -X DELETE -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets/$TMP_CERTKEY_NAME ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request: EXECUTED_REQ<ID>
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

4.3 ファイアウォール削除

4.3.1 ファイアウォールの削除

ファイアウォールを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_FW_ID=<削除対象のファイアウォールのID>
```

2. 次のAPIを実行します。

```
$ curl -i -Ss $NETWORK/v2.0/fw/firewalls/$TMP_FW_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>
```

4.3.2 ファイアウォールポリシーの削除

ファイアウォールポリシーを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_FWP_ID=<削除対象のファイアウォールポリシーのID>
```

2. 次のAPIを実行します。

```
$ curl -i -Ss $NETWORK/v2.0/fw/firewall_policies/$TMP_FWP_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>
```

4.3.3 ファイアウォールルールの削除

ファイアウォールルールを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_FWR_ID=<削除対象のファイアウォールルールのID>
```

2. 次のAPIを実行します。

```
$ curl -X DELETE -i -Ss $NETWORK/v2.0/fw/firewall_rules/$TMP_FWR_ID ¥
```



```
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>
```

4.4 仮想サーバ削除

4.4.1 グローバルIPアドレスの削除

グローバルIPアドレスを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。



注

グローバルIPアドレスを指定した仮想サーバを削除しても、自動的にグローバルIPアドレスは削除されません。

1. 以下の設定を行います。

```
$ TMP_FLOATINGIP_ID=<削除対象のグローバルIPアドレスのID>
```

2. 次のAPIを実行します。

```
$ curl -i -sS $NETWORK/v2.0/floatingips/$TMP_FLOATINGIP_ID -X DELETE ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>  
Keep-Alive: timeout=5, max=100
```

4.4.2 仮想サーバの削除

仮想サーバを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_VM_ID=<削除対象の仮想サーバのID>
```

2. 次のAPIを実行します。

```
$ curl -i -sS $COMPUTE/v2/$PROJECT_ID/servers/$TMP_VM_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>  
Keep-Alive: timeout=5, max=100
```

4.4.3 ポートの削除

ポートを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

ただし、ポートがアタッチされていた仮想サーバが削除された時点で、同時にポートも削除されます。

1. 以下の設定を行います。

```
$ TMP_PORT_ID=<削除対象のポートのID>
```

2. 次のAPIを実行します。

```
$ curl -i -Ss $NETWORK/v2.0/ports/$TMP_PORT_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>  
Keep-Alive: timeout=5, max=100
```

4.4.4 キーペアの削除

キーペアを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_KEYPAIR_NAME=<削除対象のキーペアのID>
```

2. 次のAPIを実行します。

```
$ curl -X DELETE -Ss -i $COMPUTE/v2/$PROJECT_ID/os-keypairs/$TMP_KEYPAIR_NAME ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>  
Keep-Alive: timeout=5, max=100  
Content-Type: text/html; charset=UTF-8  
Content-Length: 0
```

4.5 セキュリティグループ削除

4.5.1 セキュリティグループルールの削除

セキュリティグループルールを削除する手順を解説します。



セキュリティグループルールが含まれるセキュリティグループを削除すると、同時にセキュリティグループルールも削除されます。ここでは、特に1つのセキュリティグループルールを削除する方法を解説しています。

1. 以下の設定を行います。

```
$ TMP_SGR_ID=<削除対象のセキュリティグループルールのID>
```

2. 次のAPIを実行します。

```
$ curl -i -Ss $NETWORK/v2.0/security-group-rules/$TMP_SGR_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>
```

4.5.2 セキュリティグループの削除

セキュリティグループを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_SG_ID=<削除対象のセキュリティグループのID>
```

2. 次のAPIを実行します。

```
$ curl -i -s $NETWORK/v2.0/security-groups/$TMP_SG_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>
```

4.6 ネットワーク環境の削除

4.6.1 仮想ルータとサブネットをデタッチ

仮想ルータに接続されているサブネットをデタッチする手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_ROUTER_ID=<デタッチする仮想ルータのID>
```

```
$ TMP_SUBNET_ID=<デタッチするサブネットのID>
```

2. 次のAPIを実行します。

```
$ curl -Ss $NETWORK/v2.0/routers/$TMP_ROUTER_ID/remove_router_interface -X PUT ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"subnet_id": "$TMP_SUBNET_ID"}' | jq .
```

```
{  
  "subnet_id": "<デタッチ対象のサブネットのID>",  
  "tenant_id": "<プロジェクトID>",  
  "port_id": "<接続されていたポートID>",  
  "id": "<デタッチ対象の仮想ルータID>",  
  "availability_zone": "<アベイラビリティゾーン>"  
}
```

4.6.2 仮想ルータの削除

仮想ルータを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_ROUTER_ID=<削除対象の仮想ルータのID>
```

2. 次のAPIを実行します。

```
$ curl -i -Ss $NETWORK/v2.0/routers/$TMP_ROUTER_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>
```

4.6.3 サブネットの削除

サブネットを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_SUBNET_ID=<削除対象のサブネットのID>
```

2. 次のAPIを実行します。

```
$ curl -i -sS $NETWORK/v2.0/subnets/$TMP_SUBNET_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>
```

4.6.4 ネットワークの削除

ネットワークを削除する手順を解説します。

削除は、基本的に作成した手順の逆順で、削除を実行します。

1. 以下の設定を行います。

```
$ TMP_NETWORK_ID=<削除対象のネットワークのID>
```

2. 次のAPIを実行します。

```
$ curl -i -sS $NETWORK/v2.0/networks/$TMP_NETWORK_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>
```

A: 付録

A.1 証明書登録におけるpayloadの指定方法

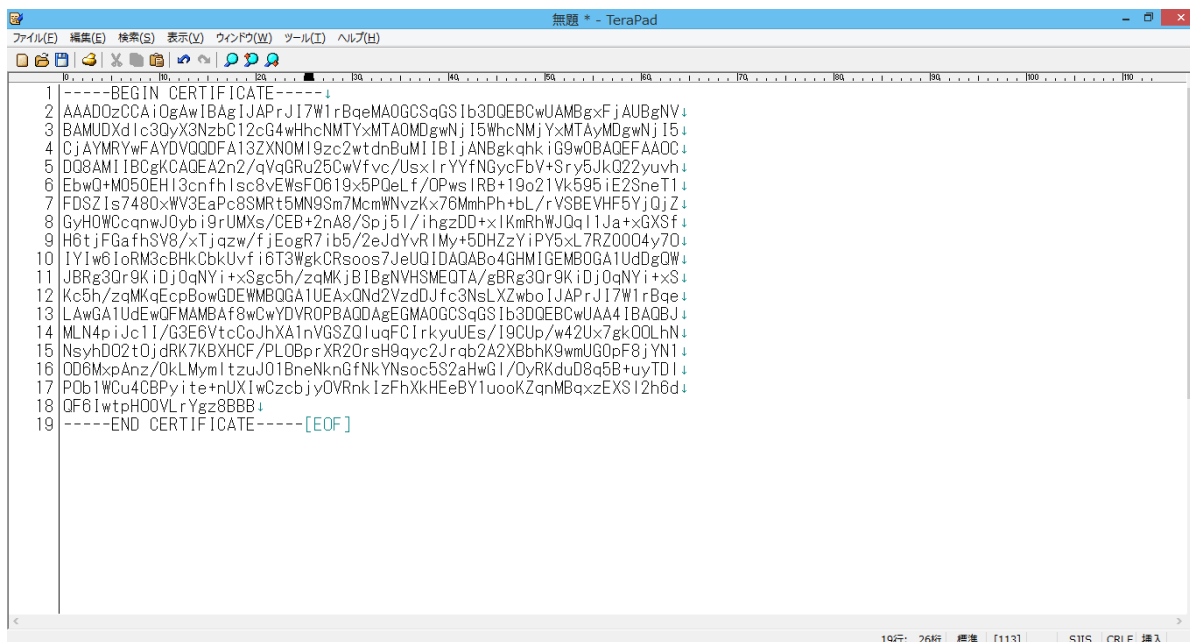
SSL-VPN用証明書登録(V2サービス/自己署名証明書利用)で実施するpayloadの改行コードのLF(\n)置換について解説します。

payloadを指定する際に、証明書の改行コードをLF(\n)に変更する必要があります。

例として、CA証明書の場合の変更前と変更後について、以下に記載します。

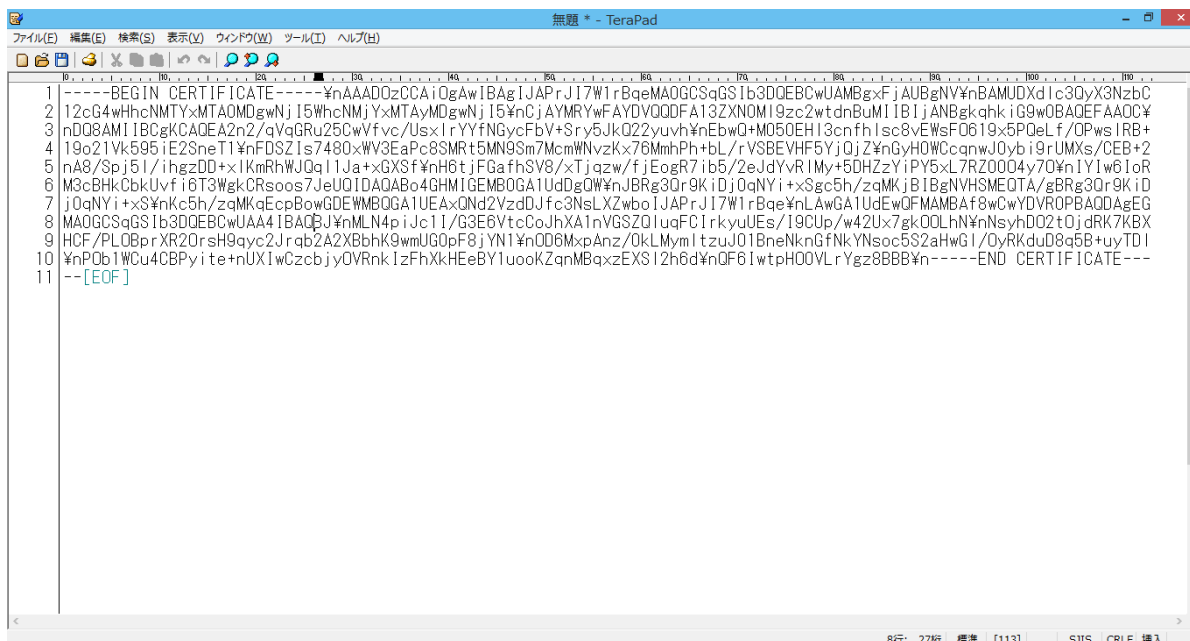
他の証明書については、それぞれの証明書ごとにpayloadを指定してください。

【変更前】改行コードが含まれている。



```
1 -----BEGIN CERTIFICATE-----
2 AAAD0zCCA10gAwIBAgIJAPrJ17W1rBqeMAOGCSqGSIb3DQEBCwUAMBgxFjAUBgNV
3 BAMUDXdIc3QyX3NzbC12cG4wHhcNMTYxMTA0MDgwNj15WncNMjYxMTAyMDgwNj15
4 CjAYMRYwFAYDQ0QDFAl3ZXNOMl9zc2wtndBuM1IB1jANBgkqhkiG9w0BAQEFAAOC
5 D08AMIIBCgKCAQEA2n2/qVqGRu25CwVfvc/UsxlrYYfNGycFbV+Sry5JkQ22yuvh
6 EbwQ+M050EH13cnfhIsc8vEwSF0619x5PQeLf/OPwsIRB+19o21Vk595iE2SneT1
7 FDSZ1s7480xwV3EaPc8SMRt5MN9Sm7McmWNVzkx76MmhPh+bL/rVSBVHF5YjQjZ
8 GyH0WCcqnwJ0yb19rUMXs/CEB+2nA8/Spj51/ihgZDD+XlKmrhWJQq11Ja+XGx3f
9 H6tjFGafhSV8/xTjqzW/fjEogR7ib5/2eJdYvRlMy+5DHZzYiPY5xL7RZ0004y70
10 IYIw6IoRM3cBhkCbKlvf16T3WgkCRsoos7JeUQIDAQABO4GHMIGEMBOGA1UdDgQW
11 JBRg30r9KIDj0qNYi+XGc5h/zqMKjB1BgNVHSMEQTA/gBRg30r9KIDj0qNYi+XG
12 Kc5h/zqMKjB1BgNVHSMEQTA/gBRg30r9KIDj0qNYi+XGc5h/zqMKjB1BgNVHS
13 LAwGA1UdEwQFMAMBAf8wCwYDVR0PBAQDAgEGMAOGCSqGSIb3DQEBCwUAA4IBAQBj
14 MLN4piJc11/G3E6VtcCoJhXA1nVGSZQluqFC1rkyuUes/19CUp/w42Ux7gk0OLhN
15 NsyhDO2t0jdrK7KBXhCF/PL0BprXR20rsH9qyc2Jrbq2A2XBbhK9wmUG0pF8jYN1
16 OD6MxpAnz/0kLMymIztzU01BneNknGfNkYNSoc5S2aHwG1/OyRKduD8q5B+uyTDI
17 POb1Wcu4CBPyite+nUX1wCzcbjyOVRnk1zFhXkHEeBY1uooKZqnMBqzEXS12h6d
18 QF6IwtpH00VLRygz8BBB
19 -----END CERTIFICATE-----[EOF]
```

【変更後】改行コードをLF(\n)に置換することによって1行で表記されている。



```
1 -----BEGIN CERTIFICATE-----%nAAAD0zCCA10gAwIBAgIJAPrJ17W1rBqeMAOGCSqGSIb3DQEBCwUAMBgxFjAUBgNV%nBAMUDXdIc3QyX3NzbC
2 12cG4wHhcNMTYxMTA0MDgwNj15WncNMjYxMTAyMDgwNj15%nCjAYMRYwFAYDQ0QDFAl3ZXNOMl9zc2wtndBuM1IB1jANBgkqhkiG9w0BAQEFAAOC%
3 nD08AMIIBCgKCAQEA2n2/qVqGRu25CwVfvc/UsxlrYYfNGycFbV+Sry5JkQ22yuvh%NEbwQ+M050EH13cnfhIsc8vEwSF0619x5PQeLf/OPwsIRB+
4 19o21Vk595iE2SneT1%NFDSZ1s7480xwV3EaPc8SMRt5MN9Sm7McmWNVzkx76MmhPh+bL/rVSBVHF5YjQjZ%NGyH0WCcqnwJ0yb19rUMXs/CEB+2
5 nA8/Spj51/ihgZDD+XlKmrhWJQq11Ja+XGx3f%NH6tjFGafhSV8/xTjqzW/fjEogR7ib5/2eJdYvRlMy+5DHZzYiPY5xL7RZ0004y70%NIYIw6IoR
6 M3cBhkCbKlvf16T3WgkCRsoos7JeUQIDAQABO4GHMIGEMBOGA1UdDgQW%NJBRg30r9KIDj0qNYi+XGc5h/zqMKjB1BgNVHSMEQTA/gBRg30r9KID
7 j0qNYi+XGc5h/zqMKjB1BgNVHSMEQTA/gBRg30r9KIDj0qNYi+XGc5h/zqMKjB1BgNVHSMEQTA/gBRg30r9KIDj0qNYi+XGc5h/zqMKjB1BgNVHS
8 MAOGCSqGSIb3DQEBCwUAA4IBAQBj%NMLN4piJc11/G3E6VtcCoJhXA1nVGSZQluqFC1rkyuUes/19CUp/w42Ux7gk0OLhN%NNsyhDO2t0jdrK7KBX
9 hCF/PL0BprXR20rsH9qyc2Jrbq2A2XBbhK9wmUG0pF8jYN1%NOD6MxpAnz/0kLMymIztzU01BneNknGfNkYNSoc5S2aHwG1/OyRKduD8q5B+uyTDI
10 %NPOb1Wcu4CBPyite+nUX1wCzcbjyOVRnk1zFhXkHEeBY1uooKZqnMBqzEXS12h6d%NQF6IwtpH00VLRygz8BBB%N-----END CERTIFICATE----
11 --[EOF]
```

A.2 証明書の形式変換

証明書の形式変換を実施する機能

ポータルからクライアント証明書を発行した場合、PKCS#12形式でダウンロードされます。クライアント証明書を利用するアプリケーションによってはPEM形式への変換が必要となります。

証明書の形式変換

PKCS#12形式の証明書をPEM形式に変換するコマンド実行例を記載します。

本章ではOpenssl を利用した実行例を記載します。



注

PKCS#12形式の証明書を操作する場合、パスワードを入力する必要があります。下記の<PKCS#12用パスワード>にはポータル上で設定した証明書用のパスワードを入力してください。

- クライアント証明書の形式変換

```
$ openssl pkcs12 -in <クライアント証明書名>.p12 -clcerts -nokeys -out <クライアント証明書名 (任意)>
```

```
Enter Import Password:<PKCS#12用パスワード>  
MAC verified OK
```

- 秘密鍵の抽出

秘密鍵の抽出には「暗号化された秘密鍵」と「暗号化されていない秘密鍵」の2つの方法があります。

アプリケーションによっては「暗号化された秘密鍵」に対応していない場合があります。アプリケーションの仕様をご確認のうえ利用してください。

- 暗号化された秘密鍵

```
$ openssl pkcs12 -in <クライアント証明書>.p12 -nocerts -out <クライアント証明書用秘密鍵名 (任意)>
```

```
Enter Import Password:<PKCS#12用パスワード>  
MAC verified OK  
Enter PEM pass phrase:<秘密鍵暗号化パスワード>  
Verifying - Enter PEM pass phrase:<秘密鍵暗号化パスワード>
```



注

暗号化された秘密鍵を作成するため、PKCS#12形式の証明書を読み込むためのパスワード以外に、暗号化のために任意のパスワードを入力する必要があります。

このパスワードは実際にアプリケーションが秘密鍵を読み込む時に入力することとなりセキュリティの強化につながります。

- 暗号化されていない秘密鍵

```
$ openssl pkcs12 -in <クライアント証明書名>.p12 -nocerts -nodes -out <クライアント証明書用秘密鍵名 (任意)>
```

```
Enter Import Password:<PKCS#12用パスワード>  
MAC verified OK
```

クライアント証明書と秘密鍵の整合性確認

クライアント証明書と秘密鍵は対応するものを使用する必要があります。クライアント証明書と秘密鍵が対応していない場合はエラーになります。

クライアント証明書と秘密鍵の整合性を確認するコマンド実行例を記載します。

1. PEM形式クライアント証明書の情報表示


```
openssl x509 -modulus -noout -in <クライアント証明書名>
```

2. 秘密鍵の情報表示

```
openssl rsa -modulus -noout -in <クライアント証明書用秘密鍵名>
```

上記「1.」「2.」のコマンド結果に表示された情報が一致する場合、クライアント証明書と秘密鍵は対応しています。

A.3 APIアクセス環境の設定 (Windows)

APIによるアクセスを行う場合にcURLコマンドが必要になりますが、Windowsでは、OSの機能としてcURLを提供していません。Windows環境でcURLコマンドを使用できるようにするため、ここでは例としてCygwinを使用した環境構築手順を説明します。Cygwinの使用実績は多数確認されていますが、Cygwinについて動作保証するものではありません。ユーザーの責任においてご利用ください。

1. Cygwinのインストール

インストーラのダウンロード

以下のURLからCygwinのインストーラをダウンロードします。使用しているWindowsの環境に合わせてダウンロードをしてください。

<https://cygwin.com/install.html>

64bitOSの場合は、`setup-x86_64.exe` (Ver2.882)

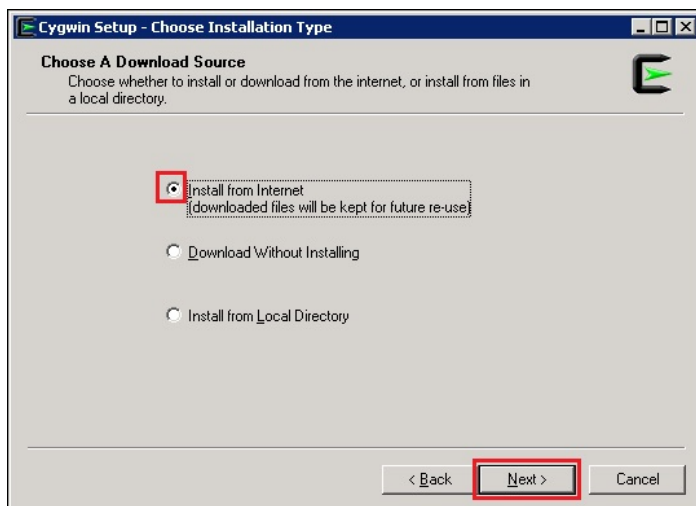
32bitOSの場合は、`setup-x86.exe` (Ver2.882)



2. パッケージのダウンロードとインストール

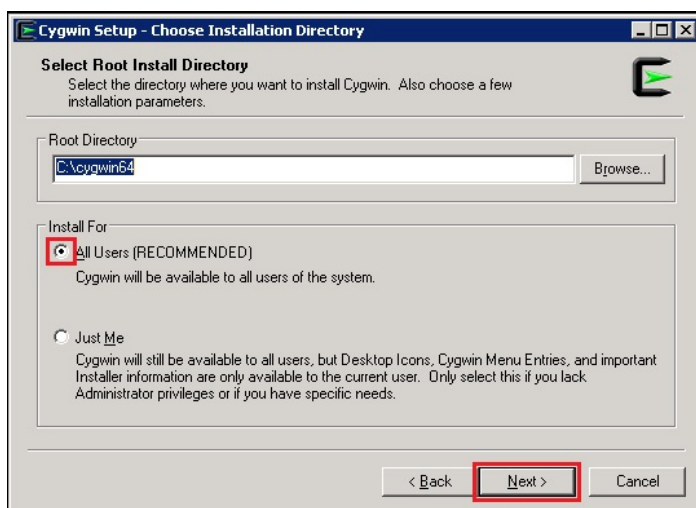
ダウンロードしたインストーラを実行します。

インターネット経由でインストールを実行します。デフォルトの「Install from Internet」のまま[Next]をクリックします。



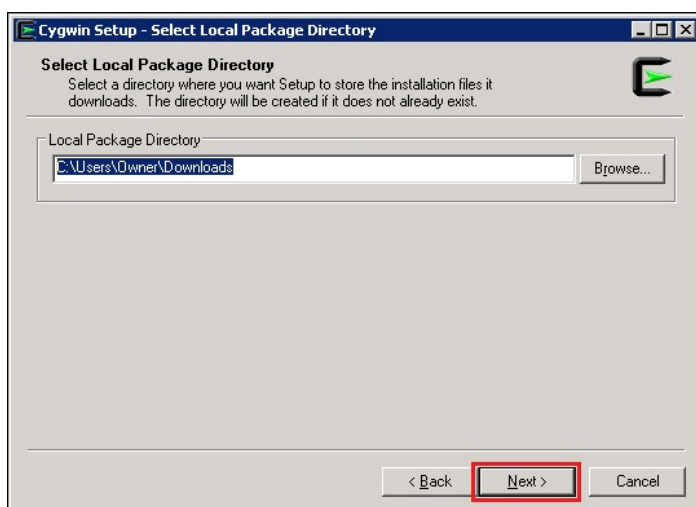
3. Cygwinで利用するルートディレクトリとユーザーの指定

ルートディレクトリは特に指定がなければデフォルトのままにします。使用できるユーザーは「All Users (RECOMMENDED)」を選択し[Next]をクリックします。



4. インストールするパッケージの保存先を指定

特に指定がなければデフォルトのままにします。[Next]をクリックします。



5. インターネット接続方法の指定

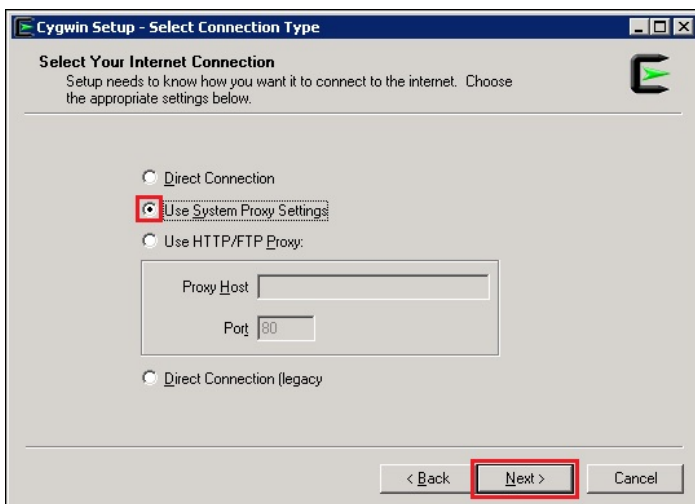
インターネット経由でインストールを実行するために、外部に接続するプロキシを指定する必要がある環境では、「Use Internet Explorer Proxy Settings」または、「Use HTTP/FTP Proxy」を選択します。手順では例として「Use Internet Explorer Proxy Settings」を選択しています。選択後は[Next]をクリックします。

・Direct Connection (プロキシを指定せずに接続)

・Use Internet Explorer Proxy Settings (ブラウザのプロキシ設定をそのまま使う)

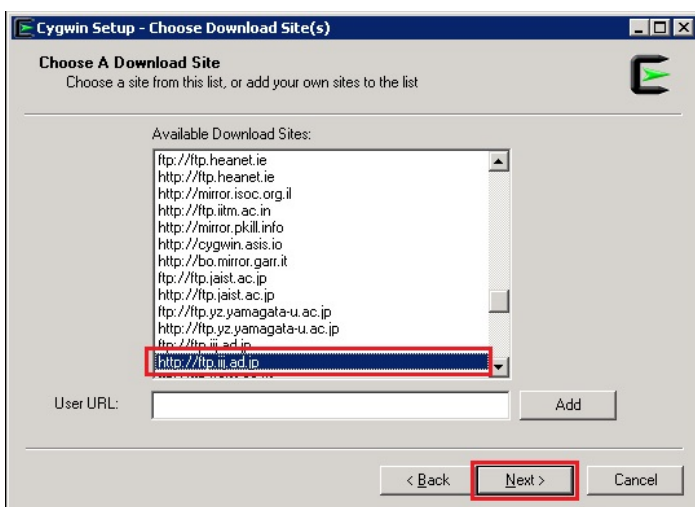
「インターネットのプロパティ」・[接続]タブ->「LANの設定」・プロキシ サーバ」の設定

・Use HTTP/FTP Proxy (プロキシを直接指定する)



6. ダウンロードするサイトの選択

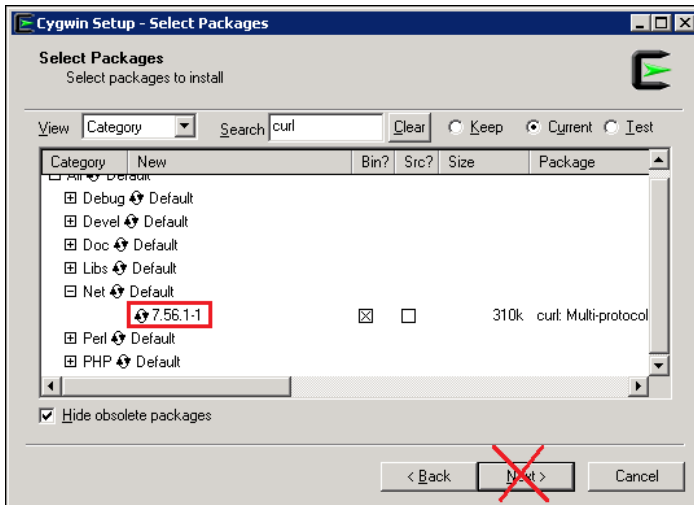
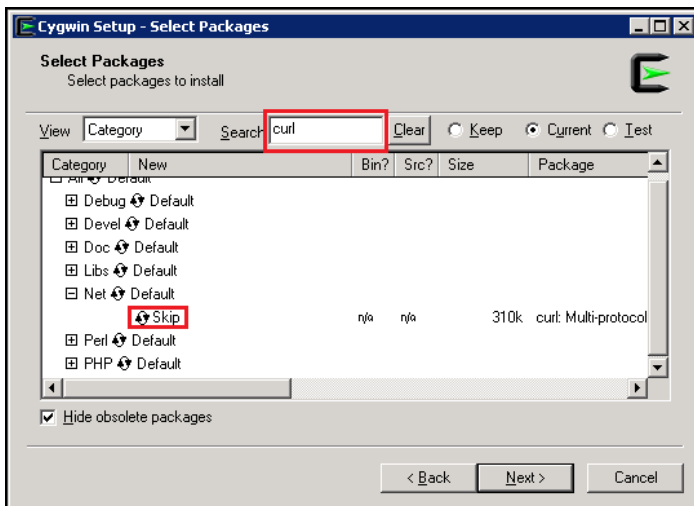
ダウンロード先のサイトが一覧表示されます。サイトを選択し[Next]をクリックします。



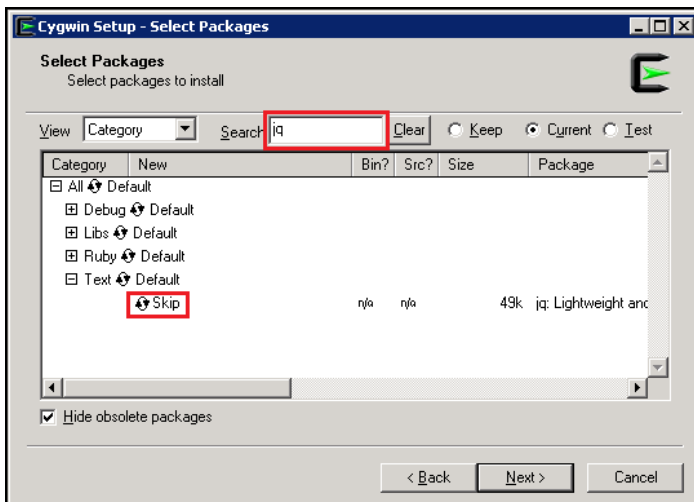
7. インストールするパッケージの「curl」を検索枠に入力

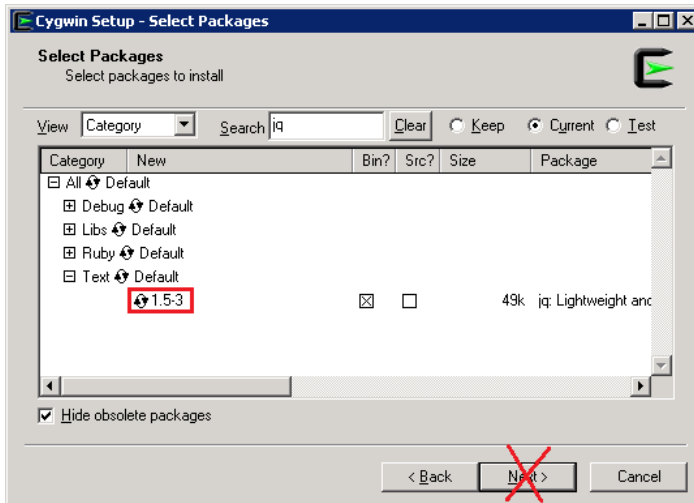
「Net」-「curl」の7.49以降(以下画面では7.56.1-1)を選択します。

「Skip」をクリックすると、インストールされるバージョン番号が表示されます。



8. インストールするパッケージの「jq」を検索枠に入力
 「Text」-「jq」の1.5-1以降(以下画面では1.5-3)を選択します。
 「Skip」をクリックすると、インストールされるバージョン番号が表示されます。



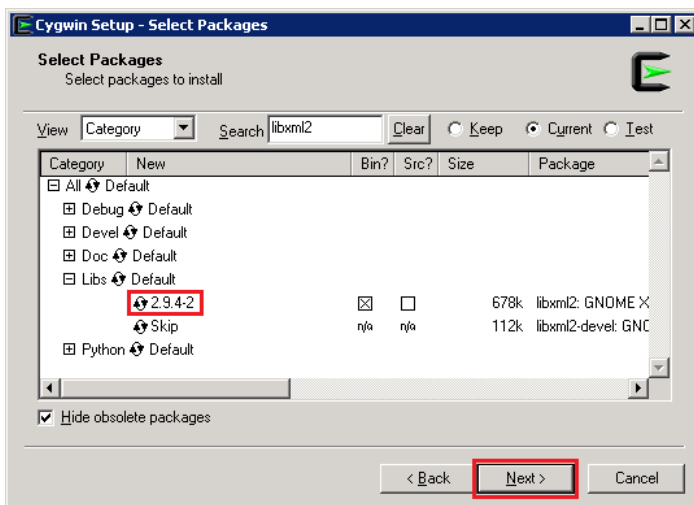
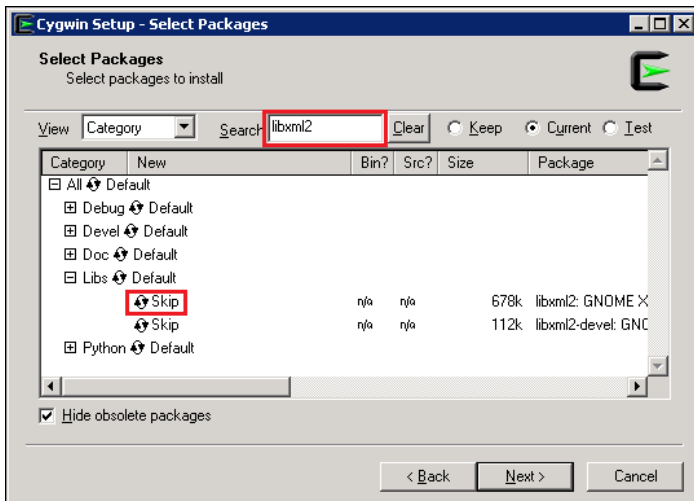


9. インストールするパッケージの「libxml2」を検索枠に入力

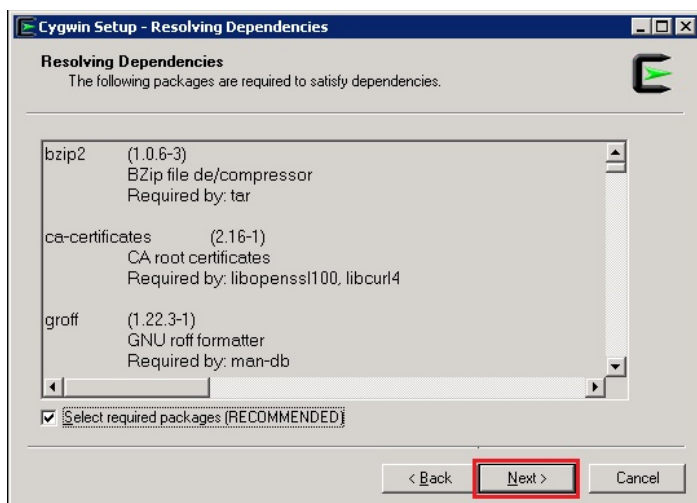
「Libs」-「libxml2」の2.9.3-1以降(以下画面では2.9.4-2)を選択します。

「libxml2:GNOME XML library(runtime)」パッケージをインストールします。

「Skip」をクリックすると、インストールされるバージョン番号が表示されます。[Next]をクリックします。

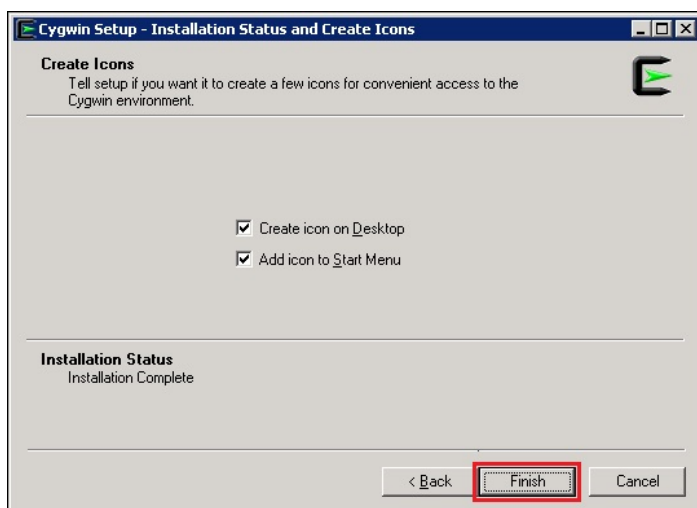


10. 依存関係の確認画面が表示されます。[Next]をクリックします。



11. 「Finish」をクリックしてインストーラを終了します。

「Create icon on Desktop」にチェックが付いている事を確認します。デスクトップに[Cygwin64 Terminal]アイコンが作成されます。

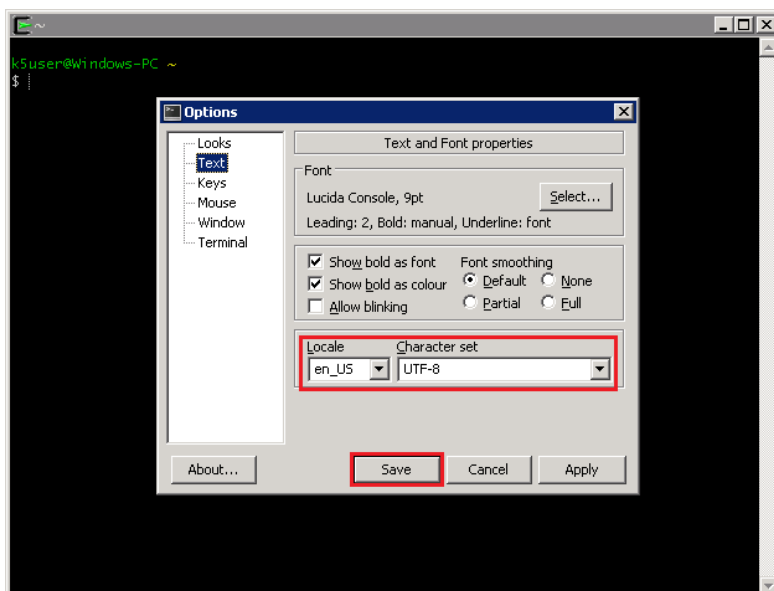


12. Cygwin Terminalの言語設定

デスクトップに作成された[Cygwin64 Terminal]アイコンをダブルクリックして起動します。

Terminalのタイトルバーを右クリックして、Optionsを選択します。

Textの Localeを[en_US]または[ja_JP]、Character setを[UTF-8]に設定します。



13. インストールした curl/jq/xmllint の確認

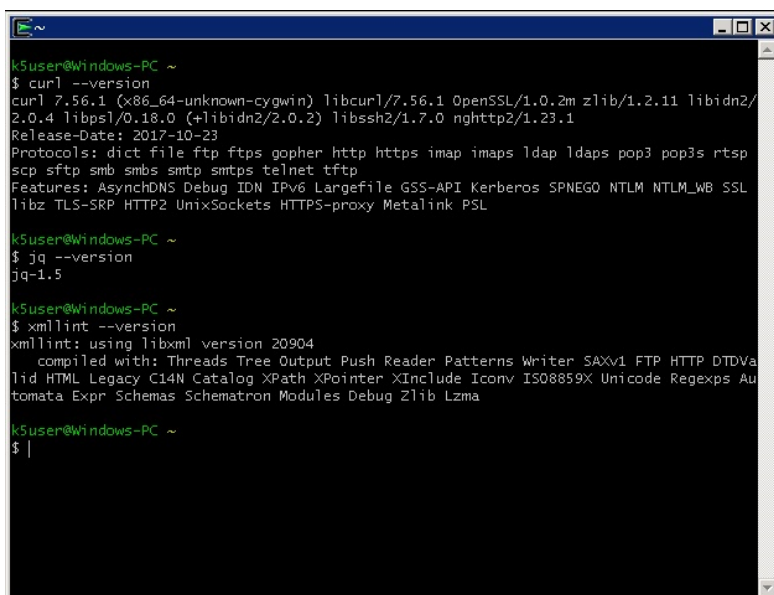
「Cygwin64 terminal」を起動して、以下のコマンドを実行します。

```
$ curl --version
```

```
$ jq --version
```

```
$ xmllint --version
```

※ コマンドが見つからない場合は、インストーラを再び起動してパッケージのインストールを実施してください。



14. curlのプロキシ設定

curlコマンドをプロキシを経由して発行する必要がある場合は、curlの初期設定ファイルに、プロキシの情報を記述する必要があります。

「C:\cygwin64\home\<ユーザー名>」ディレクトリに「.curlrc」をテキストエディタで作成します。ただし、ファイル内の改行コードはLF(UNIX改行)にします。

「.curlrc」には以下の2行を記述します。

```
proxy="<プロキシサーバ名>:<ポート番号>"
```

```
proxy-user="<ユーザー名>:<パスワード>"
```

FUJITSU Hybrid IT Service FJcloud-O

IaaS API ユーザーズガイド

Version 1.19.4

発行日 2023-10-26

All Rights Reserved, Copyright 富士通株式会社 2015-2023

- 本書の内容は、改善のため事前連絡なしに変更することがあります。
- 本書の無断複製・転載を禁じます。