ORACLE

ファイル連携機能に関する弊社理解

2025年10月

日本オラクル株式会社

ファイル連携に関する前提

- ファイル連携のための仕組みは各自治体で構築する必要がある
- 基盤構築ベンダもしくは住基システムを提供するベンダがファイル連携の仕組みを構築することが多い
- ファイル連携の仕様は決められているが、実装(APIなど)は構築するベンダにより異なる
- ファイル連携を利用するベンダは、実装するベンダから連携のための手順を入手し利用システムに実装する必要がある

ファイル連携機能の仕様

ファイル連携機能の要件は、「共通機能の標準仕様」ページ
https://www.digital.go.jp/policies/local_governments/common-feature-specification よりダウンロード可能な「ファイル連携に関する詳細技術仕様書 2.4版」
https://www.digital.go.jp/assets/contents/node/basic_page/field_ref_resources/4d056a04-6eba-4109-9850-a786d3e71971/29a74f97/20250430_policies_local_governments_common_10.docx
に記載されています。

また、ガバメントクラウドにおけるシステム間の認証については、
「ガバメントクラウド利用システムにおけるセキュリティ対策(共通) 」(最終更新: 2025年7月28日)
https://guide.gcas.cloud.go.jp/general/security-tech の記載をもとに本資料を作成しています。

求められていること ①

ファイル保存先

連携ファイルは基本的にObject Storageに保存することが求められている。

「共通機能を提供する事業者はオブジェクトストレージ上に業務の組み合わせごとのバケットを作成すること。」
(「ファイル連携に関する詳細技術仕様書 2.4版」P1)

システム間データ連携

ファイルでのデータ連携はObject Storageが提供するAPI等を利用し行うことが求められている。

「利用側業務システムは、オブジェクトストレージが提供するツール（API等）を利用し、連携ファイルを取込できること。」
(「ファイル連携に関する詳細技術仕様書 2.4版」P10)

「もしファイルでの連携が必要な場合は、オブジェクトストレージのサービスを使ってAPIでファイル連携する。」

(「 ガバメントクラウド利用システムにおけるセキュリティ対策(共通) 」内「3.システム間データ連携」)

求められていること ②

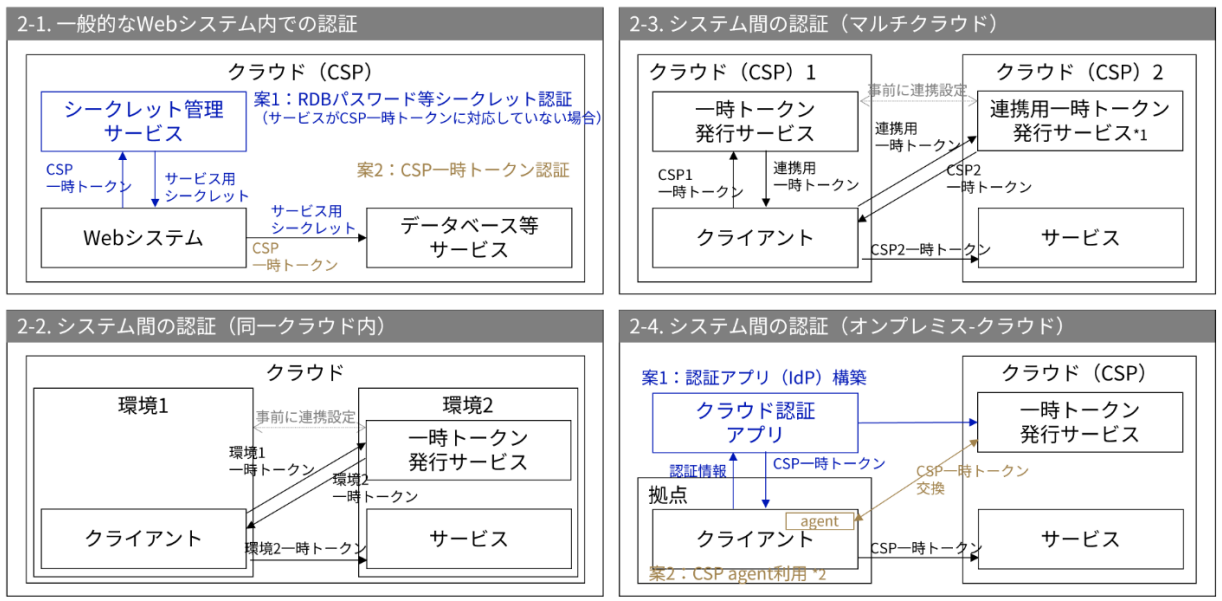
認証方式

同一CSP内での認証はCSPの認証認可機能を利用し、異なるCSPやオンプレミスとの認証は一時トークンあるいは認証サーバを利用することが求められている。

「同一クラウド内では、クラウドの機能として用意されている、サービスに紐づけた一時トークンが使えるため、これを使って認証を行う。」

「異なるクラウド間では、OIDCによるIDフェデレーションと一時トークンを組み合わせてのクラウドサービス連携を行う。(略) 一部のクラウドサービスでは利用できないこともあるため、(略) その場合は、図2-4.の構成に近づき、認証システム（IdP）を構築する方式を用意する必要がある。」

「オンプレミスとクラウドの間では、クラウドの一時トークンを発行するための認証サーバ（認証アプリ）をOAuth M2M認証などの仕組みを使って開発して用意するか、クラウドサービスの一時トークンを使ったオンプレミスサーバとのファイル連携機能を利用する。」



ガバメントクラウドでの認証パターン（2. システムの認証）

出典：「ガバメントクラウド利用システムにおけるセキュリティ対策(共通) 」内「2.システムの認証」
https://guide.gcas.cloud.go.jp/general/security-tech (最終更新: 2025年7月28日)

参考：「ファイル連携に関する詳細技術仕様書 2.3版」におけるファイル連携手順

ファイル連携の具体的な手順として、旧版である「ファイル連携に関する詳細技術仕様書 2.3版」では、
「図2-6 オブジェクトストレージにおける連携」(P10) に、同一CSP内ではCSPの認証認可機能を利用したファイル格納、取得、異なるCSPやオンプレミスとの連携では閉域網からのみアクセス可能な「認証認可サーバ」、「アクセストークン」、「一時的なクレデンシャル」を利用したAPI連携によるファイル格納、取得が求められていた。

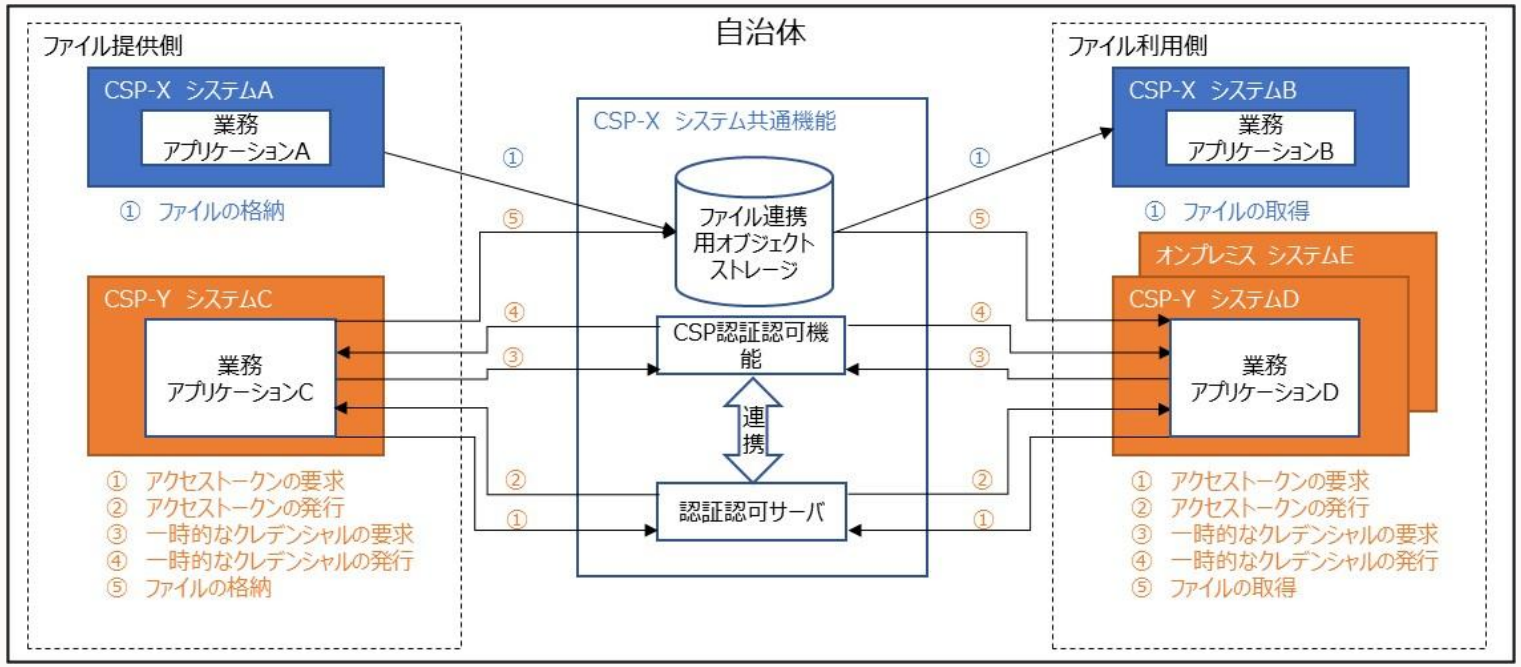


図2-6 オブジェクトストレージにおける連携

出典：ファイル連携に関する詳細技術仕様書 2.3版

参考：ファイルサーバの構築

ここまで手順を記載していたオブジェクトストレージを利用したAPIによるファイル連携が困難と判断した場合には、ファイルサーバを構築してSFTPやSCPプロトコルを利用してファイル連携を行うことが認められている。

「既存システムとの連携等、オブジェクトストレージを利用したファイル連携が困難な場合において、ファイルサーバを構築の上、ファイル連携を行うこと。」(「ファイル連携に関する詳細技術仕様書 2.4版」 P12)

「提供側業務システムは、SFTP、SCP等による伝送データの暗号化 を行うこと。」(P15)

OCIにおけるファイル連携

ファイル連携機能に関する弊社理解(2025年9月版)

ファイル連携機能の方式は、基本的に一時的なクレデンシャルを利用してObject StorageへのAPIアクセスで実施することが求められている。

OCIでは「一時的なクレデンシャルを発行するサービス」として、OCI Identity Domainsの一機能、**OCI IAM Workload Identity Federation(WIF)**を提供している。

この機能を利用することで、認証認可サーバとOpenID Connectで連携して、一時的なクレデンシャル(User Principal Session Tokens(UPST))を利用して、Object StorageへのAPIアクセスが可能である。

OCI IAM Workload Identity Federation

https://blogs.oracle.com/oracle4engineer/post/ja-oci-iam-workload-identity-federation

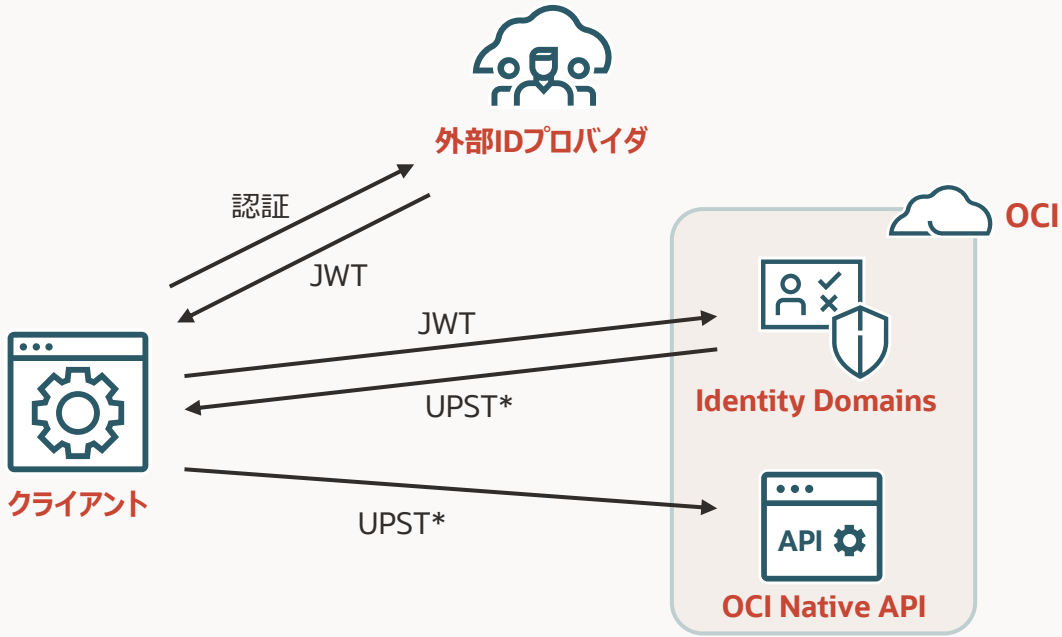
OCI IAM Workload Identity Federation(WIF) [リリース日: 2025年7月17日]

外部の認証情報でOCIにアクセスする JWT交換機能のサポート

外部 (他クラウド・オンプレミス・CI/CD等)のワークロードが、OCI IAM認証を利用して短期間のトークン方式でアクセス可能になります。

外部IDプロバイダが発行する署名付きJWTを、
OCI Identity Domainsが受け取り、一時的な認証トークン
「UPST(User Principal Security Token)」に交換します。

UPSTを使用することでAPIキーを直接保存することなく
OCI リソースに安全にアクセスできます。



* User Principal Security Token

<参考>
OCIドキュメント「トークン交換権限付与タイプ: UPSTのJSON Webトークンの交換」
https://docs.oracle.com/ja-jp/iaas/Content/Identity/api-getstarted/json_web_token_exchange.htm

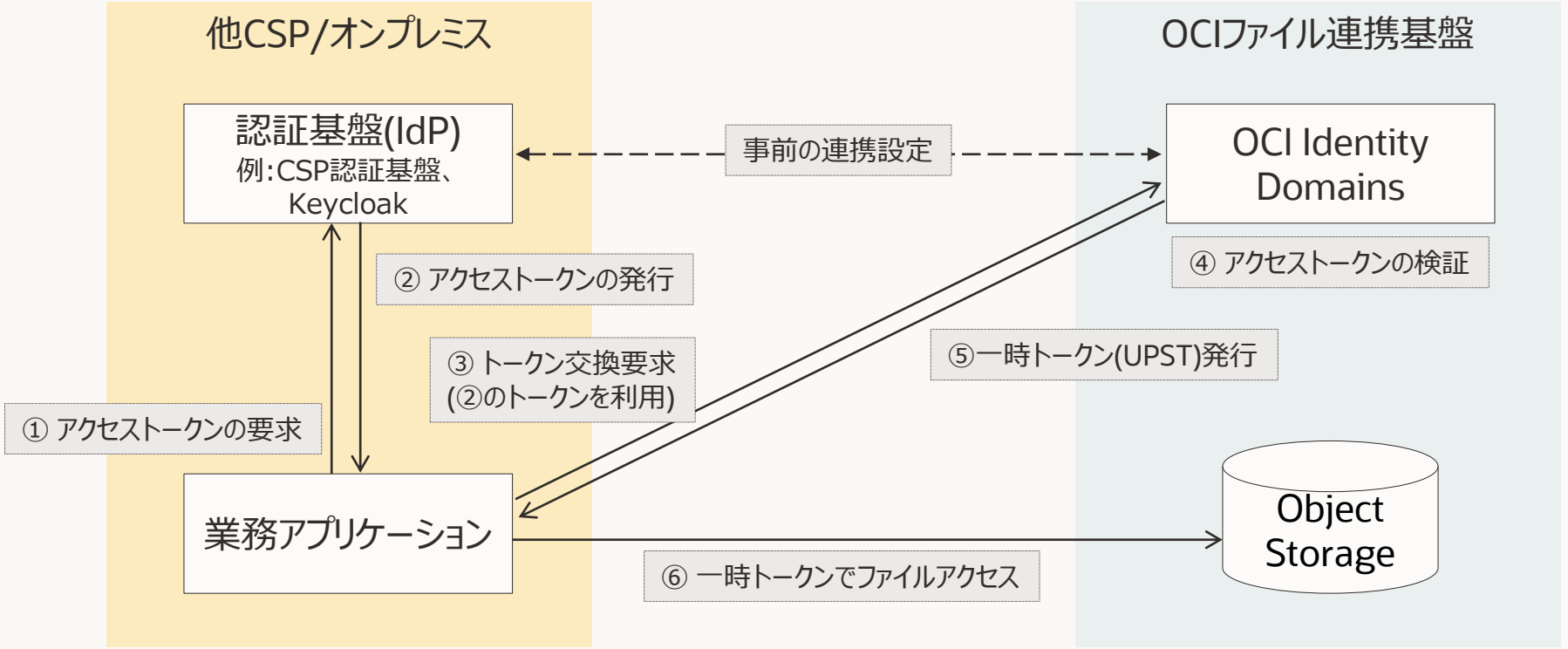
技術仕様に合わせたOCIでのUPSTによるファイル連携の例

認証基盤を他CSP/オンプレミスに構築するパターン

この図では、「ガバメントクラウド利用システムにおけるセキュリティ対策(共通)」図2-3に合わせて、他CSP環境に認証基盤を構築しています。

CSP標準の認証基盤が閉域網で利用できない場合、Keycloakなどで別に認証基盤(IdP)を構築する必要があります。

認証基盤はOCI側に構築することも可能です（次ページ）



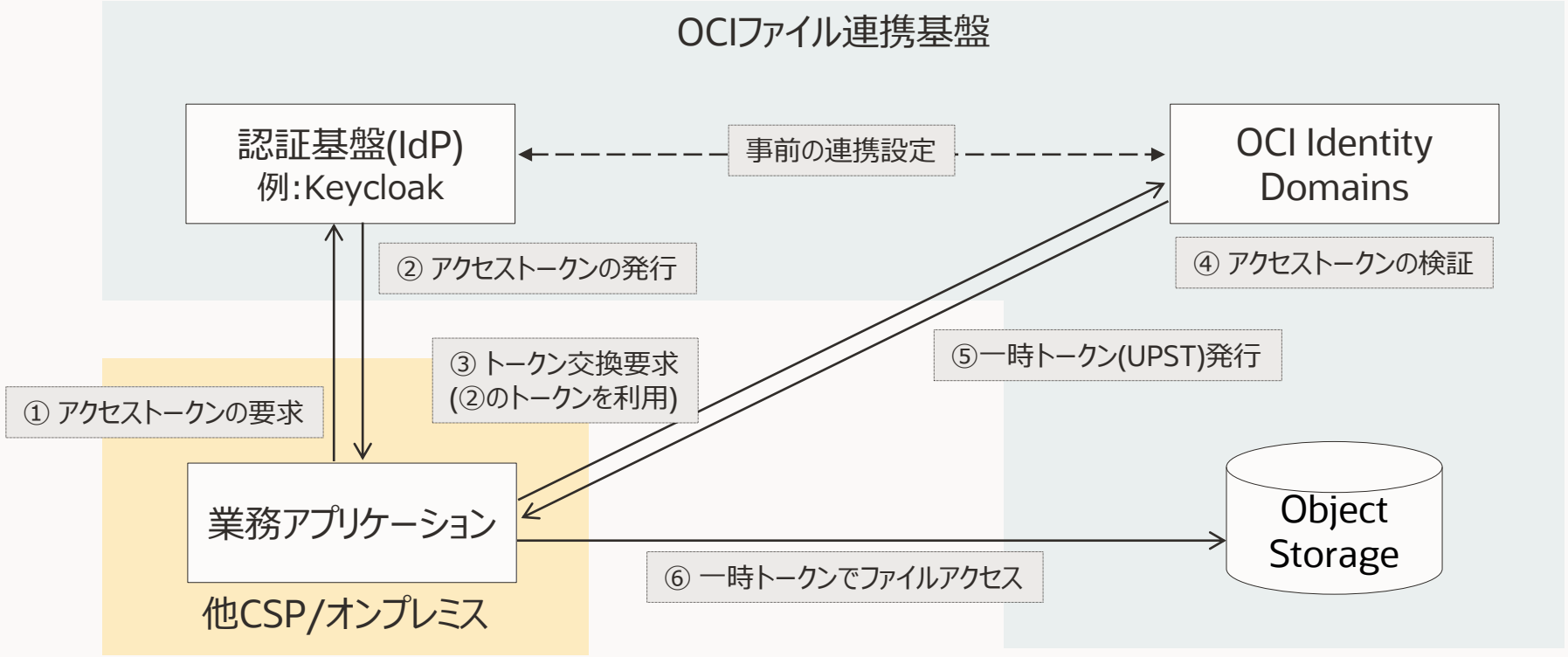
※①～②の認証の連携に関しては、OCIサービス外のため技術仕様に合わせて記載しています。
②のアクセストークンとしてはJSON Web Token(JWT)形式のトークンを利用可能です。

技術仕様に合わせたOCIでのUPSTによるファイル連携の例

認証基盤をOCIに構築するパターン

ファイル連携基盤であるOCI環境上に認証基盤を構築することも可能です。

同一基盤上に認証基盤、一時トークン発行サービス、オブジェクトストレージが構成されますので、「ファイル連携に関する詳細技術仕様書」旧2.3版の図2-6に相当する構成となります。



※②のアクセストークンとしてはJSON Web Token(JWT)形式のトークンを利用可能です。

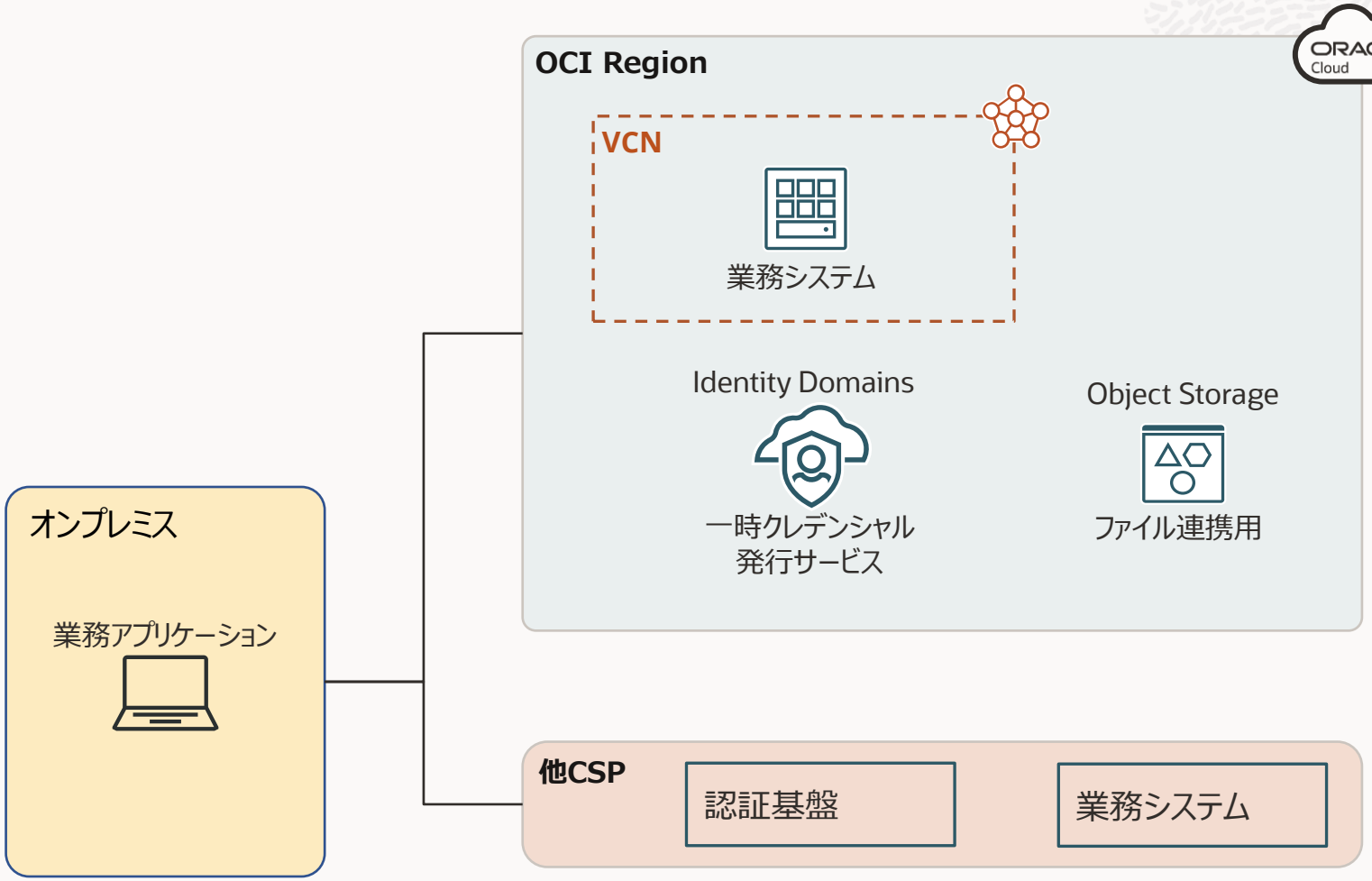
ガバメントクラウドにおけるファイル連携の構成パターン

ガバメントクラウドにおけるファイル連携の構成として、本資料では以下のパターンをご提示しています。
使用する業務システムの仕様やアウトバウンド通信量に応じて、ファイル連携基盤を設置するクラウドを検討して下さい。

パターン	ファイル連携基盤の構築場所	認証認可方法	認証基盤の構築場所
①	OCI環境	一時トークン(UPST)の利用	他CSP環境 (p11の構成)
②	OCI環境	一時トークン(UPST)の利用	OCI環境 (p12の構成)
③	OCI環境	カスタムアプリケーションによる認証	OCI環境
④	他CSP環境	CSPの認証認可機能の利用	他CSP環境

⑤	OCI上にファイルサーバを構築する(Object Storageの利用が困難な場合,P7の構成)

ガバメントクラウドにおけるファイル連携の構成パターン①

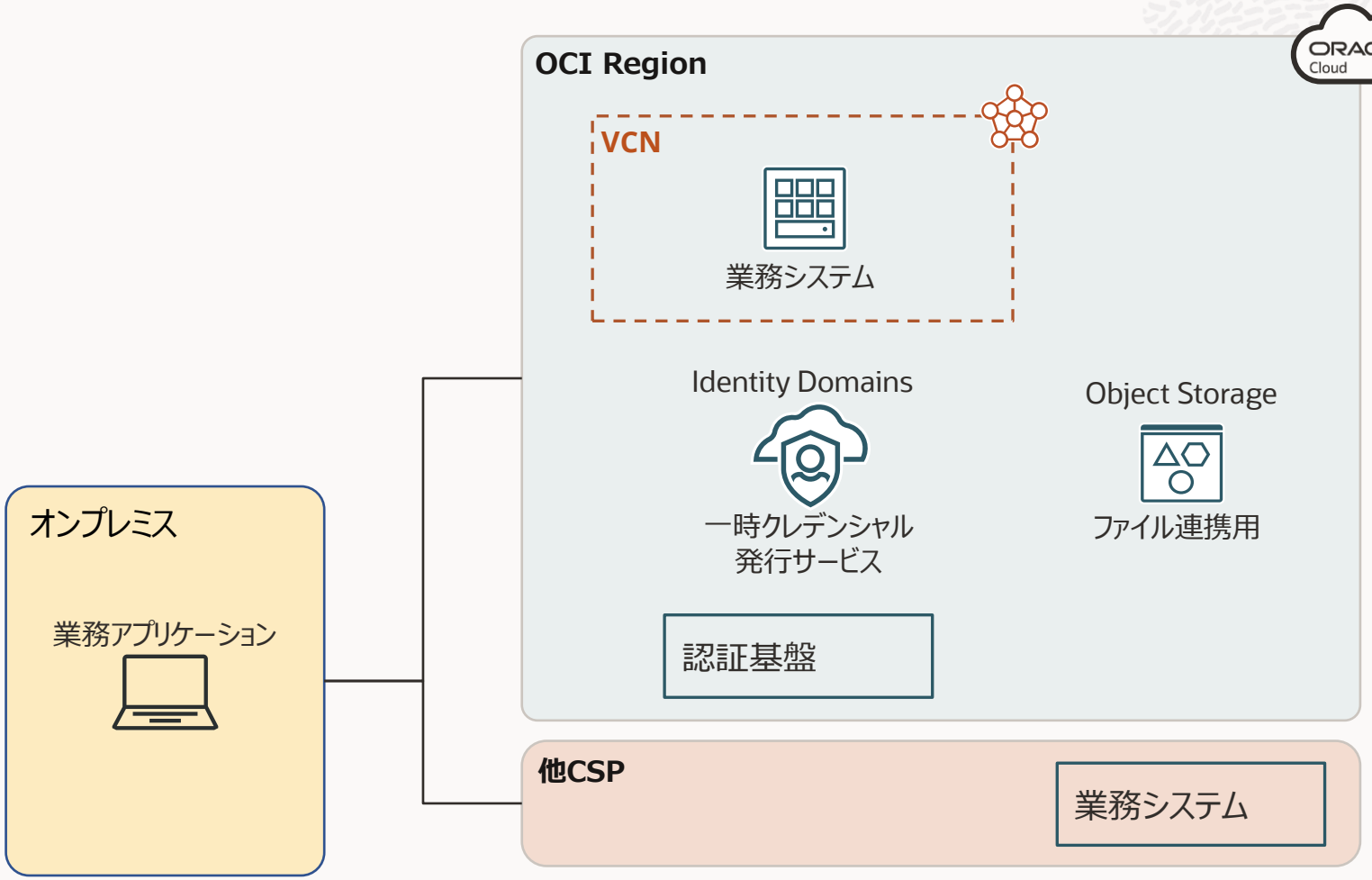


ファイル連携基盤を OCI に構成します。

OCI 内にある業務システムは、インスタンスプリンシパルを使用して、Object Storage にアクセスします。

他CSP やオンプレミスからのアクセスでは、認証は他CSP環境上の認証基盤で行われ、OCI IAM Workload Identity Federationによって発行される一時トークン(UPST)を利用して、Object Storage にアクセスします。

ガバメントクラウドにおけるファイル連携の構成パターン②

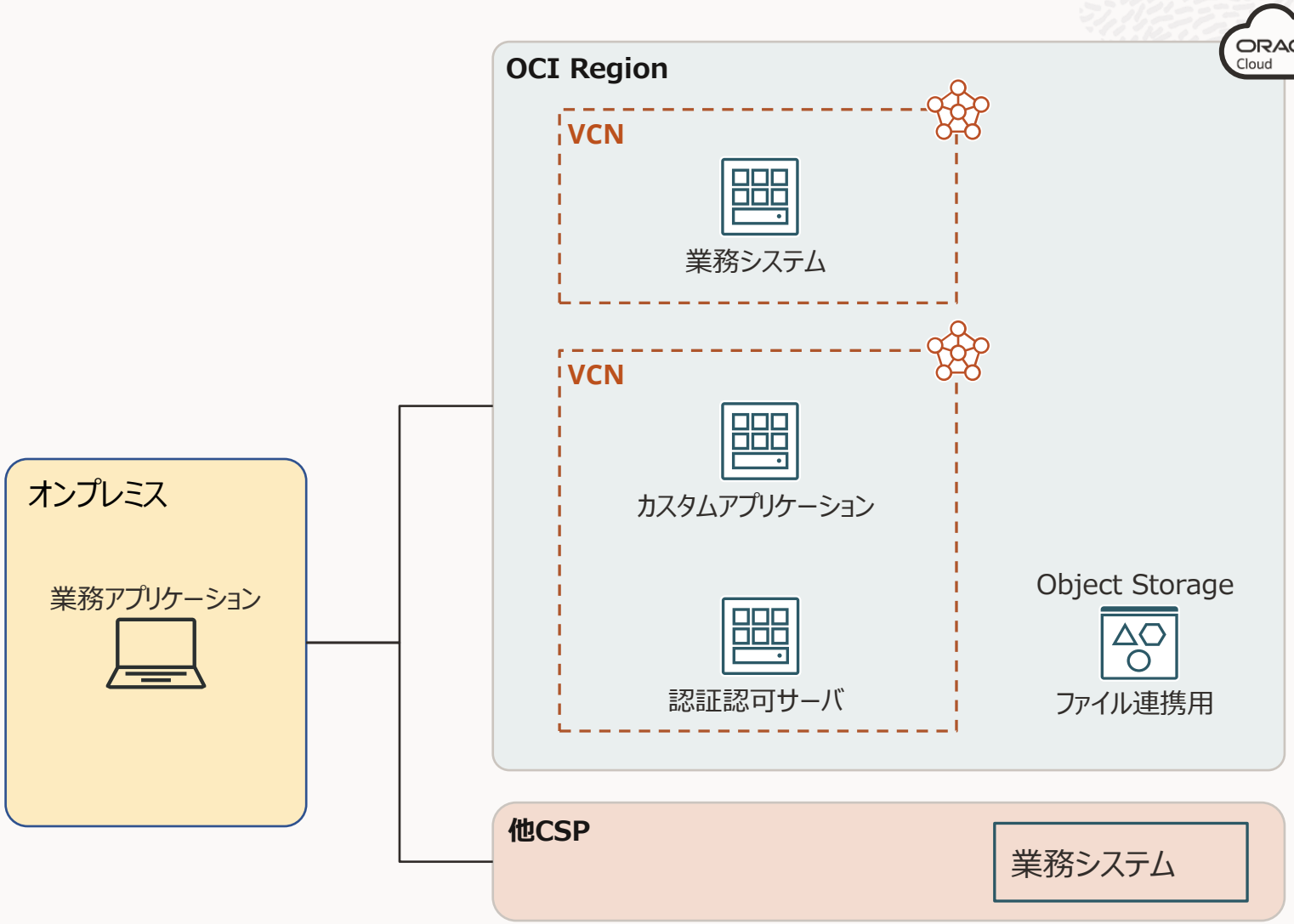


ファイル連携基盤を OCI に構成します。

OCI 内にある業務システムは、インスタンスプリンシパルを使用して、Object Storage にアクセスします。

他CSP やオンプレミスからのアクセスでは、認証はOCI環境上の認証基盤で行われ、
OCI IAM Workload Identity Federationによって発行される一時トークン(UPST)を利用して、Object Storage にアクセスします。

ガバメントクラウドにおけるファイル連携の構成パターン③



ファイル連携基盤を OCI に構成します。

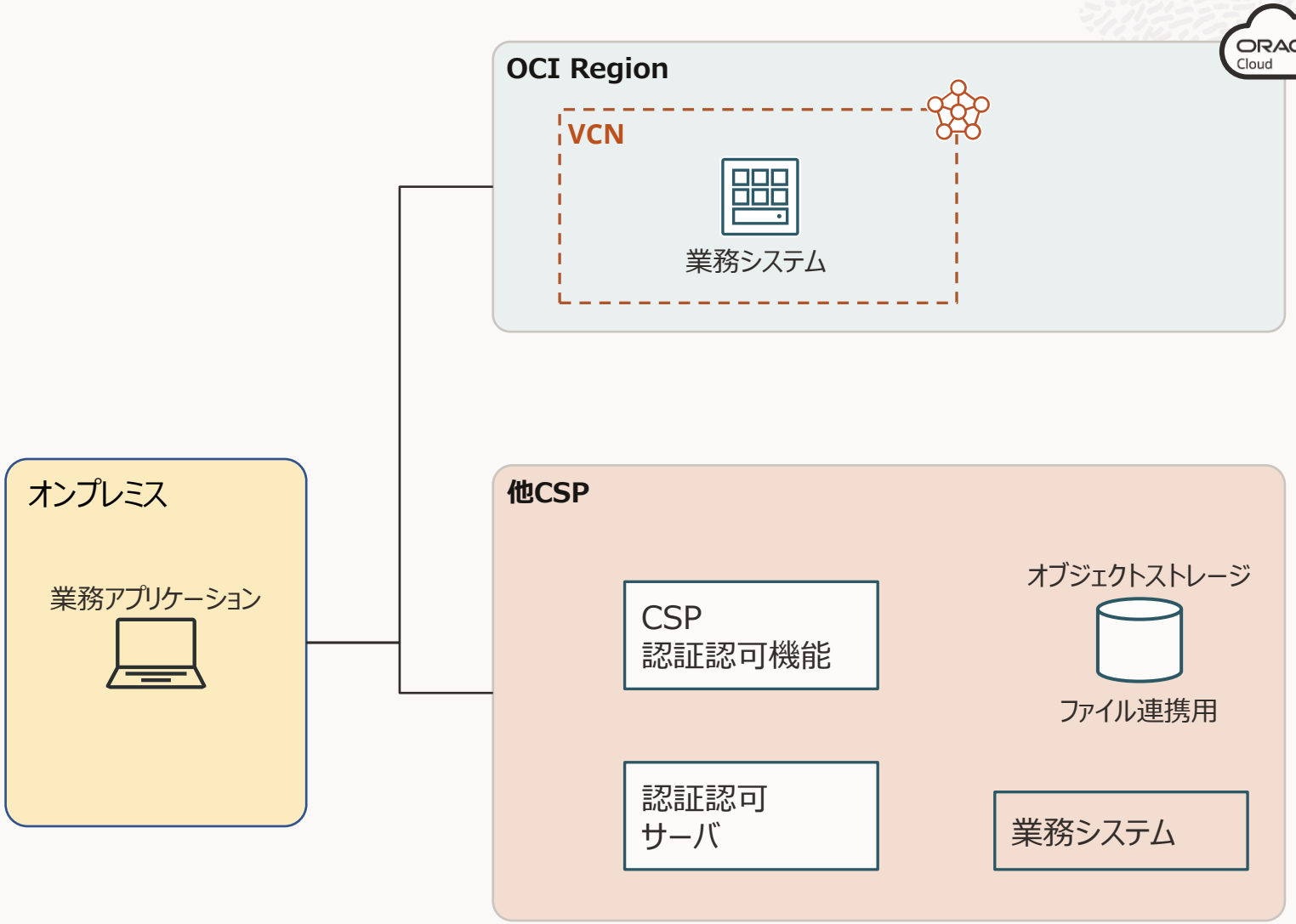
OCI 内にある業務システムは、インスタンスプリンシパルを使用して、Object　Storage にアクセスします。

以前はOCIには「一時的なクレデンシャルを発行するサービス」がなく、カスタムアプリケーション(CSP認証認可機能)の開発が必要でした(Appendix参照)。

他CSP やオンプレミスからは、認証認可サーバとカスタムアプリケーションを経由して、Object Storage にアクセスします。

認証認可サーバとカスタムアプリケーションは1つのサーバで構成することも可能です。

ガバメントクラウドにおけるファイル連携の構成パターン④

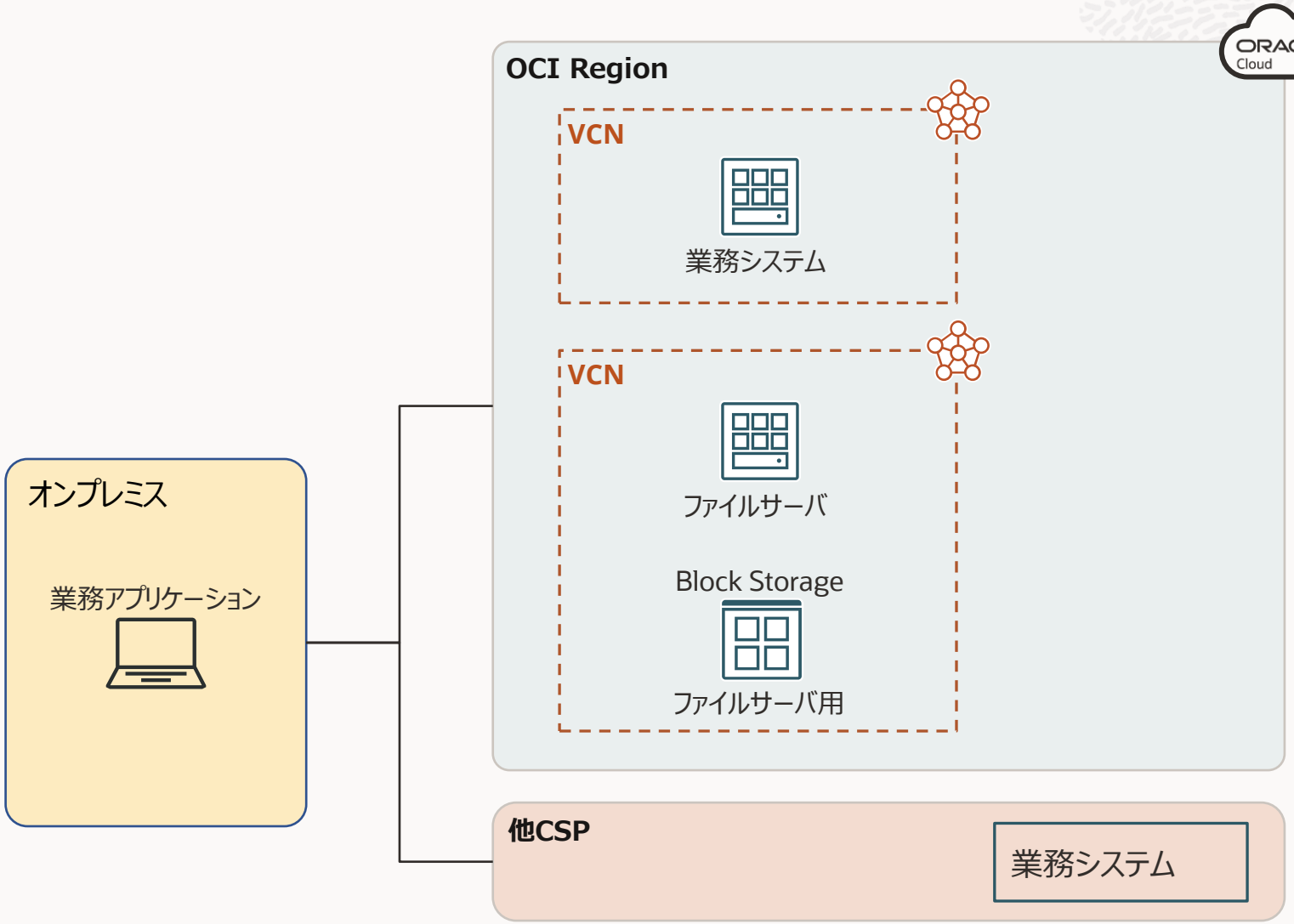


ファイル連携基盤を 他CSP に構成します。

同一CSP 内にある業務システムは、CSP内の認証を使用して、Object Storage にアクセスします。

OCI やオンプレミスからは、認証認可サーバとCSP認証認可機能を経由して、オブジェクトストレージ にアクセスします。

ガバメントクラウドにおけるファイル連携の構成パターン⑤



ファイル連携基盤を、ファイルサーバとして OCI に構成します。

OCI 内および他CSP、オンプレミスにある業務システムは、SFTPやSCPプロトコルにて暗号通信でアクセスします。

「ファイル連携に関する詳細技術仕様書 2.4版」には、オブジェクトストレージの使用が困難である場合の選択肢として記載されています。

OCIFSにて、Oracle LinuxサーバにObject Storageをマウントすることも可能です。

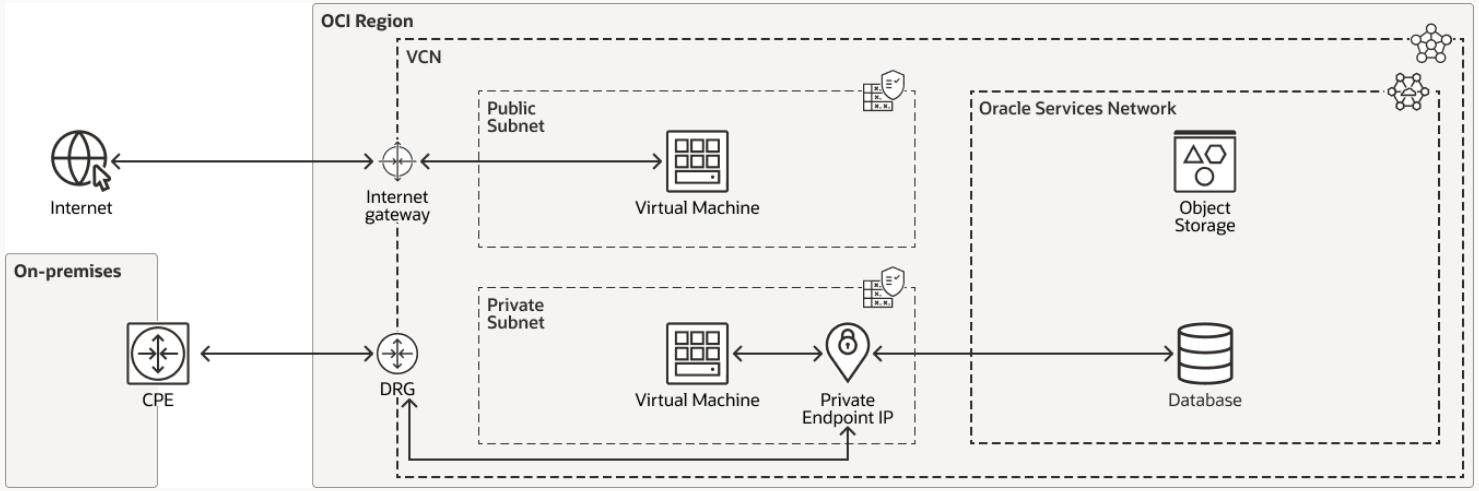
【ご参考】オブジェクト・ストレージのプライベート・エンドポイント [リリース日: 2024年8月27日]

プライベート・エンドポイントがサポートされ、仮想クラウド・ネットワーク(VCN)またはオンプレミス・ネットワークからのプライベートIPアドレスを使用してObject Storageサービスにアクセスできるようになりました。

ネットワークトラフィックは OCI 内に留まり、パブリック インターネットを通過することはありません。

<OCIプライベートエンドポイントの利点>

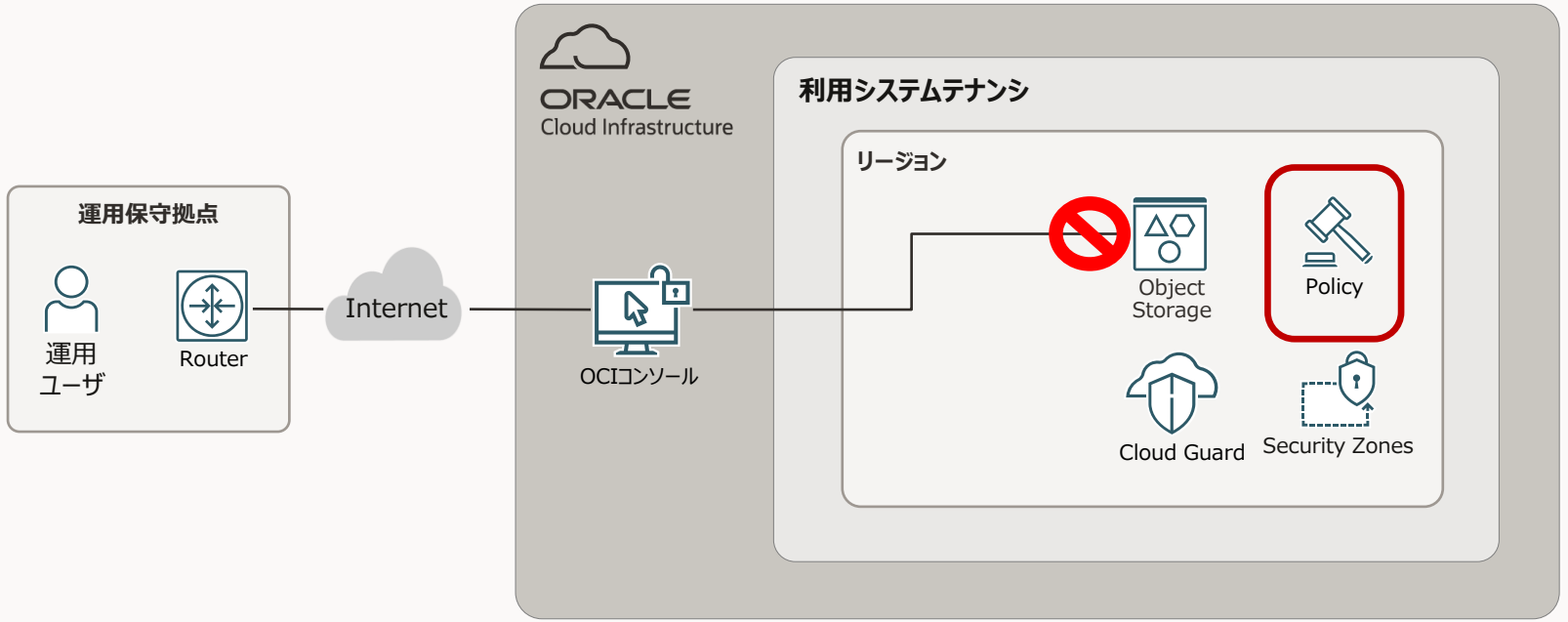
1. 単一のOCIサービスへの安全でプライベートなアクセス
2. 25以上のサービスが利用可能
3. OCIプライベートエンドポイントには追加料金はかかりません



【ご参考】安全なシステム
インターネット経由でのObject Storage接続を抑止

オンプレミスや事業者とガバメントクラウドのネットワーク接続は、セキュリティが十分担保された上でインターネット経由での接続が基本となっています。しかし、自治体情報システムにおける三層分離の考え方に基づき、インターネットからのアクセスを抑止する必要があります。

デジタル庁様から払い出される利用システムテナンシには、予防的統制や発見的統制にてパブリック・アクセスが制限されていますが、APIキーやOCI管理コンソールを経由したインターネットアクセスも制限する必要があるため、ポリシーの設定で補完する必要があります。



Cloud Guard、Security Zones
(予防的統制、発見的統制)

オブジェクト・ストレージに対し、パブリック・アクセスを禁止する。

Policy

ネットワーク・ソースに登録した必要最小限のIPアドレスからのみオブジェクト・ストレージにアクセスできるようポリシーにて制御する。

Appendix

OCIにおけるファイル連携(従来方式：構成パターン③)

ファイル連携機能に関する弊社理解(2024年12月版)

ファイル連携機能の方式は、基本的にObject StorageへのAPIアクセスで実施することが求められている。

Object StorageへのAPIアクセスで実現する場合、同一CSP内での連携であれば、すべてのCSPはCSP機能で完結して連携することが可能だが、異なるCSPやオンプレミスと連携する場合には、すべてのCSPで「認証認可サーバ」を構築する必要があり、CSP機能だけで完結させることはできない。

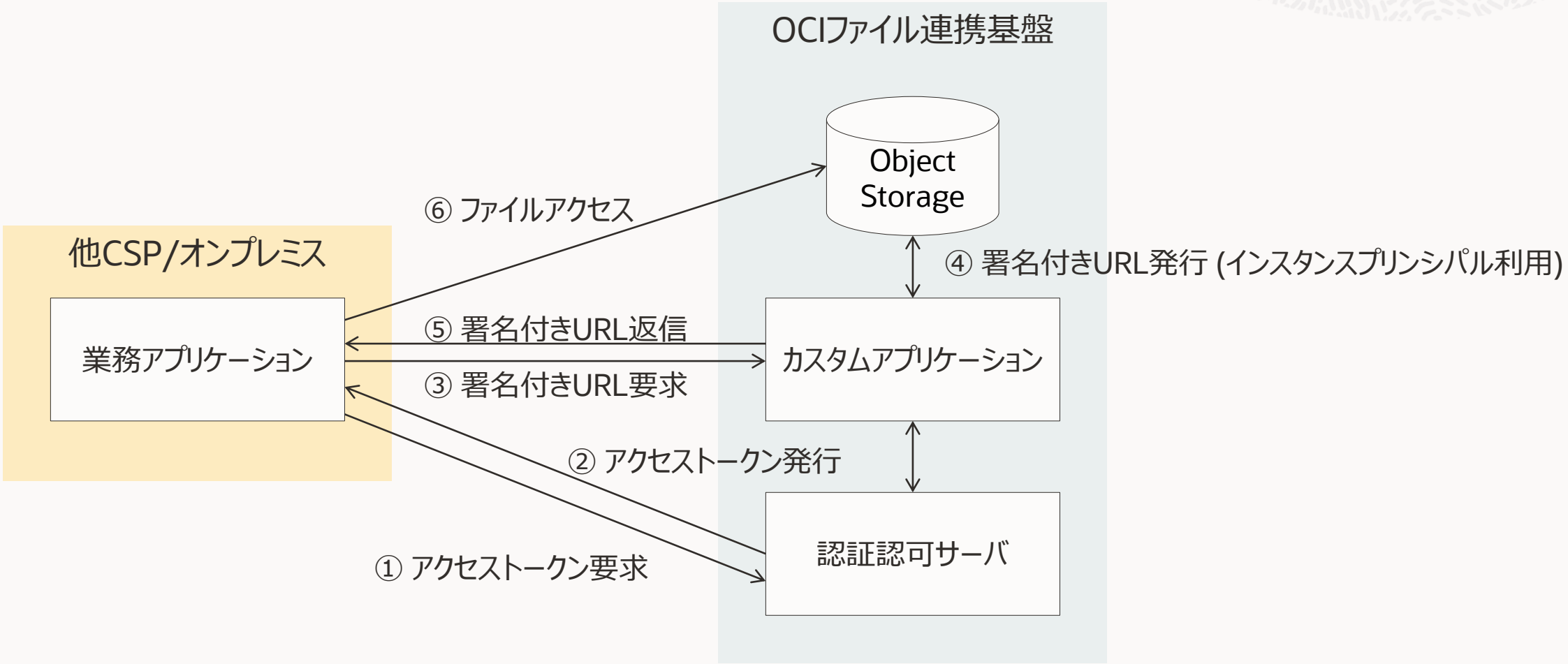
OCIでは「一時的なクレデンシャルを発行するサービス」がないため、「地方公共団体情報システム 認証機能・ファイル連携機能に関するリファレンスガイド（Oracle Cloud Infrastructure編）」にある通り、カスタムアプリケーション(CSP認証認可機能)の開発が必要とある。ただし、「署名付きURLを発行する方法」が許容されているため、「一時クレデンシャル」をクライアントに戻す代わりに「オブジェクト・ストレージの事前認証済リクエストURL」を作成し、クライアントに戻すカスタムアプリケーションが必要となる。追加手順となるのは、「オブジェクト・ストレージの事前認証済リクエストURL」の作成のみと最小限である。

オブジェクト・ストレージの事前認証済リクエスト

https://docs.oracle.com/ja-jp/iaas/Content/Object/Tasks/usingpreauthenticatedrequests.htm

APIによるファイル連携が困難な場合には、当面の回避策としてファイルサーバへのSFTP、SCPアクセスで実施することも認められている。

技術仕様に合わせたOCIでのカスタムアプリケーションの一例



※　①～③の認証の連携に関しては、OCIサービス外のため技術仕様に合わせて記載しています。

Appendix

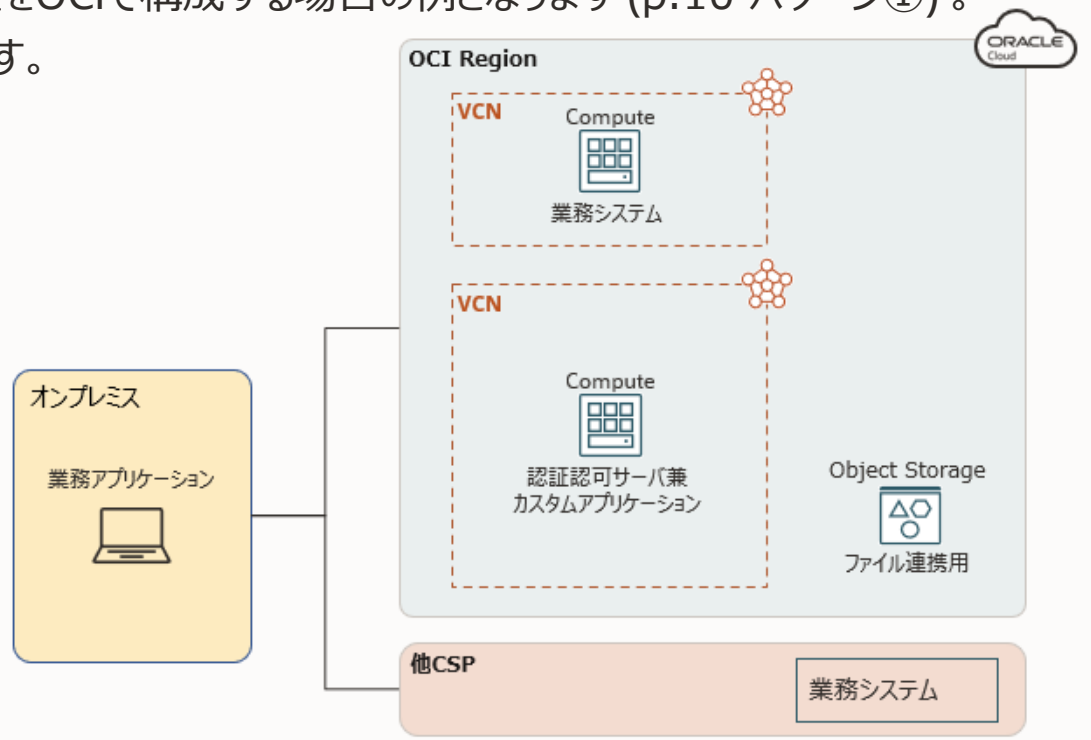
カスタムアプリケーション 環境構築ガイド

環境構築ガイド
概要

こちらの環境構成ガイドは、他CSPやオンプレミスとのファイル連携基盤をOCIで構成する場合の例となります (p.10 パターン①) 。
本ガイドでは以下の構成についての手順とサンプルコードをご紹介します。

【概要】

- 認証認可サーバとカスタムアプリケーションを1つのサーバで構成する
  ※認証認可の設定は本ガイドには含まれません (詳細は次ページ記載)
- インスタンスプリンシパルを使用して「オブジェクト・ストレージの事前認証済リクエストURL (署名付きURL) 」を取得する
- 署名付きURLを使用して、業務アプリケーションからファイルアクセス (格納/取得) する
- 取得後の対象ファイルを削除する



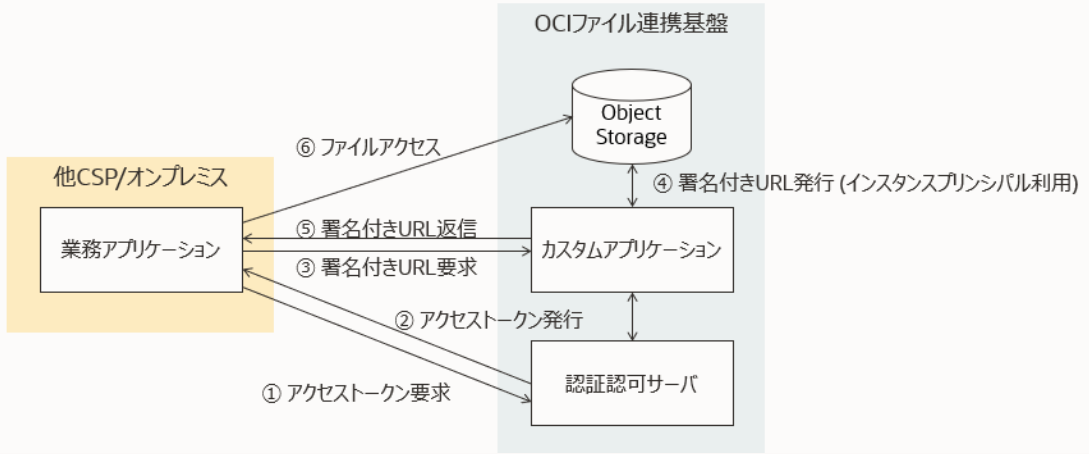
構成イメージ (ファイル連携基盤をOCIで構成)

本ガイドの手順は一例であり、コードやコマンドは必要に応じて編集してお使いいただきますようお願いいたします。

環境構築ガイド
ご参照いただくにあたっての注意点

- 本ガイドはファイル連携基盤の流れのうち、署名付きURLの要求からファイルアクセスまでの手順の例になります。

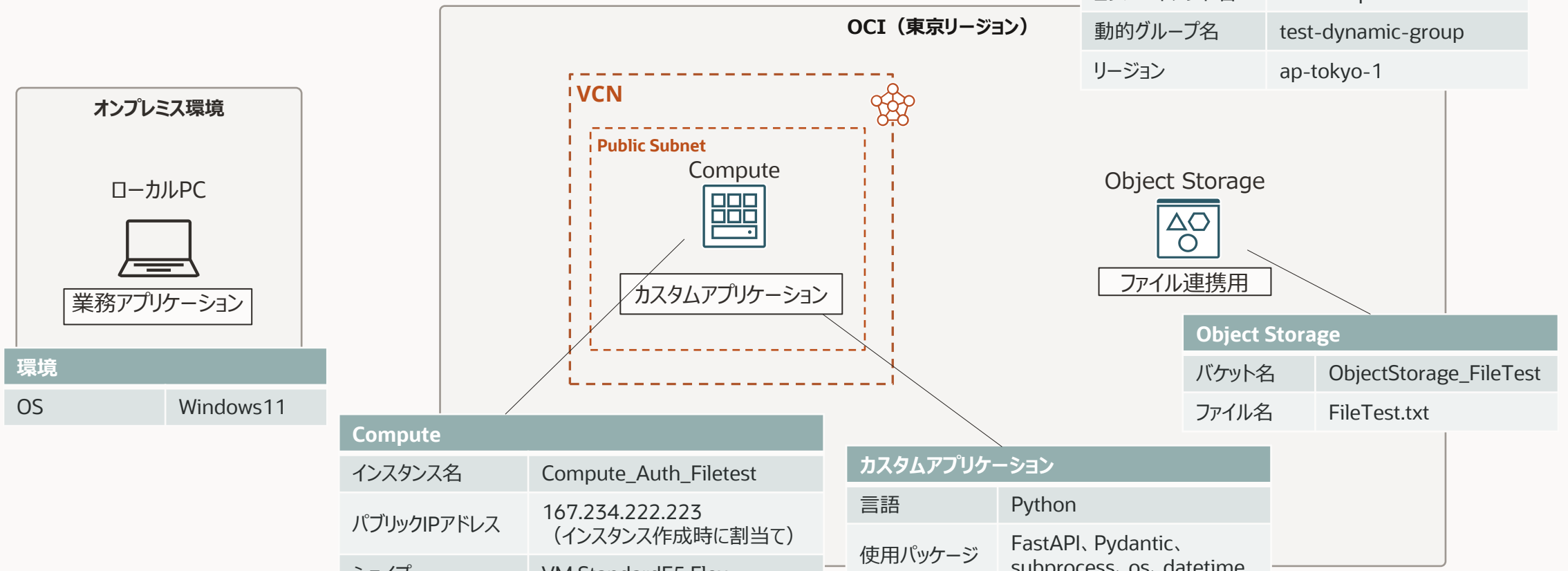
- 認証認可サーバの構成やアクセストークンの要求および発行の手順、カスタムアプリケーションとの連携方式については含まれておりませんので、別途ご検討いただきますようお願いいたします。



- 本ガイドは手順の例として最小限の構成としております。実際の運用において可用性を高めるための冗長化等が必要な場合には別途ご検討いただきますようお願いいたします。

環境構築ガイド
アーキテクチャ図

以降の手順で使用している環境情報です。適宜ご自身の環境に読み替えてご参照ください。



テナンシ	
ネームスペース名	testtenancy
コンパートメント名	test.compartment
動的グループ名	test-dynamic-group
リージョン	ap-tokyo-1

環境	
OS	Windows11

Compute	
インスタンス名	Compute_Auth_Filetest
パブリックIPアドレス	167.234.222.223 （インスタンス作成時に割当て）
シェイプ	VM.StandardE5.Flex
OS	Oracle Linux8

Object Storage	
バケット名	ObjectStorage_FileTest
ファイル名	FileTest.txt

カスタムアプリケーション	
言語	Python
使用パッケージ	FastAPI、Pydantic、 subprocess、os、datetime
コマンドライン ツール	OCI CLI
サーバツール	Uvicorn

環境構築ガイド
手順の流れ

1. 事前準備
    - Object Storage バケットやComputeの作成
2. インスタンスプリンシパルの設定
    - 動的グループとポリシーの作成
3. カスタムアプリケーションのセットアップ
    - カスタムアプリケーションの作成と起動
4. ファイルアクセス
    - 署名付きURLの発行
    - 連携ファイルの格納/取得
5. ファイルの削除

環境構築ガイド
事前準備

- ファイルを格納する Object Storage バケットの作成

  参考：[OCIチュートリアル]-[オブジェクトストレージを使う]

  https://oracle-japan.github.io/ocitutorials/beginners/object-storage/

- 認証認可サーバのセットアップ
  - インスタンスへの接続確認
  - OCI CLIのインストール

  参考：[Oracle Cloud Infrastructureドキュメント]-[クイックスタート]-[CLIのインストール]

  https://docs.oracle.com/ja-jp/iaas/Content/API/SDKDocs/cliinstall.htm

- テスト用ファイルの作成
  - Object Storageへ格納、削除するテスト用ファイルを準備する
  - 本ガイドではFileTest.txtという任意のテキストファイルを使用

環境構築ガイド
インスタンスプリンシパルの設定①

Compute (カスタムアプリケーション) に、Object Storageとのインスタンスプリンシパルを設定します。

(1) 動的グループ (Dynamic Group) の作成

カスタムアプリケーションが稼働するインスタンスを含む動的グループを作成します。

【手順】

1. OCIコンソールのメニューより、**アイデンティティとセキュリティ→アイデンティティ→動的グループ**を選択
2. 「動的グループの作成」をクリック
3. 各フィールドに以下の情報を入力し、「作成」をクリック
   - 名前：動的グループ名
   - 説明：動的グループに関する説明 (オプション)
   - 一致ルール：動的グループに関連付けるインスタンスが存在するコンパートメントのOCIDを指定

   例）instance.compartment.id = 'ocid1.compartment.XXX~~~~~'

参考：[Oracle Cloud Infrastructureドキュメント]-[アイデンティティドメインを使用するIAM]-[動的グループの作成]
https://docs.oracle.com/ja-jp/iaas/Content/Identity/dynamicgroups/To_create_a_dynamic_group.htm

環境構築ガイド
インスタンスプリンシパルの設定②

(2) 動的グループに対するポリシーを作成

作成した動的グループに対してObject Storageの操作 (manage) を許可するIAMポリシーを作成します。

【手順】

1. OCIコンソールのメニューより、**アイデンティティとセキュリティ→アイデンティティ→ポリシー**を選択
2. 適切なコンパートメントを選択し、「ポリシーの作成」ボタンをクリック
3. 各フィールドに以下の情報を入力し、「作成」をクリック
   - 名前：IAMポリシー名
   - 説明：IAMポリシーに関する説明 (オプション)
   - ポリシービルダー：「手動エディタの表示」をクリックし、以下のポリシー構文を入力

```
Allow dynamic-group <作成した動的グループ名> to manage object-family in compartment <対象コンパートメント名>
```

例）Allow dynamic-group test-dynamic-group to manage object-family in compartment test.compartment

※このポリシーはObject Storageに関わるすべての操作 (object-family) を対象としていますが、バケットレベルで制限したい場合は末尾にWhere句を追加してください。

```
where target.bucket.name='<バケット名>'
```

環境構築ガイド
インスタンスプリンシパルの設定③

(3)インスタンスプリンシパルが有効になっていることを確認

Compute上で以下のコマンドを実行してネームスペース名を確認します。

```
$ oci os ns get --auth instance_principal
{
  "data": "testtenancy"
}
```

または

```
$ export OCI_CLI_AUTH=instance_principal
$ oci os ns get
{
  "data": "testtenancy"
}
```

参考：[OCIチュートリアル]-[オンデマンドクラスタ実現のためのインスタンス・プリンシパル認証設定方法]
https://oracle-japan.github.io/ocitutorials/hpc/tech-knowhow/instance-principal-auth/

環境構築ガイド
カスタムアプリケーションのセットアップ①

インスタンスプリンシパルを用いて署名付きURLの取得およびオブジェクトの削除を行うカスタムアプリケーションをCompute上に作成します。

(1) 仮想環境の準備

新しい仮想環境を作成します。

```
$ python3 -m venv env
```

作成した仮想環境を有効化します。

```
$ source env/bin/activate
```

環境構築ガイド
カスタムアプリケーションのセットアップ②

(2) Computeに必要なPythonのパッケージをインストール

FastAPIおよびuvicornをインストールします。

```
$ pip3 install fastapi uvicorn
```

※必要に応じてsudoコマンドを追加してください。

(3) プログラムファイルの作成

テキストエディタを用いてCompute上にカスタムアプリケーションのプログラムファイル「customapp.py」を新規作成します。

本ガイドではnanoエディタを使用しておりますが、任意のエディタで問題ございません。お使いのエディタに読み替えてご参照ください。

```
$ nano customapp.py
```

環境構築ガイド
カスタムアプリケーションのセットアップ③

(4) サンプルコードの入力

「customapp.py」に次ページのサンプルコードを入力します（コピー＆ペースト可能）。
※一部環境に合わせて書き換える必要がある項目がございますのでご注意ください。

(5) プログラムファイルの保存

保存して終了します。
[Ctrl＋O] → [Enter] → [Ctrl＋X]

【参考】nanoエディタコマンド
- カーソル移動：方向キー
- 選択した領域をインデント：[Tab]
- 名前を付けて保存：[Ctrl＋O]
- エディタを閉じる：[Ctrl＋X]

(6) プログラムファイルの権限変更

必要に応じてファイルのパーミッションを変更してください。
本手順では以下の通り変更いただくことを推奨しております。

```
$ chmod 750 customapp.py
```

環境構築ガイド

カスタムアプリケーションのセットアップ④

customapp.py

#赤字で示した以下の項目は、ご自身の環境に合わせて変更してください。

- ネームスペース名
  (インスタンスプリンシパルの設定③で調べたネームスペース名)
- リージョン名 (対象のObject Storageが存在するリージョン名)
- 署名付きURLの有効期限 (分単位で指定可能)

このサンプルコードでは、「署名付きURLの発行」と「ファイルの削除」をまとめてcustomapp.pyとしていますが、どちらかの処理のみを実行する場合にはコードの不要箇所を削除してご利用ください。

また、今回はエラー処理についてはコードに含めておりません。

```
from fastapi import FastAPI
from pydantic import BaseModel
import subprocess
import os
from datetime import datetime, timedelta

app = FastAPI()

# 必要に応じて以下を変更してください
NAMESPACE = "testtenancy"  # ネームスペース名
REGION = "ap-tokyo-1"    # リージョン名
DEFAULT_EXPIRATION_MINUTES = 30  # デフォルトの有効期限（分）
```

1

```
######署名付きURLの発行######
# バケット名/オブジェクト名のデータ型定義
class PreAuthRequest(BaseModel):
    bucket_name: str
    object_name: str = None

# 署名付きURLを生成する関数
def generate_preauth_url_cli(bucket_name: str, object_name: str
= None):
    os.environ["OCI_CLI_AUTH"] = "instance_principal"

    # 現在時刻＋デフォルトの有効期限をUTCで設定
    expiration_time_utc = datetime.utcnow() +
timedelta(minutes=DEFAULT_EXPIRATION_MINUTES)
    expiration_time_str = expiration_time_utc.strftime("%Y-%m-
%dT%H:%M:%SZ")

    command = [
        "oci", "os", "preauth-request", "create",
        "--namespace", NAMESPACE,
        "--bucket-name", bucket_name,
        "--name", "PreAuthRequestSample",
        "--time-expires", expiration_time_str,
        "--query", 'data."access-uri"',
       "--raw-output"
    ]

    if object_name:
        command.extend(["--object-name", object_name, "--access-
type", "ObjectReadWrite"])
    else:
command.extend(["--access-type", "AnyObjectReadWrite"])
```

2

環境構築ガイド
カスタムアプリケーションのセットアップ⑤

```
    result = subprocess.run(command, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, universal_newlines=True, check=True)
    return
f"https://objectstorage.{REGION}.oraclecloud.com{result.stdout
.strip()}", expiration_time_utc

# APIエンドポイント: 署名付きURLの生成
@app.post("/generate_preauth_url")
async def generate_preauth_url(request: PreAuthRequest):
    preauth_url, expiration_time_utc =
generate_preauth_url_cli(request.bucket_name,
request.object_name)

    # UTCをJSTに変換
    expiration_time_jst = expiration_time_utc +
timedelta(hours=9)
    expiration_time_jst_str =
expiration_time_jst.strftime("%Y-%m-%d %H:%M:%S JST")

    return {
        "署名付きURL": preauth_url,
        "有効期限": expiration_time_jst_str
    }
```

3

```
######ファイルの削除######
# バケット名/オブジェクト名のデータ型定義
class DeleteObjectRequest(BaseModel):
    bucket_name: str
    object_name: str

# ファイルを削除する関数
def delete_object_cli(bucket_name: str, object_name: str):
    os.environ["OCI_CLI_AUTH"] = "instance_principal"

    command = [
        "oci", "os", "object", "delete",
        "--namespace", NAMESPACE,
        "--bucket-name", bucket_name,
        "--name", object_name, "--force"
    ]

    result = subprocess.run(command, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, universal_newlines=True, check=True)
    return {"message": f"Object '{object_name}' in bucket
'{bucket_name}' has been deleted successfully."}

# APIエンドポイント: ファイルの削除
@app.post("/delete_object")
async def delete_object(request: DeleteObjectRequest):
    return delete_object_cli(request.bucket_name,
request.object_name)
```

4

環境構築ガイド
カスタムアプリケーションのセットアップ⑥

(7) アプリケーションを起動

以下のコマンドを実行してカスタムアプリケーションを起動します。

```
$ uvicorn <サンプルコード名>:app --host <ホストIP> --port <任意のポート番号>
```

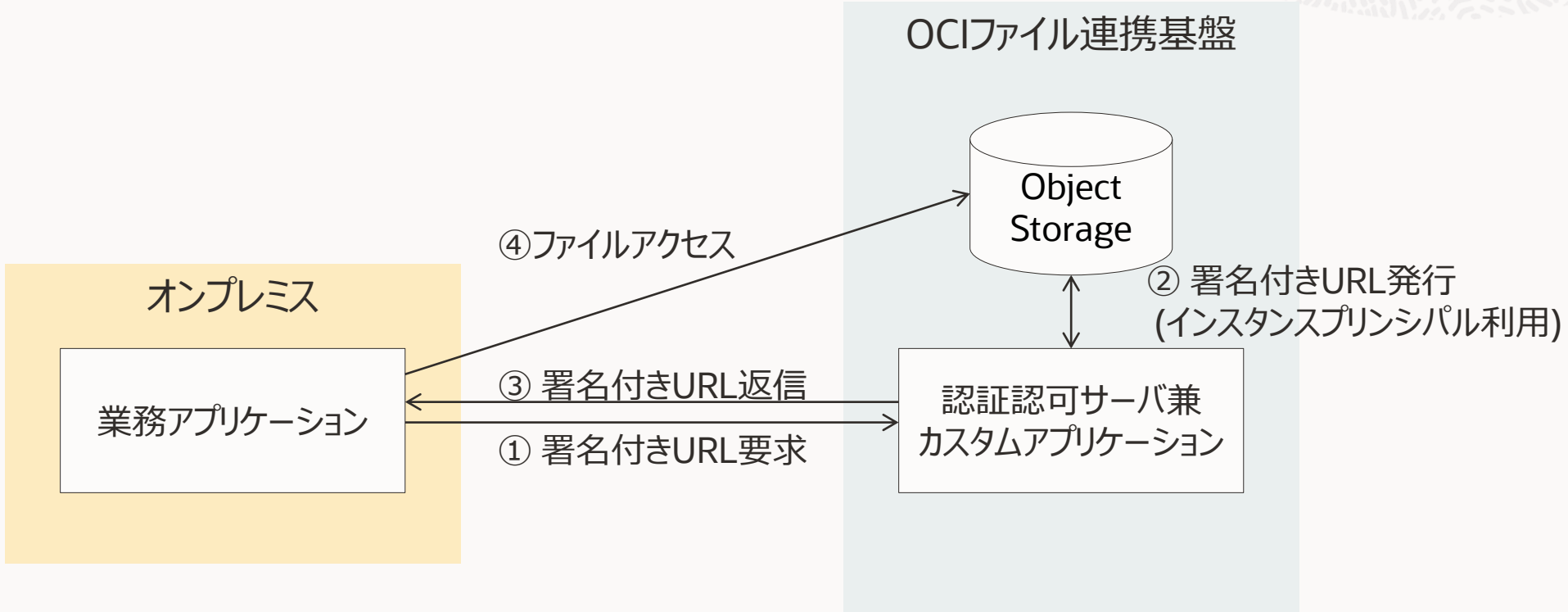
※ここで指定しているポート番号からのアクセスがComputeが存在するサブネットのセキュリティリストで許可されている必要があります。
事前にイングレス・ルールの[宛先ポート範囲]に指定するポート番号を追加してTCPトラフィックを許可してください。

実行例）

```
$ uvicorn customapp:app --host 0.0.0.0 --port 8000
INFO:     Started server process [512478]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

カスタムアプリケーションを起動した状態で以降の業務アプリケーション側の操作を行ってください。

環境構築ガイド
ファイルアクセス_手順イメージ



①署名付きURL要求
②署名付きURL発行
③署名付きURL返信
④ファイルアクセス

【署名付きURLの発行】

【連携ファイルの格納・取得】

環境構築ガイド
ファイルアクセス_【署名付きURLの発行】

- **業務アプリケーションから署名付きURLを取得**

  ローカルPCのコマンドプロンプトからリクエストを送信します。

```
> curl -X POST http://<Compute(カスタムアプリケーション)のパブリックIP>:<任意のポート番号>/generate_preauth_url -H
"Content-Type: application/json" -d "{¥"bucket_name¥": ¥"<バケット名>¥", ¥"object_name¥": ¥"<ファイル名>¥"}"
```

実行例）

```
> curl -X POST http://167.234.222.223:8000/generate_preauth_url -H "Content-Type: application/json" -d
"{¥"bucket_name¥": ¥"ObjectStorage_FileTest¥", ¥"object_name¥": ¥"FileTest.txt¥"}"

{"署名付きURL":"https://objectstorage.ap-tokyo-1.oraclecloud.com/p/HBd2zWpmg7DP3mCXQtIfOEoe-
EhUNFZ1kyVxzwaIt/n/testtenancy/b/ObjectStorage_FileTest/o/FileTest.txt","有効期限":"2024-11-05 6:24:50 JST"}
```

事前認証済リクエスト

※コンソール上でも事前承認済リクエストが作成されていることが確認できる

環境構築ガイド
ファイルアクセス_【署名付きURLの発行】

オブジェクト名まで指定せず、バケット単位で署名付きURLを発行することも可能です。

```
> curl -X POST http://<Compute(カスタムアプリケーション)のパブリックIP>:<任意のポート番号>/generate_preauth_url -H
"Content-Type: application/json" -d "{¥"bucket_name¥": ¥"<バケット名>¥"}"
```

実行例）

```
> curl -X POST http://167.234.222.223:8000/generate_preauth_url -H "Content-Type: application/json" -d
"{¥"bucket_name¥": ¥"ObjectStorage_FileTest¥"}"

{"署名付きURL":"https://objectstorage.ap-tokyo-1.oraclecloud.com/p/HBd2zWpmg7DP3mCXQtIfOEoe-
EhUNFZ1kyVxzwaIt/n/testtenancy/b/ObjectStorage_FileTest/o/","有効期限":"2024-11-05 6:24:50 JST"}
```

事前認証済リクエスト

※事前承認済みリクエストのターゲットが「バケット」でも指定できることが確認できる

環境構築ガイド
ファイルアクセス_【連携ファイルの格納・取得】

業務アプリケーションから連携するファイルを格納および取得します。

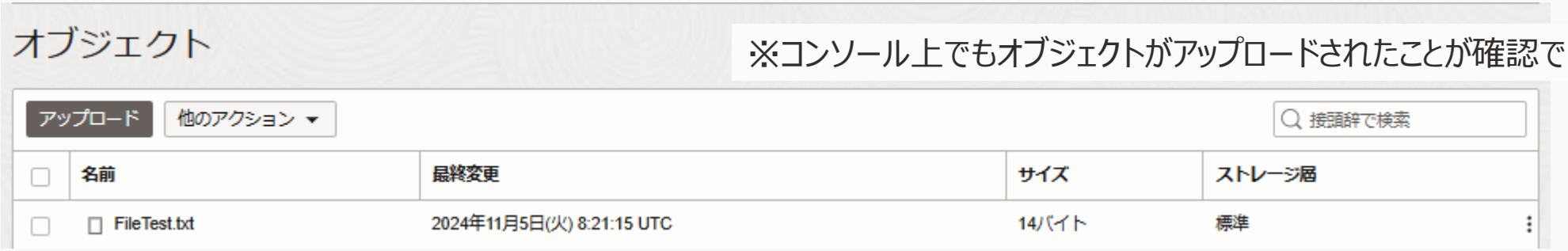
- **ファイルの格納**

```
> curl -X PUT --data-binary '＜ローカルのファイル名＞' ＜署名付きURL＞
```

※ファイルの格納（オブジェクトのアップロード）に使用する署名付きURLはオブジェクト名まで指定して発行する必要がございます。

実行例）

```
> curl -X PUT --data-binary 'FileTest.txt' https://objectstorage.ap-tokyo-1.oraclecloud.com/p/jIXq_3FGKs-iNjd4zFpj_o2DSc1CtllwM8c0hj_HdIuP_m1liBjG28cFZK/n/testtenancy/b/ObjectStorage_FileTest/o/FileTest.txt
```



環境構築ガイド
ファイルアクセス_【連携ファイルの格納・取得】

業務アプリケーションから連携するファイルを格納および取得します。

- **ファイルの取得**

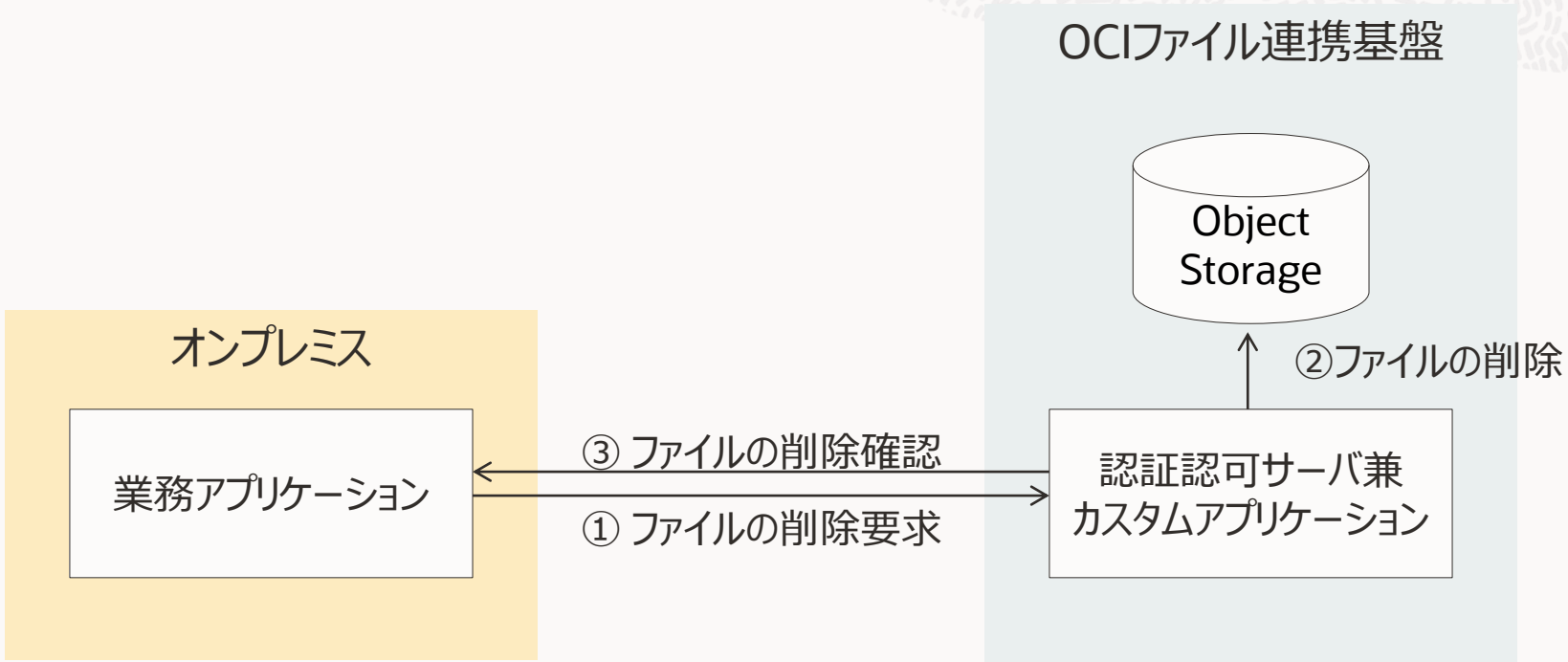
```
> curl -X GET <署名付きURL> -o "<ダウンロード先のパス>"
```

実行例）

```
> curl -X GET https://objectstorage.ap-tokyo-
1.oraclecloud.com/p/RJ5NQzz6rlpkGEW6qa5L3UqkKIWdLRhqiNwUe5HBEUbcyInpwfsNZpXtdmN/n/testtenancy/b/ObjectStor
age_FileTest/o/FileTest.txt -o "C:¥Users¥Filetest.txt"

% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    14  100    14    0     0     22      0 --:--:-- --:--:-- --:--:--    23
```

環境構築ガイド
ファイルの削除_手順イメージ



①ファイルの削除要求
②ファイルの削除
③ファイルの削除確認

環境構築ガイド
ファイルの削除

- **業務アプリケーションからファイルの削除を要求**
  ローカルPCのコマンドプロンプトからリクエストを送信します。

```
> curl -X POST http:// <インスタンスのパブリックIP> :<任意のポート番号>/delete_object -H "Content-Type:
application/json" -d " {¥"bucket_name¥": ¥" <バケット名> ¥", ¥"object_name¥": ¥" <ファイル名> ¥"} "
```

実行例）

```
> curl -X POST http://167.234.222.223:8000/delete_object -H "Content-Type: application/json" -d
"{¥"bucket_name¥": ¥"ObjectStorage_FileTest¥", ¥"object_name¥": ¥"FileTest.txt¥"} "

{"message":"Object 'FileTest.txt' in bucket 'ObjectStorage_FileTest' has been deleted successfully."}
```

以上で一連のファイル連携手順について動作確認がとれました。

環境構築ガイド
ファイル削除時に確認メッセージを出す追加手順①

前述の手順に加え、ファイル削除時に削除確認をするための確認メッセージを表示させるためには次の手順をご参照ください。

※想定されるユースケース

：オブジェクトストレージ上のファイルを運用者が手動で削除する際、削除確認のための確認メッセージを表示したい場合　など

※この手順はローカルPCでWindowsのPowerShellを使用する想定で作成しているため、お使いの環境によってはそのまま実行できない可能性がございます。適宜お使いの環境に合わせて変更してお試しいただきますようお願いいたします。

(1) プログラムファイルの作成

作成したcustomapp.pyのファイル削除手順を書き換え、あるいはファイル名を別途指定して新規作成します。

今回は新規のプログラムファイル「deletemessage.py」を新規作成します。

```
$ nano deletemessage.py
```

(2) サンプルコードの入力

「deletemessage.py」に次ページのサンプルコードを入力します。

※一部環境に合わせて書き換える必要がある項目がございますのでご注意ください。

環境構築ガイド

ファイル削除時に確認メッセージを出す追加手順②

deletemessage.py

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import subprocess
import os

app = FastAPI()

# バケット名/オブジェクト名のデータ型定義
class RequestDeleteObjectRequest(BaseModel):
    bucket_name: str
    object_name: str

class ConfirmDeleteObjectRequest(BaseModel):
    bucket_name: str
    object_name: str
    confirmation: str  # 削除確認のためのフィールド

# ファイルを削除する関数
def delete_object_cli(bucket_name: str, object_name: str):
    os.environ["OCI_CLI_AUTH"] = "instance_principal"
```

1

```
    namespace = "testtenancy"  # ネームスペース名
    command = [
        "oci", "os", "object", "delete",
        "--namespace", namespace,
        "--bucket-name", bucket_name,
        "--name", object_name, "--force"
    ]
    try:
        result = subprocess.run(command, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, universal_newlines=True, check=True)
        return {"message": f"Object '{object_name}' in bucket
'{bucket_name}' has been deleted successfully."}
    except subprocess.CalledProcessError as e:
        raise HTTPException(status_code=500, detail=f"OCI CLI
Error: {e.stderr.strip()}")

# APIエンドポイント: 削除確認と実行
@app.post("/request_delete_object")
async def request_delete_object(request:
RequestDeleteObjectRequest):
    # 確認メッセージの処理
    return {"message": f"Do you want to delete object
'{request.object_name}' in bucket '{request.bucket_name}'?"}

@app.post("/confirm_delete")
async def confirm_delete(request: ConfirmDeleteObjectRequest):
    if request.confirmation.lower() != "yes":
        return {"message": "Deletion cancelled by the user."}
    return delete_object_cli(request.bucket_name,
request.object_name)
```

2

環境構築ガイド
ファイル削除時に確認メッセージを出す追加手順③

(3) アプリケーションを起動

以下のコマンドを実行してカスタムアプリケーションを起動します。

```
$ uvicorn <サンプルコード名>:app --host <ホストIP> --port <任意のポート番号>
```

実行例）

```
$ uvicorn deletemessage:app --host 0.0.0.0 --port 8000
INFO:     Started server process [512478]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

カスタムアプリケーションを起動した状態で以降の業務アプリケーション側の操作を行ってください。

環境構築ガイド
ファイル削除時に確認メッセージを出す追加手順④

(4) メッセージを表示するスクリプトの作成

業務アプリケーション側に確認メッセージを呼び出すようなスクリプトを作成する。

例：PowerShell使用の場合、以下のコードを「delete_object.ps1」として格納

```
param (
    [string]$server_ip,
    [int]$port,
    [string]$bucket_name,
    [string]$object_name
)

# 削除確認のリクエストを送信
$response = Invoke-RestMethod -Method Post -Uri "http://${server_ip}:${port}/request_delete_object" -ContentType "application/json"
-Body (@{bucket_name=$bucket_name; object_name=$object_name} | ConvertTo-Json)

# 確認メッセージを表示
Write-Host $response.message
$user_input = Read-Host "Do you want to proceed with deletion? (yes/no)"

# 入力が "yes" の場合に削除リクエストを送信
if ($user_input -eq "yes") {
    Invoke-RestMethod -Method Post -Uri "http://${server_ip}:${port}/confirm_delete" -ContentType "application/json" -Body
(@{bucket_name=$bucket_name; object_name=$object_name; confirmation="yes"} | ConvertTo-Json)
} else {
    Write-Host "Deletion cancelled by the user."
}
```

環境構築ガイド
ファイル削除時に確認メッセージを出す追加手順⑤

(5)業務アプリケーションからファイルの削除を要求

「delete_object.ps1」を格納したディレクトリで次のコマンドを実行します。

ファイルを削除するか、[yes/no]で問われるためファイル名等確認した上で[yes]と入力します。

```
> powershell -File delete_object.ps1 -server_ip ＜インスタンスのパブリックIP＞ -port <任意のポート番号> -bucket_name
"<バケット名>" -object_name "<ファイル名>"
```

実行例）

```
> powershell -File delete_object.ps1 -server_ip 167.234.222.223 -port 8000 -bucket_name
"ObjectStorage_FileTest" -object_name "FileTest.txt"
Do you want to delete object 'FileTest.txt' in bucket 'ObjectStorage_FileTest'?
Do you want to proceed with deletion? (yes/no): yes

message
-------
Object 'FileTest.txt' in bucket 'ObjectStorage_FileTest' has been deleted successfully.
```

環境構築ガイド
参考資料

[Oracle Cloud Infrastructureドキュメント]-[アイデンティティ・ドメインのないIAM]-[インスタンスからのサービスのコール]
https://docs.oracle.com/ja-jp/iaas/Content/Identity/Tasks/callingservicesfrominstances.htm

[OCI活用資料集]-[IDおよびアクセス管理 詳細 Identity and Access Management Level 200 2024年6月]
https://oracle-japan.github.io/ocidocs/services/governance%20and%20administration/iam-200/

[Oracle Cloud Infrastructureドキュメント]-[OCIFSユーティリティ]
https://docs.oracle.com/ja-jp/iaas/oracle-linux/ocifs/index.htm

[Oracle Cloud Infrastructure Blog]-[Configuring OCI Object Storage access control policies]
https://blogs.oracle.com/cloud-infrastructure/post/configuring-oci-object-storage-access-control-policies

環境構築ガイド
用語ガイド

インスタンス・プリンシパル

インスタンスを認可者 (またはプリンシパル) にでき、サービス・リソースに対するアクションを実行できるIAMサービス機能。各コンピュート・インスタンスは、自身のアイデンティティを持ち、追加された証明書を使用して認証を行います。これらの証明書は自動的に作成され、インスタンスに割り当てられてローテーションされ、ホストに資格証明を配布してローテーションする必要がなくなります。

API キー

アプリケーションやユーザーがAPIを通じてクラウドリソースにアクセスするために使用する資格情報。APIキーは一意のキー (通常は一連の文字列やファイル形式) として生成され、認証と認可の手段として機能します。APIキーは、特定のユーザーやアプリケーションに紐付けられ、APIリクエストを安全に行うための認証情報として使用されます。

事前承認済リクエストURL (Pre-Authenticated Request: PAR)

OCIのオブジェクトストレージ内の特定のファイルやバケットへのアクセスを、一時的に他の人やアプリケーションに許可するためのURL。たとえば、特定のファイルを共有したい場合、このPARを発行すると、相手は追加の認証なしでそのファイルをダウンロードできます。PARは有効期限を設定できるため、一定期間が過ぎると自動的に無効になります。リンクには、読み取り専用や読み書き許可などのアクセス権限を設定でき、不要なときは無効にして使用を制限することも可能です。

OCIにおける「署名付きURL」とはこの事前承認済リクエストURLのことを指します。


ORACLE