ORACLE

セキュアなWeb APIの作り方

~cert-manager & Keycloak~

Oracle Cloud Hangout Café – Season 7 #4

Shuhei Kawamura
Cloud Architect
Oracle Digital, Oracle Corporation Japan
September 6, 2023

@shukawam

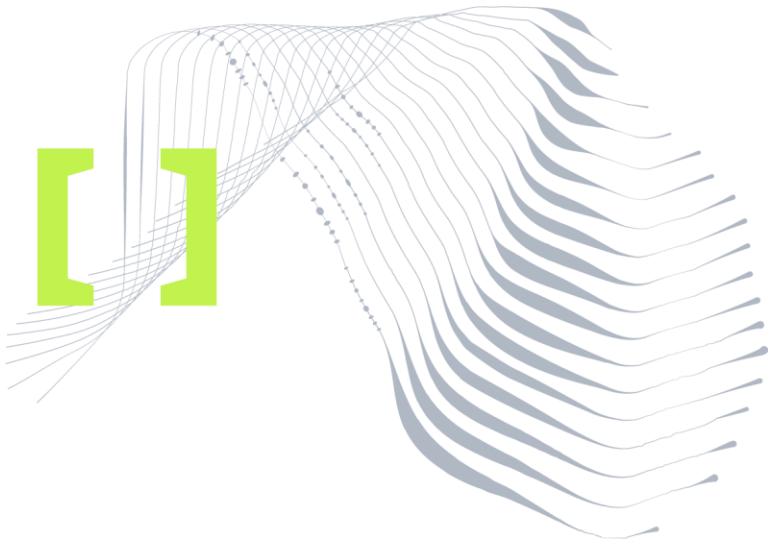
X/GitHub/Qiita

川村 修平 (Shuhei Kawamura)

- 所属
  - 日本オラクル株式会社
  - Oracle Digital
- 普段の業務
  - Digital Nativeなお客様を技術面でご支援
- コミュニティ
  - OCHaCafe
  - CloudNative Days – Observability

Agenda

1. セキュアなWeb API
2. cert-manager
3. Keycloak



セキュアなWeb API

OWASP – Open Worldwide Application Security Project

OWASP

- 組織が信頼できるアプリケーションやAPIを開発、維持できることを目的としたオープン・コミュニティのこと

OWASP API Security Project

- 潜在的にセンシティブなAPIをソフトウェア提供の一部としてデプロイする組織が増え続けていることに対処するために設計されたプロジェクトのこと
- 安全でないAPIの潜在的なリスクを強調し、これらのリスクをどのように軽減できるかを説明することで、ソフトウェア開発者とセキュリティ評価者に価値を提供することを目指す

OWASP Top 10 API Security Risks – 2023

API1:2023 – オブジェクトレベルの認可の不具合
API2:2023 – 認証の不具合
API3:2023 – オブジェクト・プロパティレベルの認可の不具合
API4:2023 – 無制限のAPIリソース消費
API5:2023 – 関数レベルの認可不具合
API6:2023 – 機密性の高いビジネスフローへの無制限アクセス
API7:2023 – サーバーサイドリクエストフォージェリ
API8:2023 – セキュリティの誤設定
API9:2023 – 不適切な在庫管理
API10:2023 – APIの非安全な消費

今日のテーマに関連！
（+ 証明書管理）

参考: https://owasp.org/API-Security/editions/2023/en/0x11-t10/

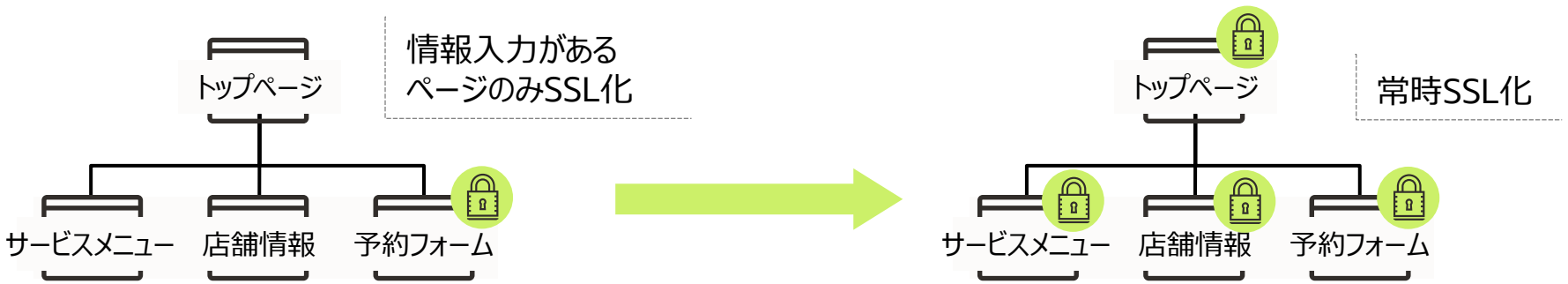
cert-manager

SSL/TLSとは

SSL(Secure Socket Layer)/TLS(Transport Layer Security)

- インターネットなどコンピュータ間の通信において、セキュアな通信を実現するためのプロトコル
- 元々は、SSLというプロトコルが利用されていたが、脆弱性が発見されたため、新たにTLSが設計
- SSLという名称が広く普及していることから、実際にはTLSを使用している場合でもSSLやSSL/TLSと表記されることも多い

- 以前は、導入に当たってのコストや、応答速度の低下の懸念があり、導入をしない／一部のみ導入するなどの対応が行われていたが、SSL化されていないWebサイトがブラウザ上で警告表示されるようになったり、SSL化されているWebサイトが検索サイトで上位に表示されるなどの変更があり“常時SSL“が求められるようになった



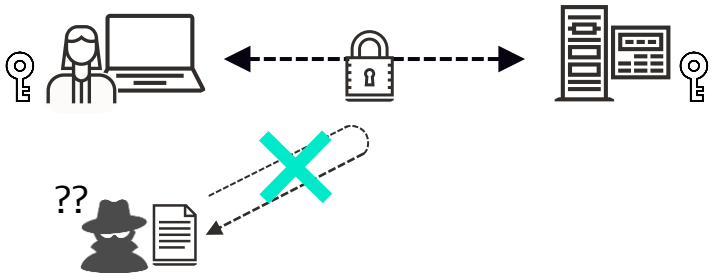
SSL/TLSの機能・役割

確かな相手と安全に通信するための技術

通信内容の暗号化

1.「鍵」を用いてデータを暗号化

2.「鍵」を用いてデータを復号

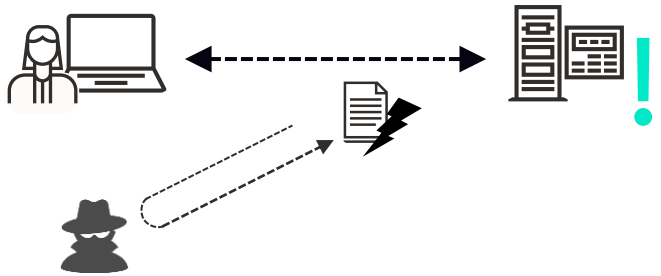


第三者が盗聴してもデータの解読が不可

改ざんの検出

1. データからハッシュ値を計算

2. 取得したデータからハッシュ値を計算・照合

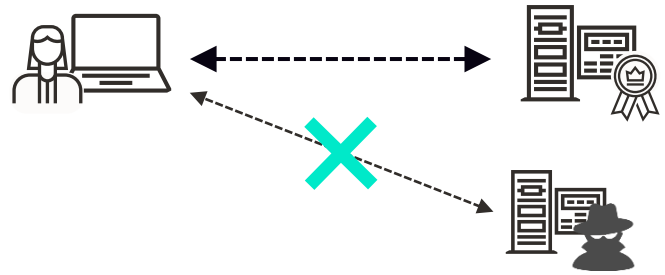


第三者が改ざんした場合、元のハッシュ値と一致しなくなり、改ざんの検知が可能

通信相手の認証

1. 電子証明書を要求
3. 証明書を検証

2. 電子証明書を送付



認証局に認められた電子証明書とそれに対応する鍵を持たないため、なりすますことが出来ない

認証局(CA)とデジタル証明書

認証局(CA)

- 公開鍵とその所有者を証明するための電子証明書を発行する機関のこと
- 認証局自体も、自身が信頼できる認証局であることを証明するために、より上位の認証局によって正当性を保証
- 最上位の認証局 → ルート認証局、そのほかの認証局 → 中間認証局／下位認証局

デジタル証明書

- 公開鍵の所有者を証明する証(サーバー証明書、クライアント証明書、ルート証明書、etc.)
- 証明書の標準フォーマット: X.509, SPKI, etc.
- 公開鍵の所有者から提出された証明書署名要求(= **C**ertificate **S**igning **R**equest)を元に、認証局の秘密鍵によってデジタル署名されることで発行される

大規模な環境になると、証明書発行・管理の一連のプロセスを人手で実施するのは辛くなってくるので、自動化する

→ **A**utomatic **C**ertificate **M**anagement **E**nvironment

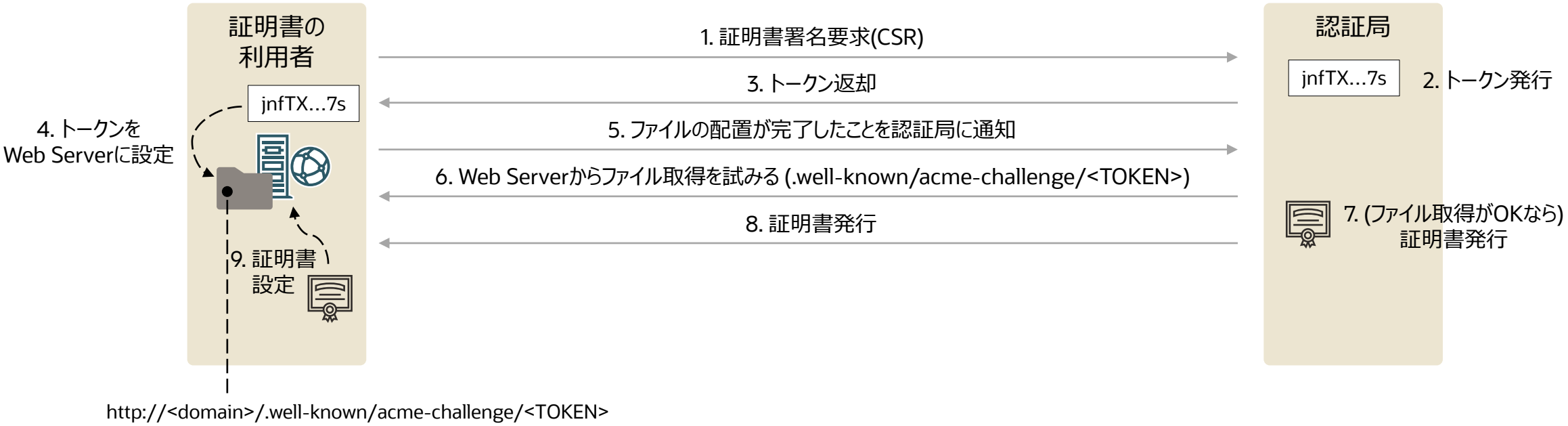
ACME

ACME(**A**utomatic **C**ertificate **M**anagement **E**nvironment)

- RFC 8555
- 証明書管理における一連の手続きを自動化するためのプロトコルを規定
  - 鍵ペアの作成
  - 証明書署名要求の作成、認証局への送信
  - ドメイン名利用権の検証
  - 証明書の設定、更新
- ACME標準で定義されているチャレンジを使用し、証明しようとしているドメイン名が制御下にあることを検証
  - HTTP-01チャレンジ
  - DNS-01チャレンジ
  - TNS-ALPN-01チャレンジ（割愛）

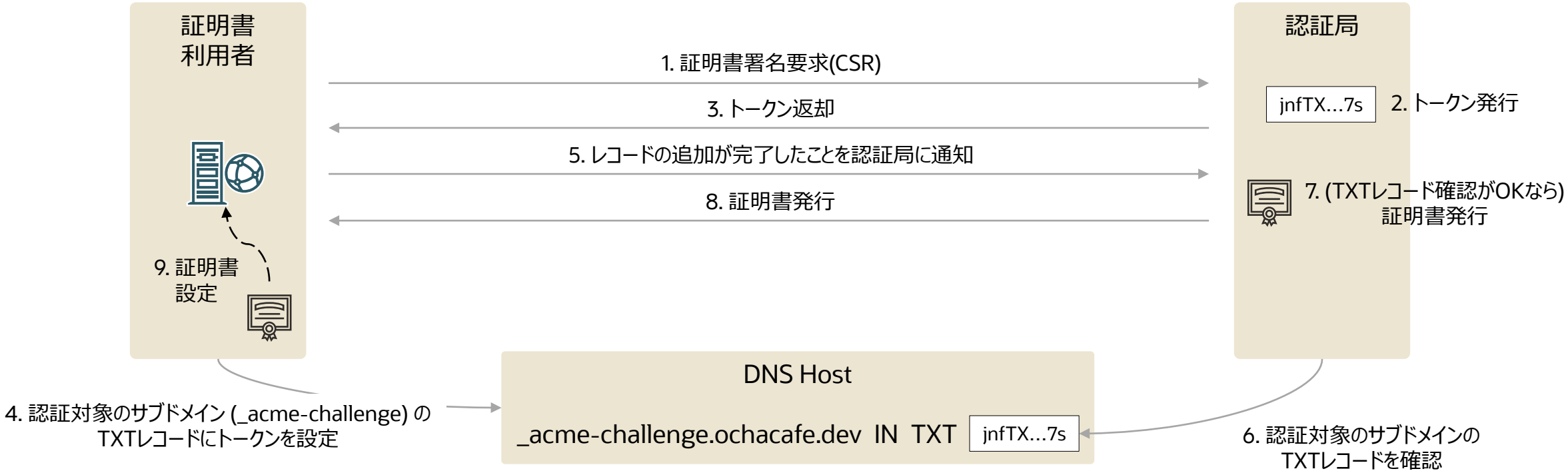
HTTP-01チャレンジ

- 最も多く使われているチャレンジ
- ドメイン設定に関する知識が不要で、証明書の取得が簡単に自動化可能
- 80番ポートでのみ実行可能（=80番ポートがブロックされている場合には動作しない）
- ワイルドカード証明書の発行は不可



DNS-01チャレンジ

- HTTP-01が動作しない環境下でも適切に動作する
- ワイルドカード証明書の発行が可能
- DNS-01に対応したDNSプロバイダでのみ動作する



ACMEクライアントの実装

※https://letsencrypt.org/ja/docs/client-options/ から一部引用

Bash
GetSSL
acme.sh
dehydrated
ght-acne.sh
bacme
wdfcert.sh

C
OpenBSD acme-client
uacme
acme-client-portable
Apache httpd
mod_md
CycloneAcme

C++
acme-lw
esp32-acme-client

Clojure
certificaat

Configuration management tools
Ansible acme_certificate module
Terraform ACME Provider
Ansible collection: acme

D
oacme-lw-d

Domino
CertMatica
HCL Domino

Docker
Crypt::LE
acme.sh
letsproxy

Go
Caddy
Lego
acmetool
Lets-proxy2
autocert
Traefik
ACMEz
Step CLI
J8a
certmanager ← 今日はこれ

HAProxy
HAProxy client

Java
PJAC
ManageEngine Key Manager Plus

Kubernetes
KCert

Lua
Mako Server's ACME Plugin

Microsoft Azure
Azure WebApp SSL Manager
App Service Acmebot
Key Vault Acmebot
Az-Acme

nginx
njs-acme
lua-resty-auto-ssl
Nginx ACME
lua-resty-acme

Node.js
Greenlock for Express.js
acme-http01-azure-key-vault-middleware

Openshift
openshift-acme

Perl
acme
Crypt::LE

PHP
kelunik/acme-client
Hiawatha
FreeSSL.tech Auto
Yet another ACME client
itr-acme-client PHP library
Acme PHP
RW ACME client

Python
ACME Tiny
simp_le
acmebot
sewer
acme-dns-tiny (Python 3)
Automatoes
acertmgr
acme-cert-tool
serverPKI

Rust
ACMEd
acme-redirect

Windows / IIS
Crypt::LE
win-acme
Posh-ACME
Certes
ACME-PS
kelunik/acme-client
Certify The Web
WinCertes Windows client
GetCert2
TekCERT

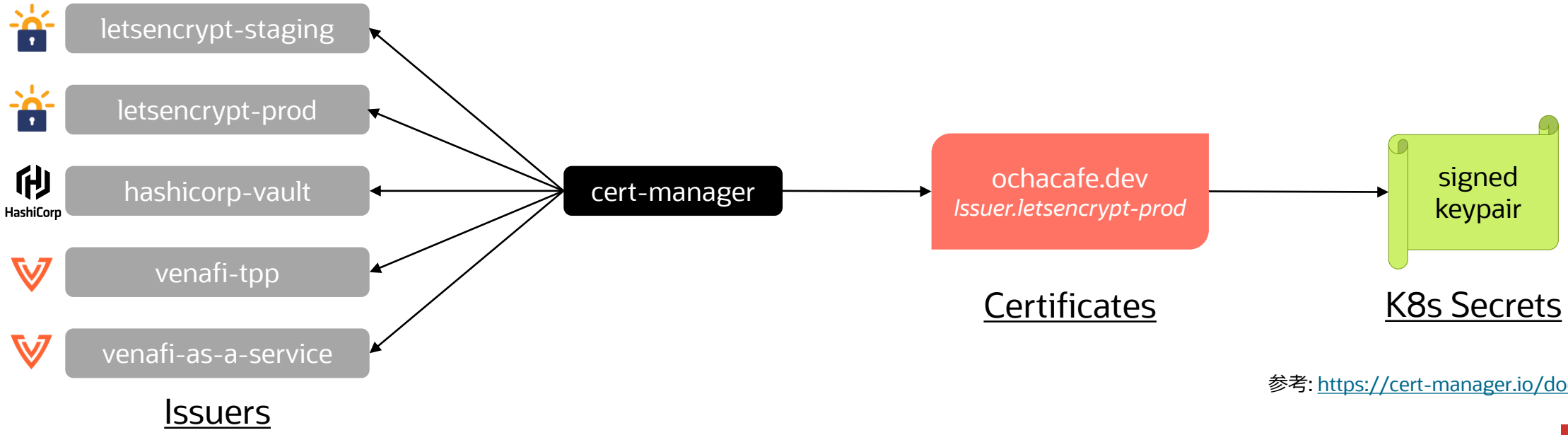
Server
Certera

※その他、ライブラリや連携プロジェクト多数…

cert-manager

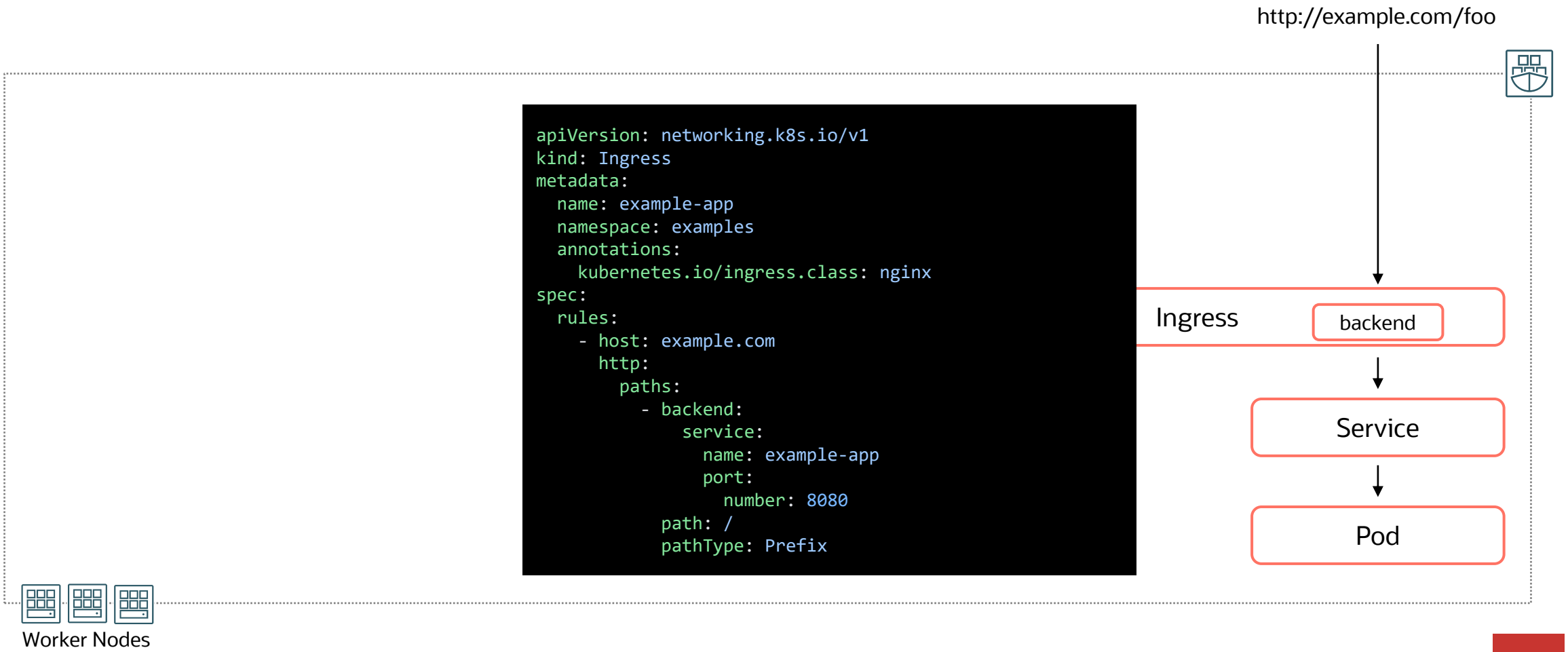
https://github.com/cert-manager/cert-manager

- *cert-manager is the de facto standard for X.509 certificates in Kubernetes environments*
  - *by Ricardo Torres, Chief Engineer of Open Source & Cloud Native at The Boeing Company*
- Jetstackが開発したKubernetes, OpenShiftでX.509証明書を扱うためのカスタム・コントローラー
  - CNCF incubating project
  - 証明書の取得、更新、利用のプロセスを簡素化



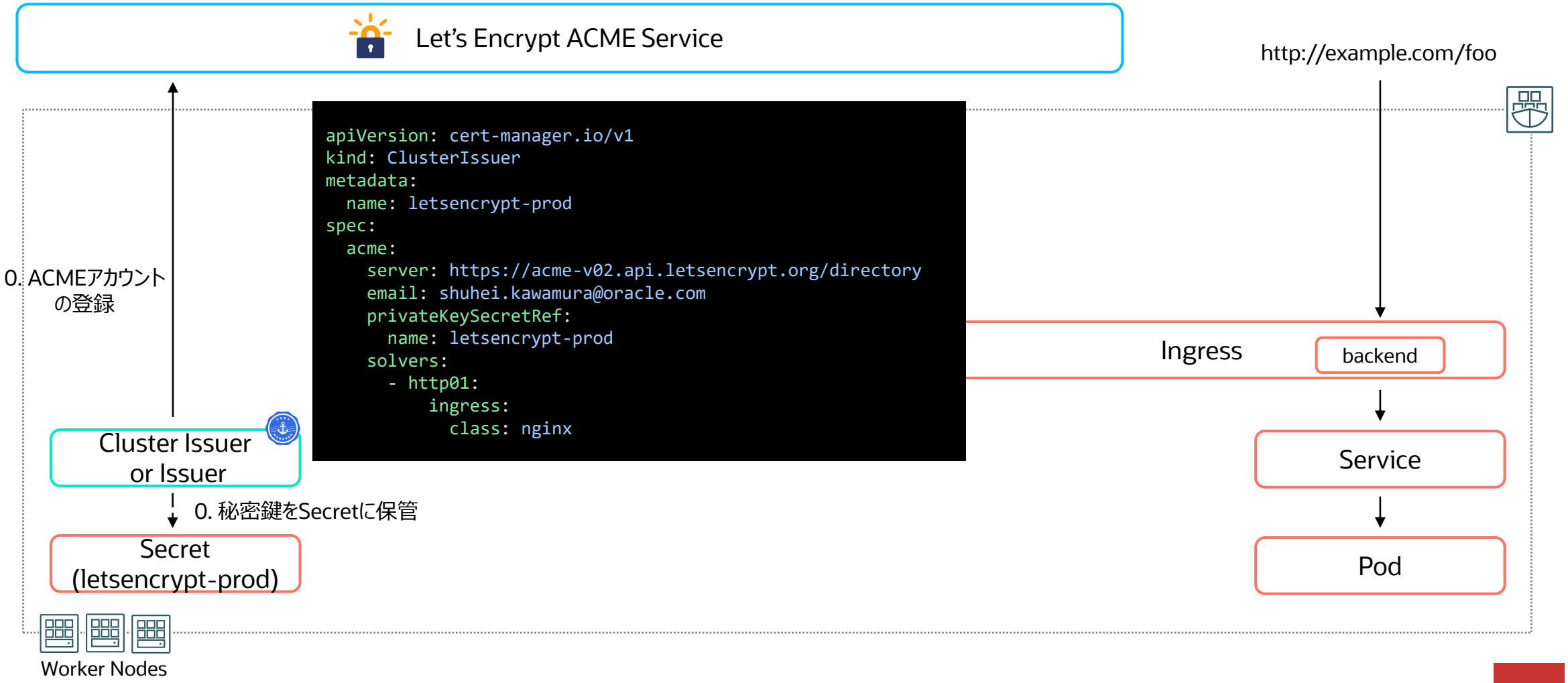
参考: https://cert-manager.io/docs/

cert-manager architecture overview w/ Let's Encrypt

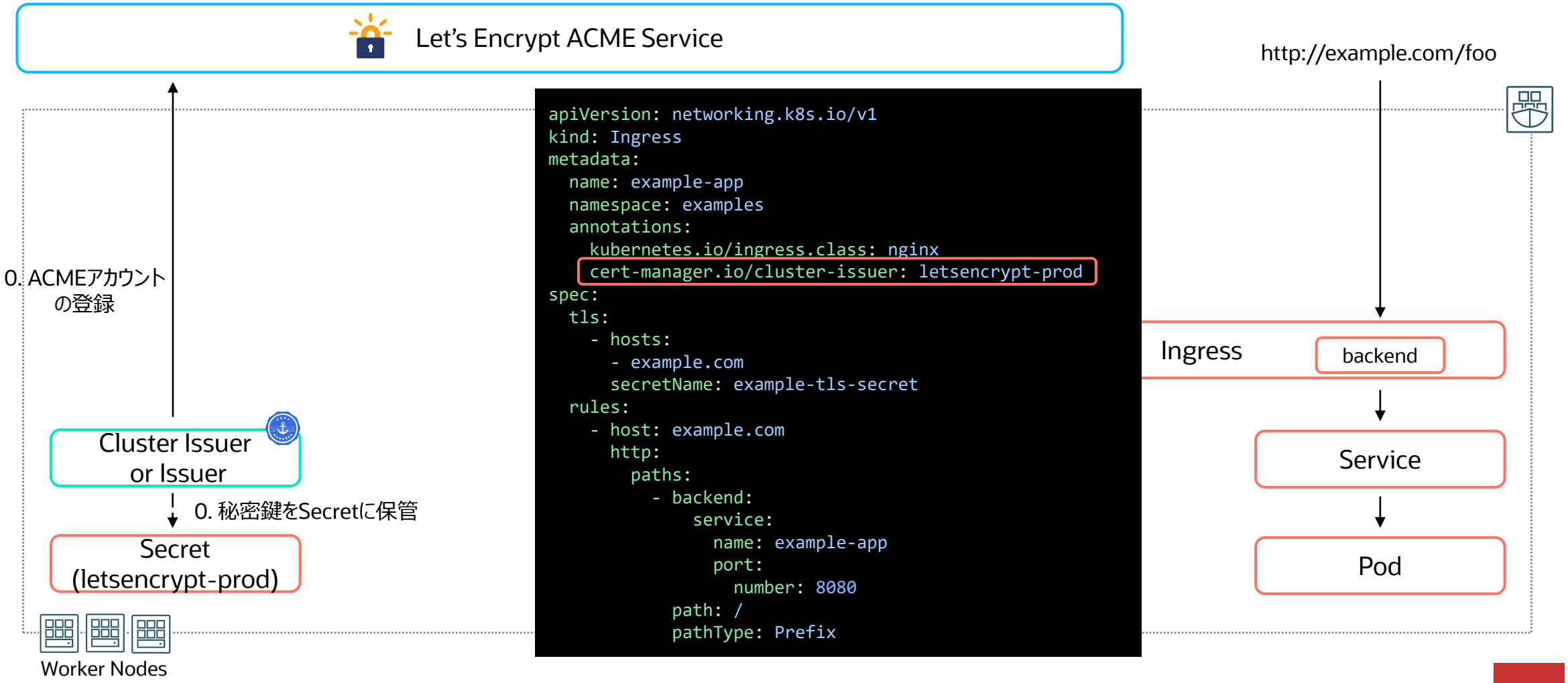



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-app
  namespace: examples
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
    - host: example.com
      http:
        paths:
          - backend:
              service:
                name: example-app
                port:
                  number: 8080
            path: /
            pathType: Prefix
```

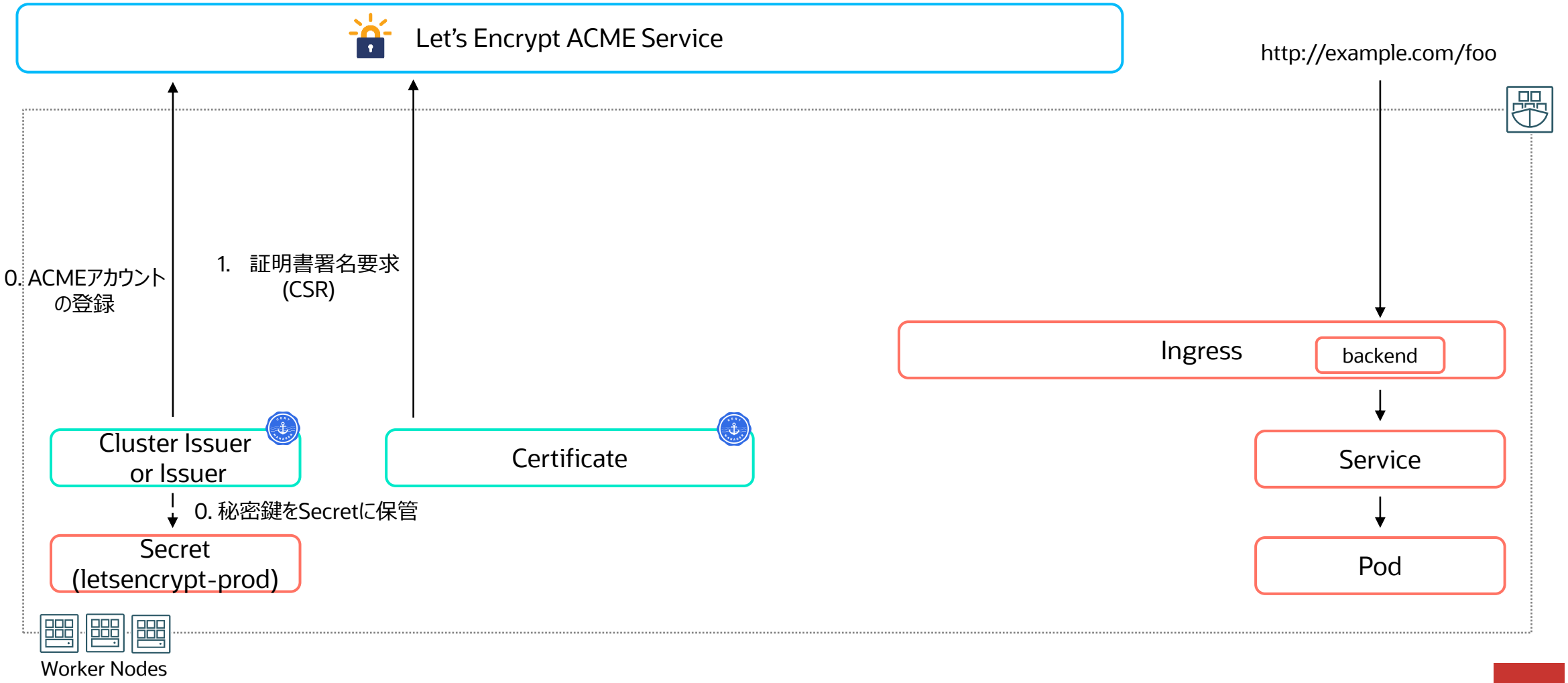
cert-manager architecture overview w/ Let's Encrypt



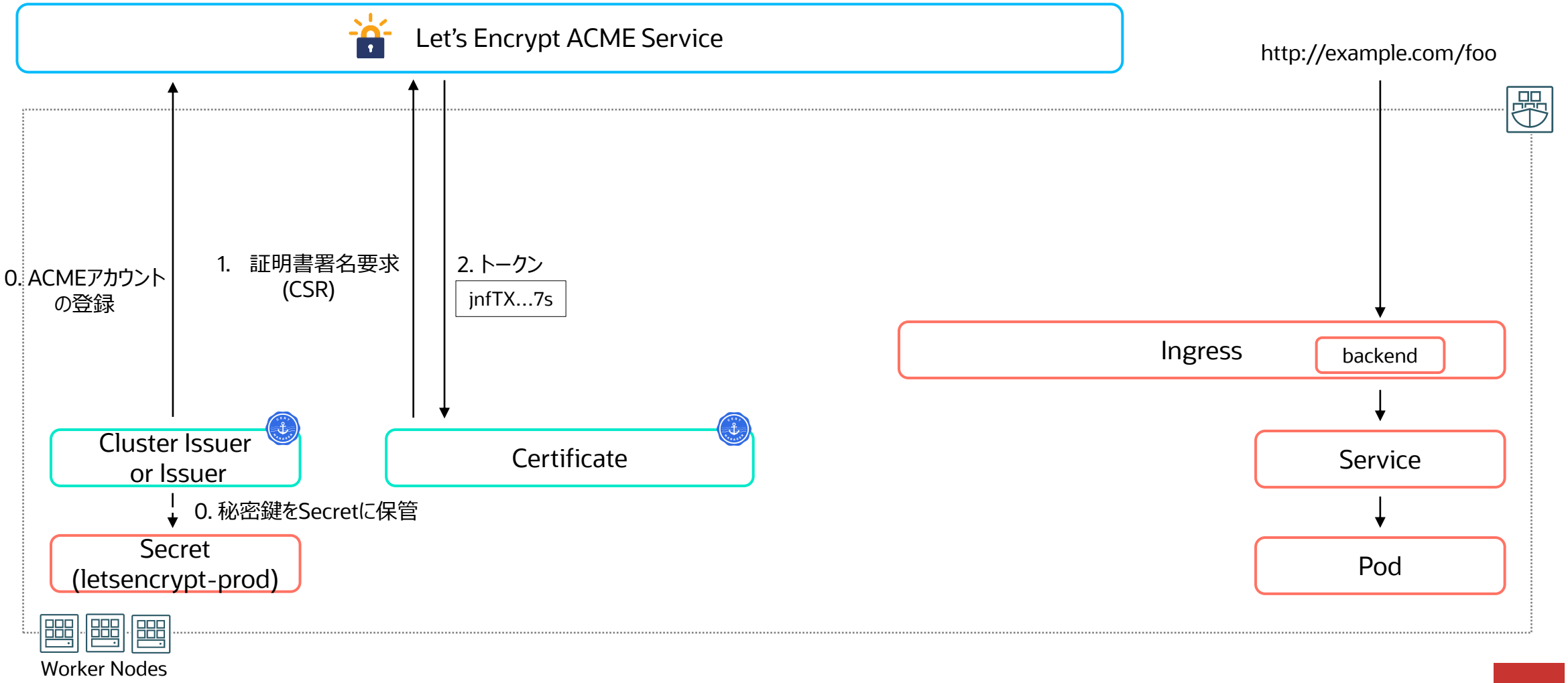
cert-manager architecture overview w/ Let's Encrypt



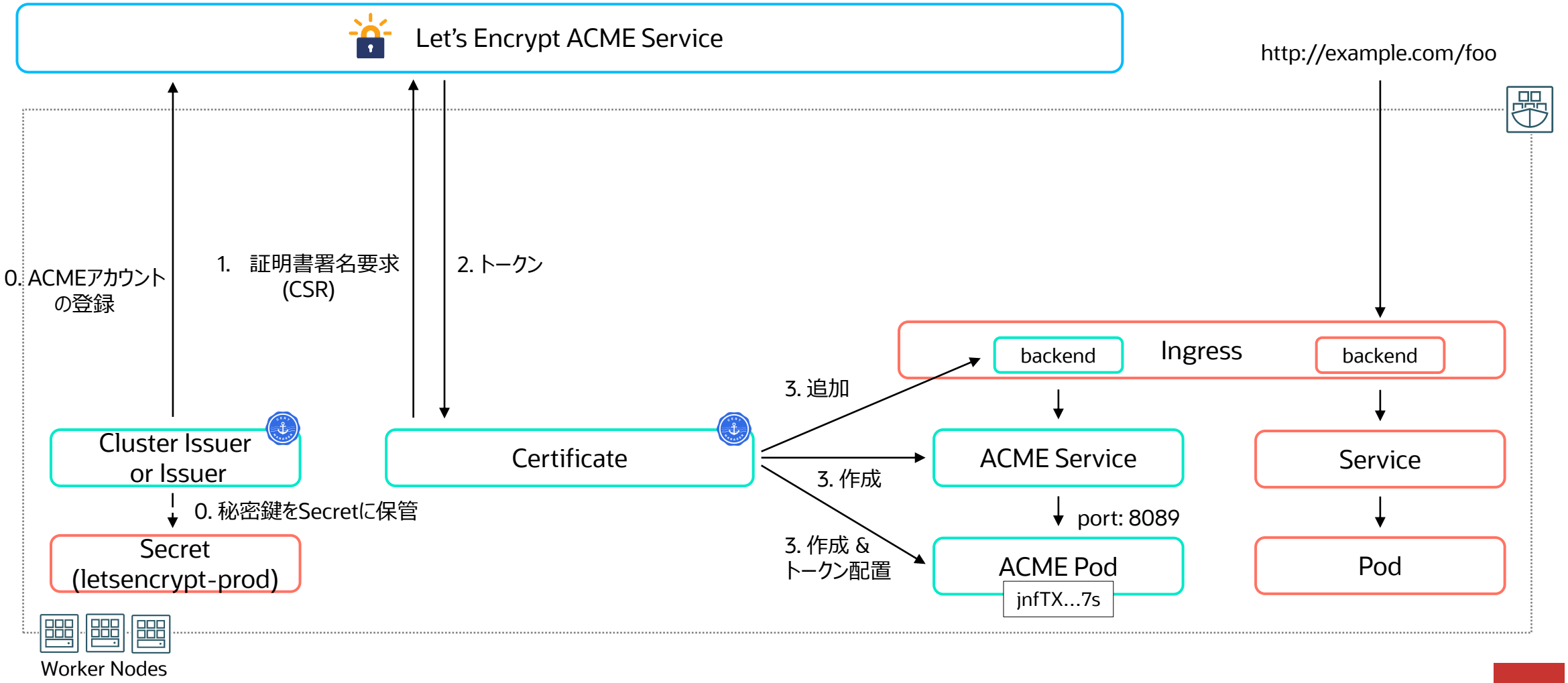
cert-manager architecture overview w/ Let's Encrypt



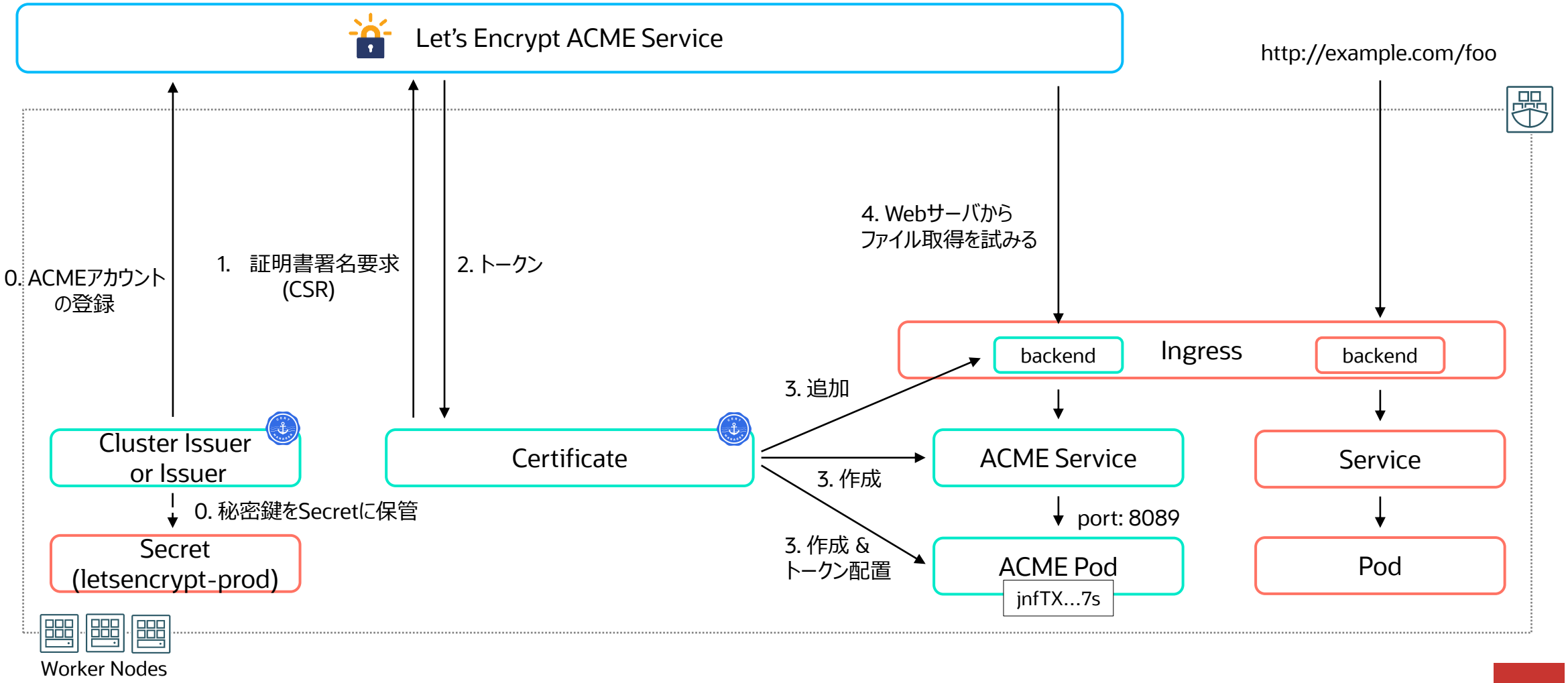
cert-manager architecture overview w/ Let's Encrypt



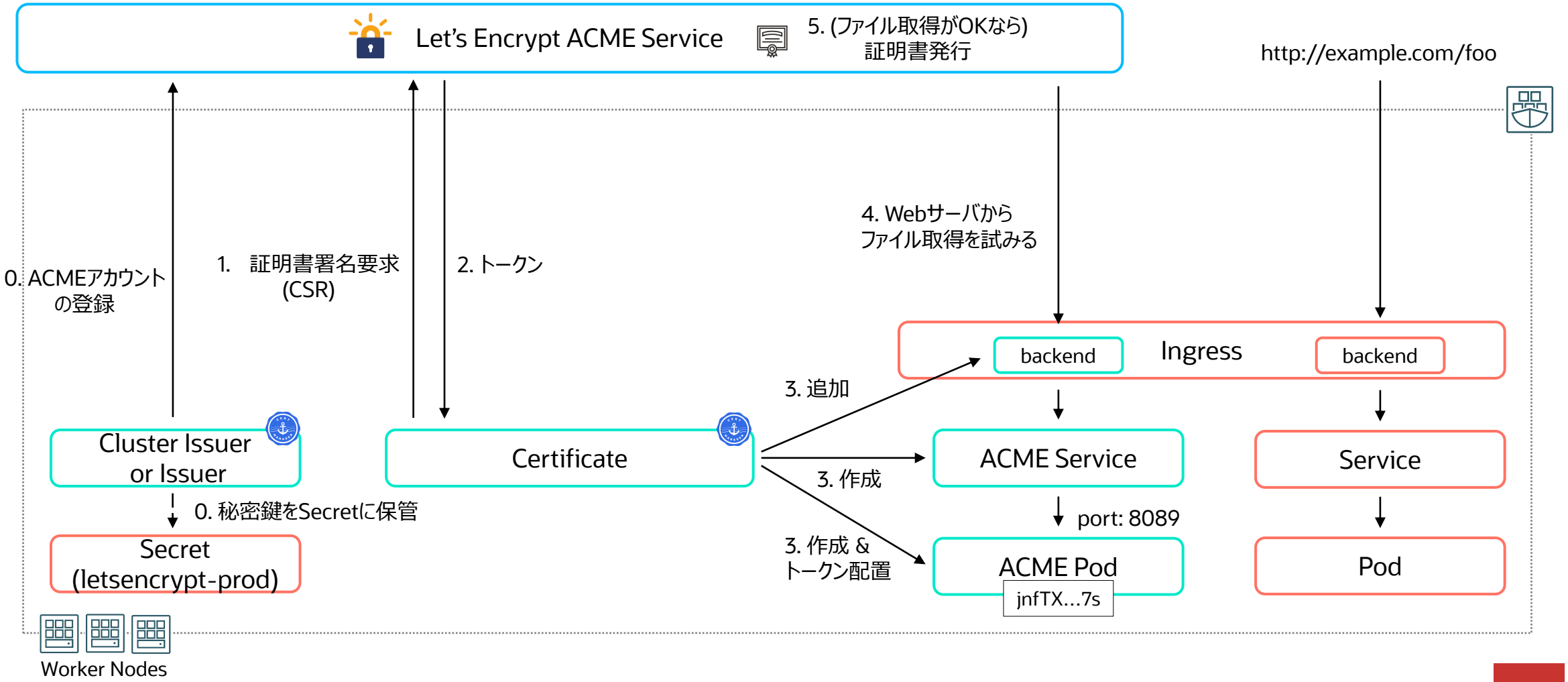
cert-manager architecture overview w/ Let's Encrypt



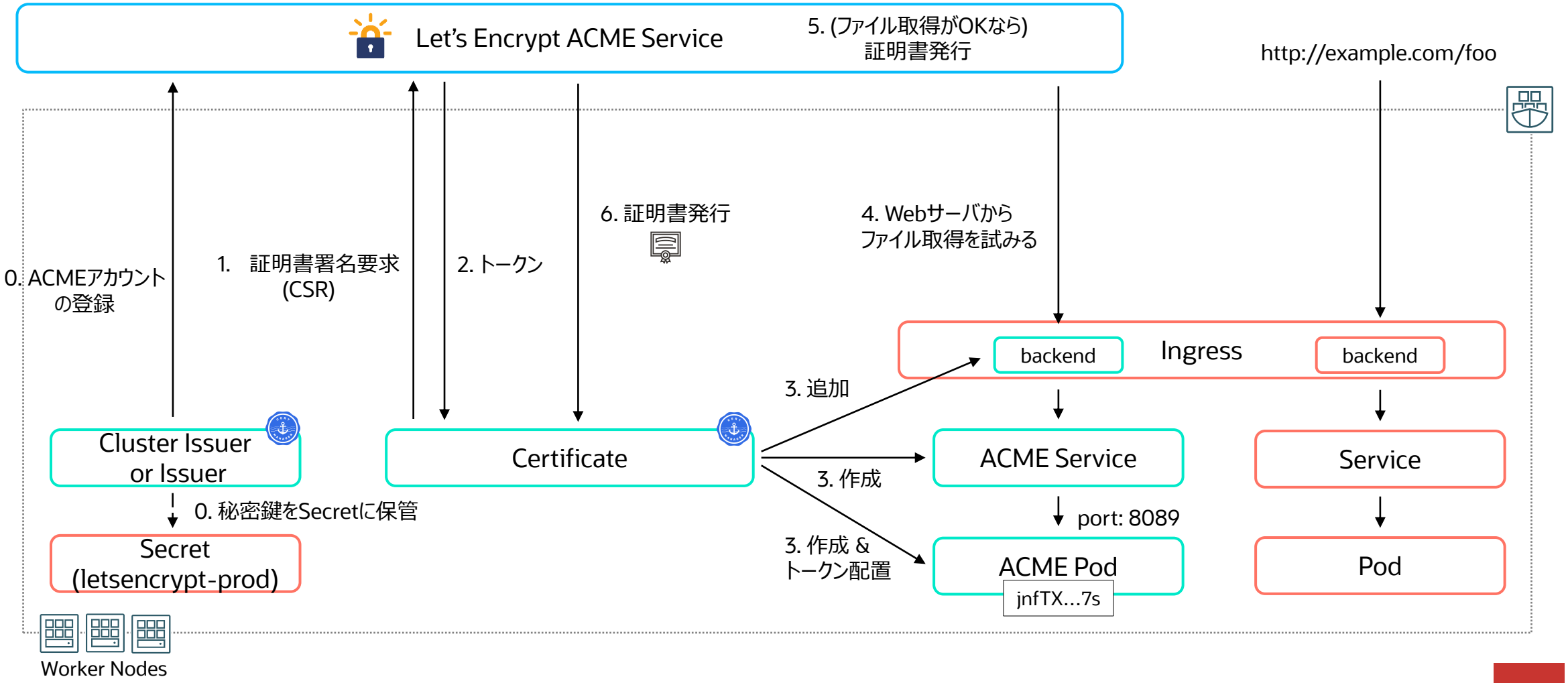
cert-manager architecture overview w/ Let's Encrypt



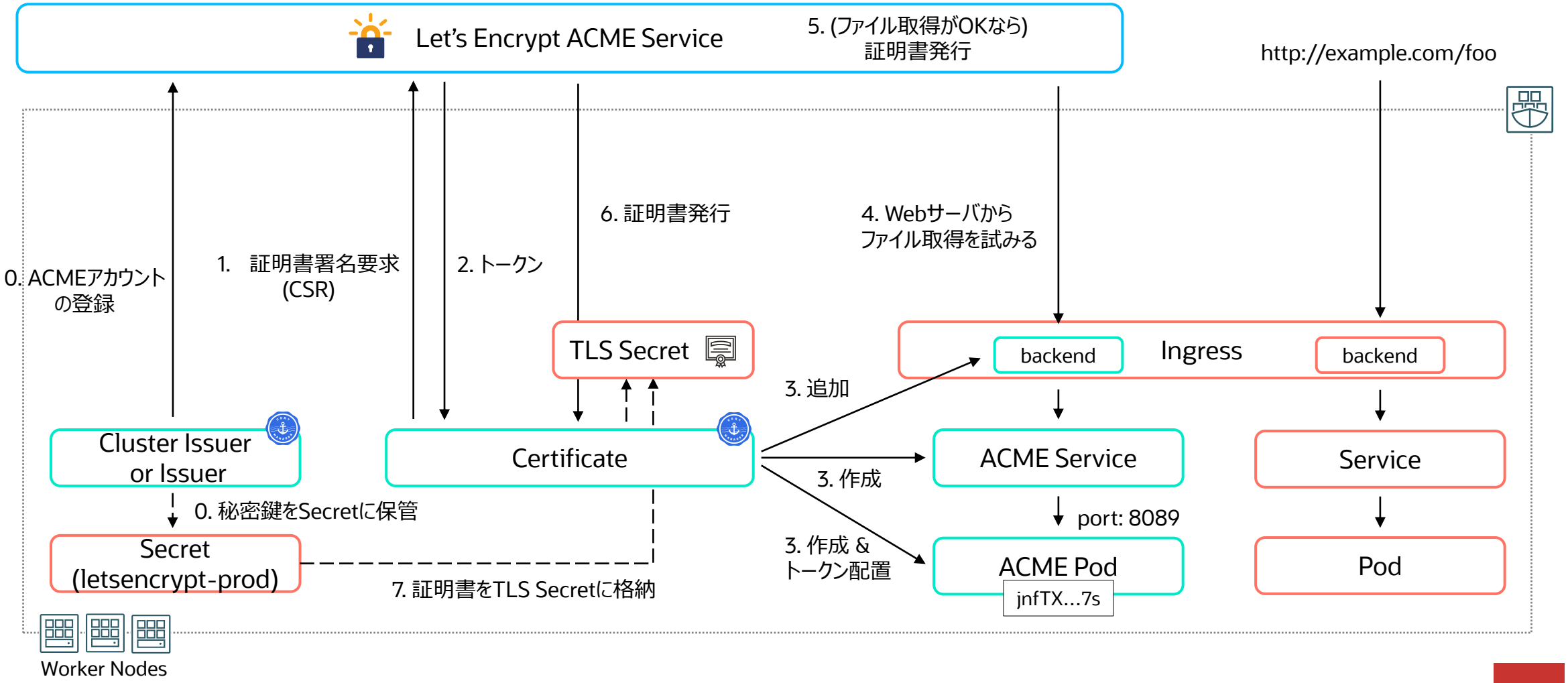
cert-manager architecture overview w/ Let's Encrypt



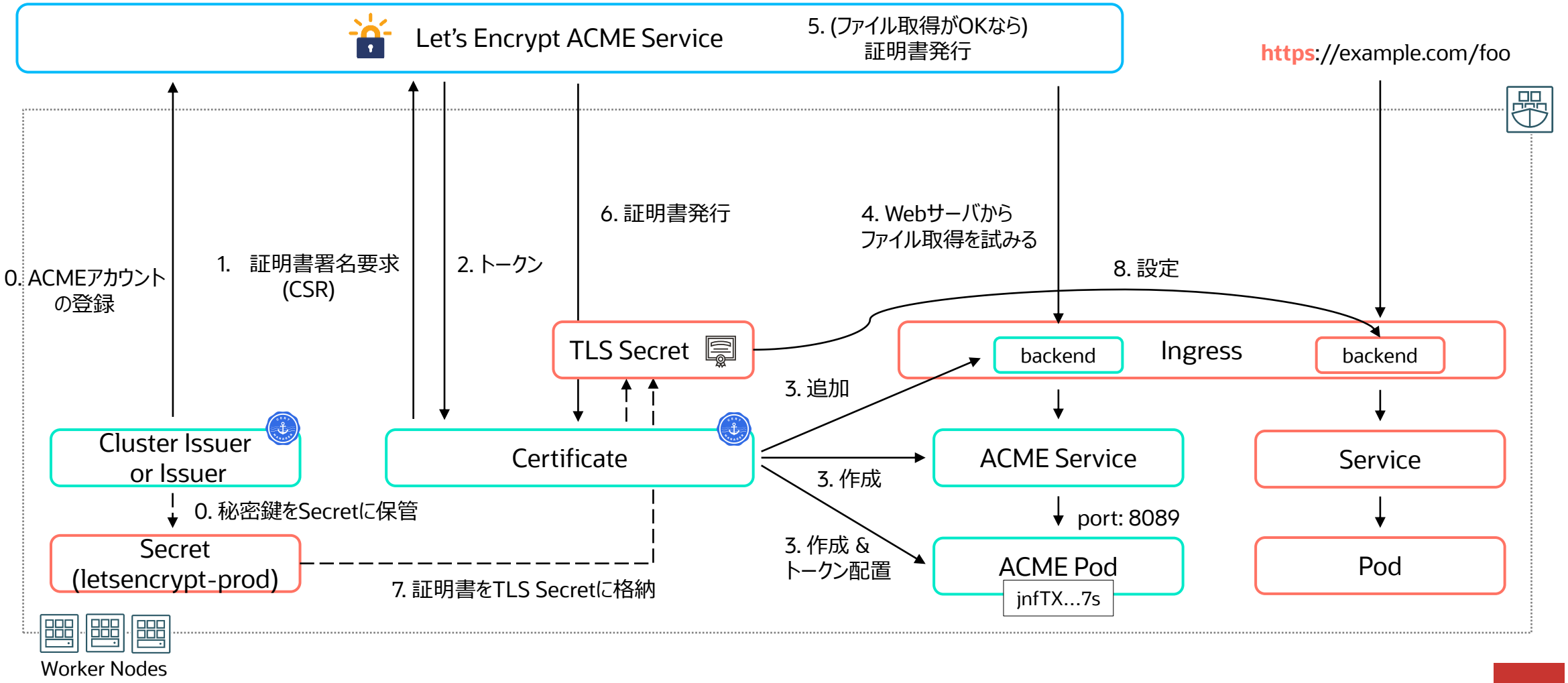
cert-manager architecture overview w/ Let's Encrypt



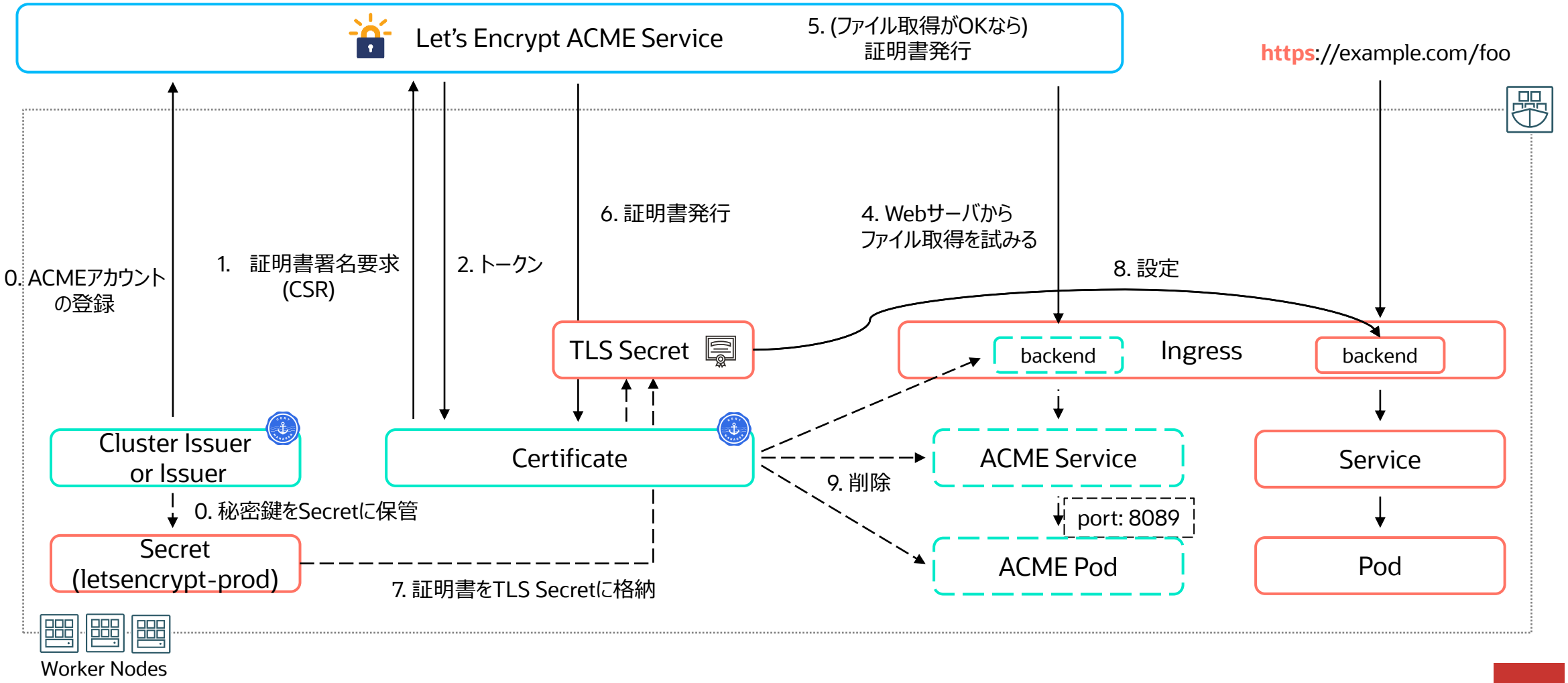
cert-manager architecture overview w/ Let's Encrypt



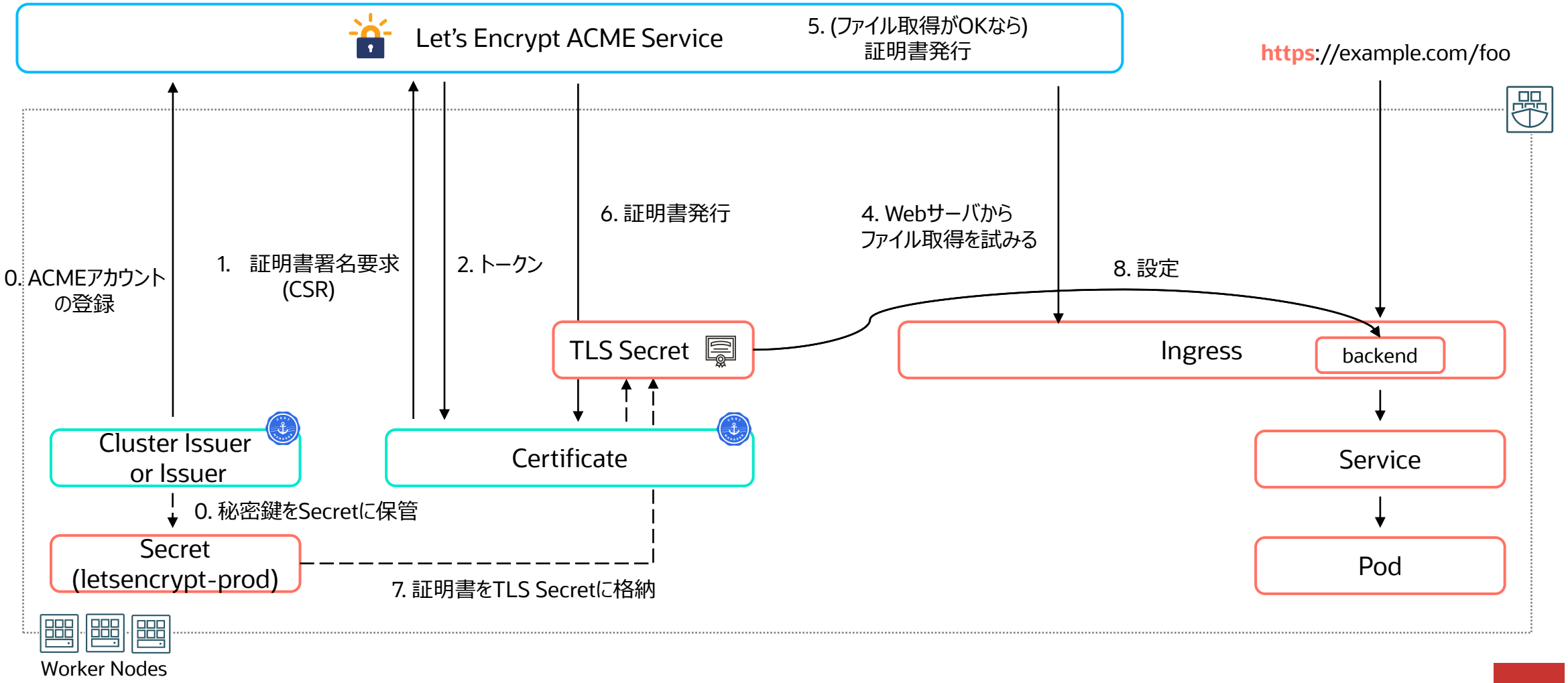
cert-manager architecture overview w/ Let's Encrypt



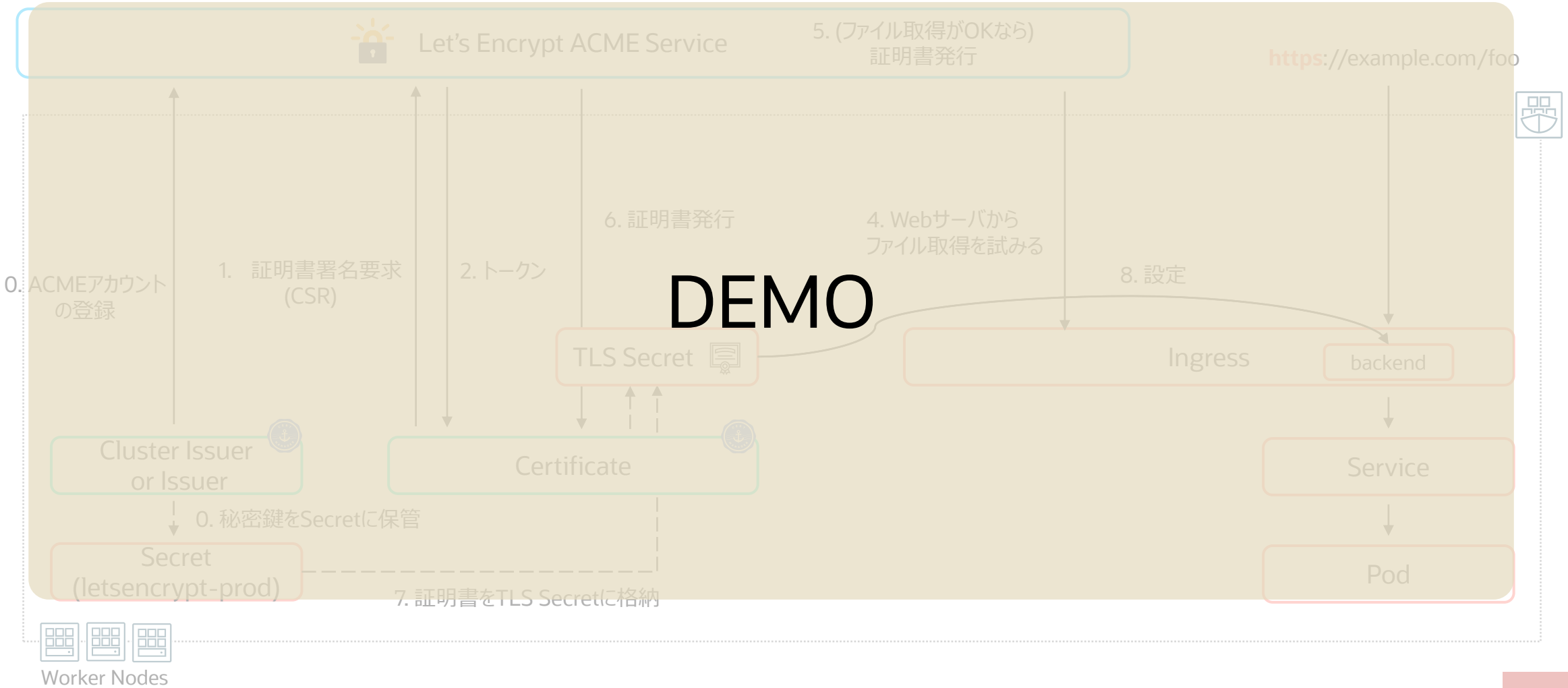
cert-manager architecture overview w/ Let's Encrypt



cert-manager architecture overview w/ Let's Encrypt



cert-manager architecture overview w/ Let's Encrypt



証明書の更新

証明書の更新プロセス → cert-managerのコア機能に含まれ、自動的に行われる！

更新処理がトリガーされる例:

- Certificate.spec.secretNameに記載されたSecret(= 証明書の実体)が存在しない
- 発行された証明書の公開鍵が、Secretに格納された秘密鍵と一致しない場合
- 証明書の更新が必要な場合
  - 有効期限切れ、更新時期が現在 or 過去の場合
- cmctlを用いて手動で更新をトリガーした場合
  - cmctl: クラスタ内のcert-managerとそのリソース管理のためのコマンドラインツール
- etc.

参考: https://cert-manager.io/docs/faq/#when-do-certs-get-re-issued

cert-managerまとめ

- KubernetesでX.509証明書を簡潔に扱うためのカスタム・コントローラー
- Let's Encrypt, HashiCorp Vault, Venafi, プライベートPKIなど様々なIssuerに対応
- 証明書自動管理のプロトコル（=ACME）とチャレンジ（HTTP | DNS -01）を最低限理解するとcert-managerが裏側で実施していることの理解がしやすい
- Let's EncryptをIssuerに使っている場合は、レート制限に注意！
  - https://letsencrypt.org/ja/docs/rate-limits/

Keycloak

認証と認可

認証（Authentication/AuthN）

- ID/パスワードに代表される情報を用いてユーザー／システムの本人性を検証すること
- 認証の3要素
  - 知識 – Something You Know(SYK)/What You Know(WYK)
  - 所有 – Something You Have(SYH)/What You Have(WYH)
  - 生体 – Something You Are(SYA)/What You Are(WYA)
- HTTP 401 Unauthorized – *The request requires user authentication.*

認可（Authorization/AuthZ）

- ユーザー／システムに対して特定の処理を行うことを許可すること
- HTTP 403 Forbidden – *The server understood the request, but id refusing to fulfill it.*

➜ OpenID Connect 1.0, OAuth 2.0という標準仕様が広く使われる

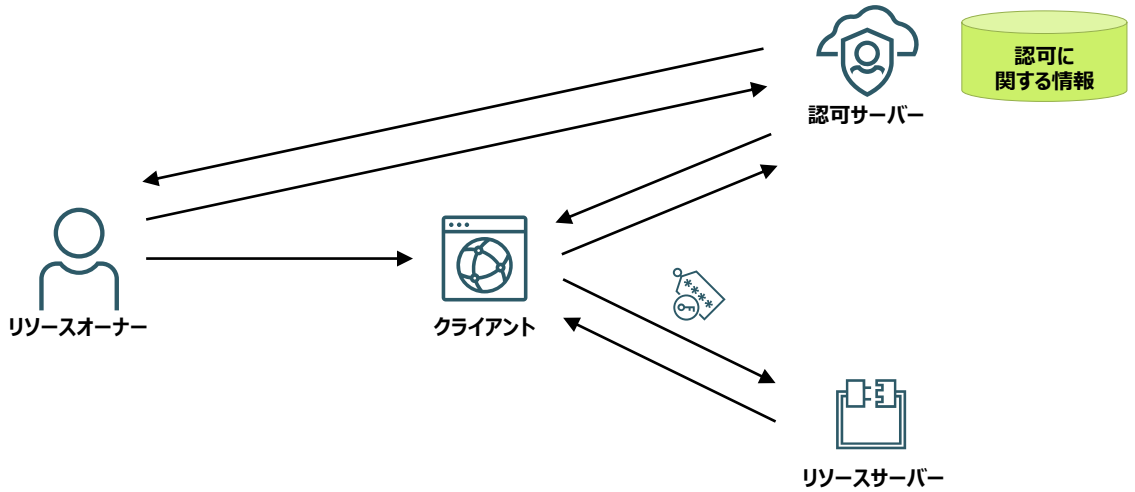
ふんわりと理解するOAuth 2.0

認可を扱うフレームワークでRFC 6749で定義

認可サーバーから発行されるアクセストークンを用いてクライアントからリソースサーバーに対するアクセス制御を行う

アクセストークンは、実装方法が決まっていないが大別すると以下の2種類

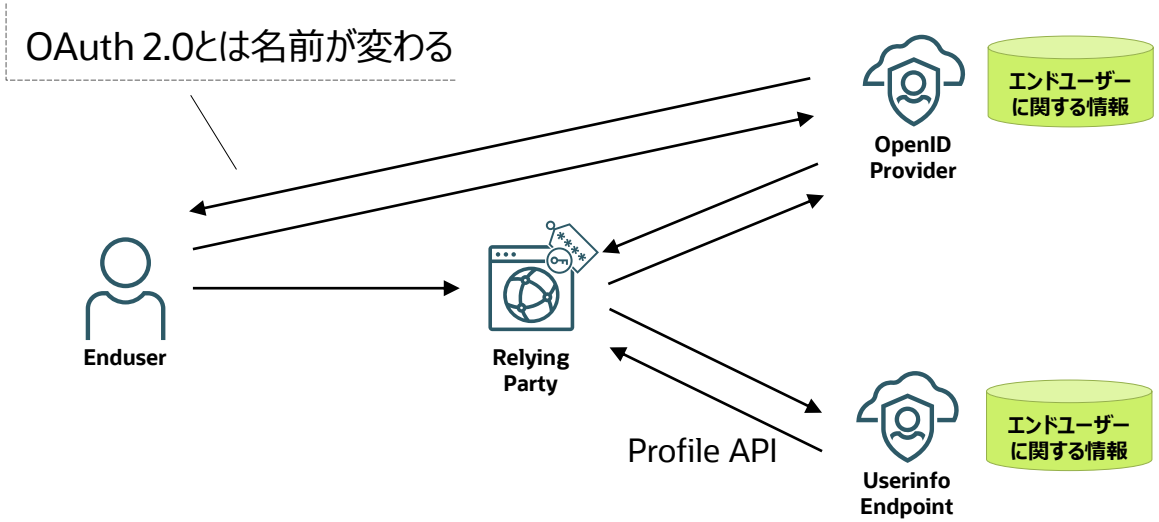
- 識別子型: ランダムな文字列を用いる。その文字列を元に認可サーバーに問い合わせる（= トークンイントロスペクション）と、認可に必要な情報が得られる
- 内包型: JWT等のエンコード方式を用いて認可に必要な情報を内包している形式



ふんわりと理解するOpenID Connect 1.0

OAuth 2.0でエンドユーザーの認証には足りない要素を自前で補い**“無理やり”実施する** → OAuth認証
…は、色々よろしくないということでOAuth 2.0ベースに安全に**認証(ID)連携する仕様**を定義 → OpenID Connect 1.0

OpenID Connect 1.0 = OAuth 2.0 + IDトークン（認証連携に必要な情報をまとめたもの） + Profile API
アクセストークンとは違い、IDトークンは実装方法（エンコード方法／クレームの内容）が標準化されている



OpenID Connect 1.0の肝は、
- エンドユーザーの認証をRPの代わりにOPが実施すること
- 認証した結果を信頼するために必要な情報（IDトークン）をOP → RPに連携すること
- 信頼するために必要な情報が含まれるようにIDトークンが標準化されていること
  - 誰が？誰を？誰のために？どんな認証方式で？、etc.

参考: マイクロサービスの認証・認可とJWT

JOSE, OAuth 2.0, OpenID Connect 1.0について

OAuth 2.0, OpenID Connect 1.0をサービスに組み込む

- 認可サーバー／OpenID Providerを自分たちで実装する
- 認可サーバー／OpenID Providerを実装したプロダクト／サービスを用いる
  - Keycloak, Okta, Auth0, Identity Domains, …

→ 今回のテーマは、“**認可サーバー／OpenID Providerを実装したプロダクト／サービスを用いる**“こと

Identity Domains/
IDCS

…

okta

Auth0

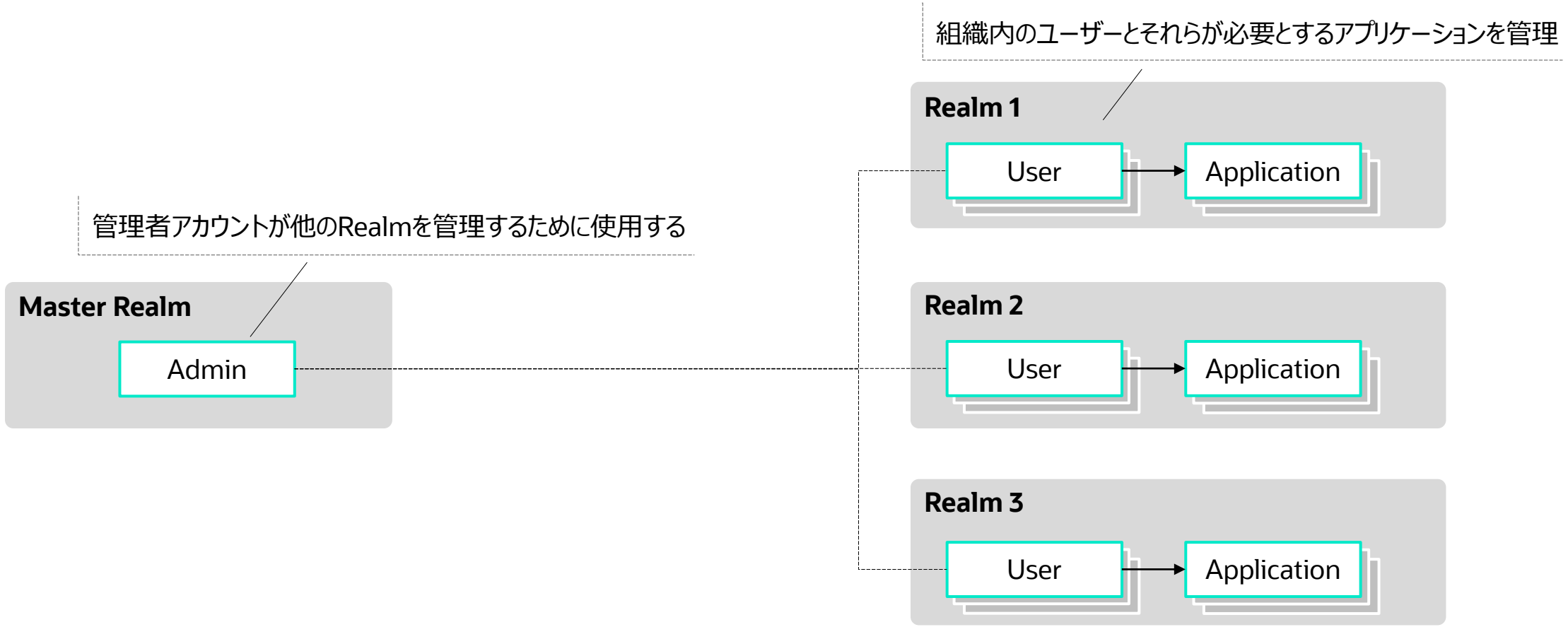
Keycloak

https://github.com/keycloak/keycloak

- Open Source Identity and Access Management For Modern Applications and Services
- CNCF incubating project
- 最新版は、Version 22.0.1 (※2023/09現在)
  - Version 17.0.0にデフォルトのディストリビューションがWildFlyからQuarkusに
- 開発者が自分で実装する必要のあるセキュリティ機能を提供
- Realmという論理単位でユーザー等を管理することでマルチテナントな運用も可能とする
- 様々なプラットフォームに対応
  - Docker, Kubernetes, OpenJDK, OpenShift, Podman

Keycloak - Realm

ユーザー、資格情報、ロール、グループを管理するKeycloakの論理単位



参考: https://www.keycloak.org/docs/latest/server_admin/index.html#the-master-realm

Keycloakが提供するセキュリティ機能 1/2

*: 本日扱う内容

- Single-Sign On/Out *
- Standard Protocol Support * – OpenID Connect, OAuth 2.0, SAML
- Identity Brokering & Social Login * – Google, GitHub, Facebook, Twitter(X), and other IdP
- User Federation – LDAP, Active Directory
- Kerberos Bridge
- Centralized Management for admins and users *
- Theme support *
- Two-factor Authentication * – Google Authenticator, FreeOTP, …

参考: https://www.keycloak.org/docs/latest/server_admin/index.html#features

Keycloakが提供するセキュリティ機能 2/2

*: 本日扱う内容

- Login flows *
- Session Management *
- Token mappers *
- CORS support
- Service Provider Interface(SPI)
- Client adapter – JavaScript applications, Wildfly, JBoss EAP, Tomcat, Jetty, Spring, etc.
- Supports any platform/language support
  - OpenID Connect Relying Party library *
  - SAML 2.0 Service Provider library

参考: https://www.keycloak.org/docs/latest/server_admin/index.html#features

Keycloakが提供するセキュリティ機能

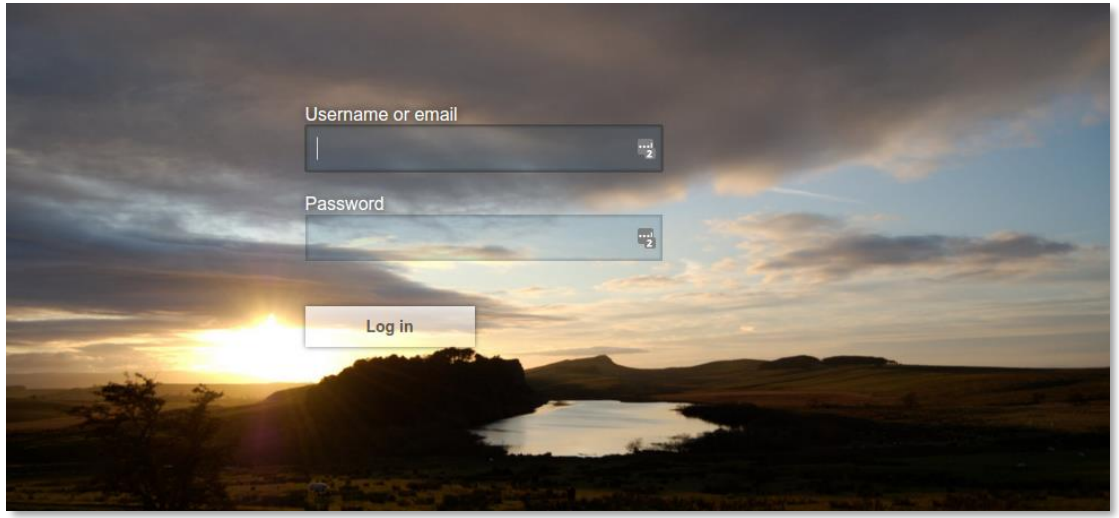
テーマのカスタマイズ

ユーザー向けのページのカスタマイズが可能

- Account management, Admin console, Emails, Login forms, Welcome page

themes/<theme-name>/<theme-type>に必要なファイルを格納することでページのカスタマイズが可能

- テーマに関する設定ファイル（theme.properties）
- 静的コンテンツ（HTML, CSS, JavaScript, Image, Message Bundle, …）



引用: https://www.keycloak.org/docs/22.0.1/server_development/#_themes

K8s上にKeycloakを構築している場合は、

- 必要なファイル（設定ファイル、静的コンテンツ）をtarやzip等にアーカイブする
- アーカイブしたコンテンツをConfigMapに設定する
- ConfigMapを/opt/keycloak/themes/<name>マウントする
- initContainersでコンテンツを/opt/keycloak/themes/<name>に配置する

等の対応が考えられる

Keycloakが提供するセキュリティ機能

Two-Factor Authentication(2FA)

2FA = リソースやデータにアクセスするために、2 種類の本人証明を要求すること

- e.g. 記憶（ID/Password）+ 所有（Authenticator）

Keycloakでは、以下に対応

- TOTP(Time-based)/H(Hash-based)OTP: Google Authenticator, FreeOTP
- FIDO/Passkey対応のAuthenticator: Security key by Yubico, Windows Hello, iPhone, …

後述するLogin flowsのカスタマイズにて、認証フローに組み込むことが可能

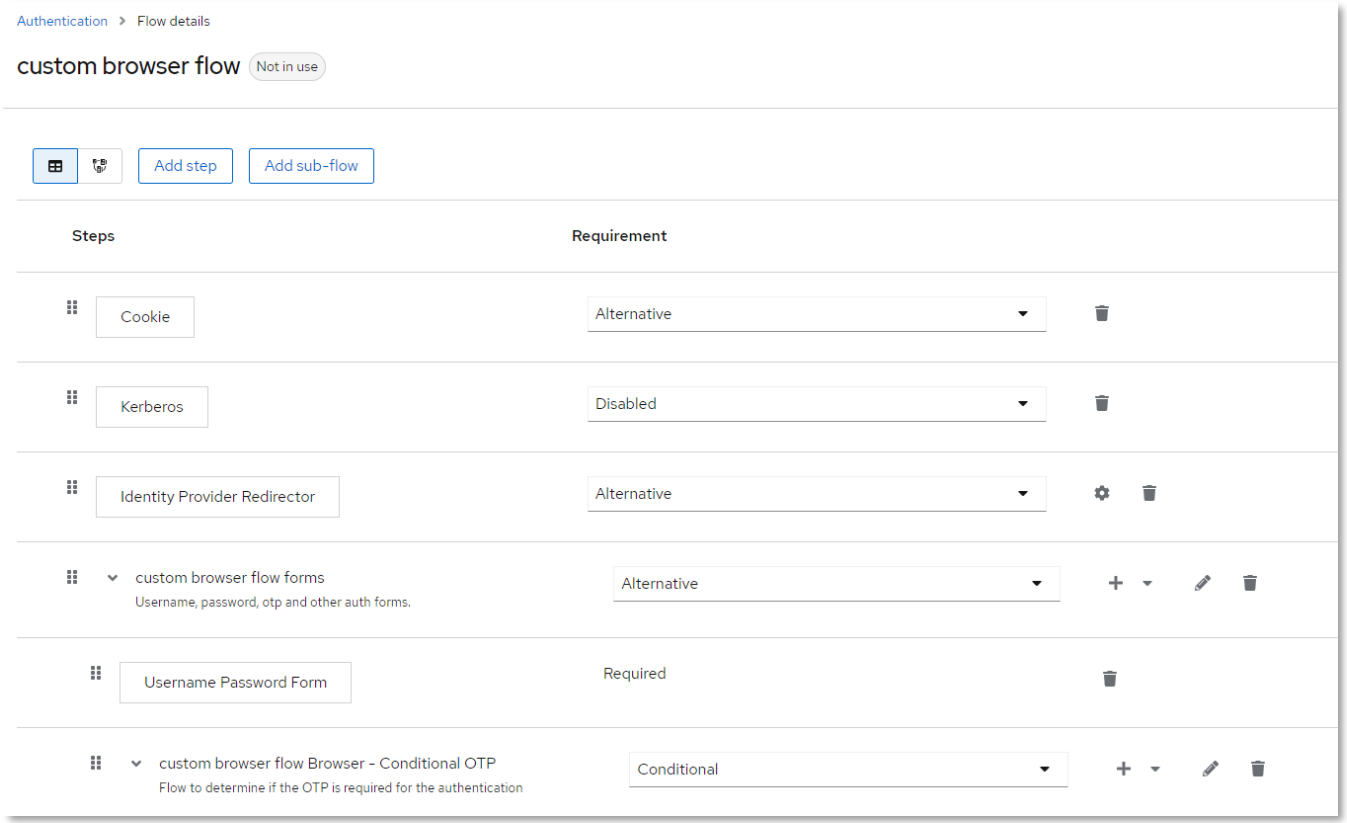
Keycloakが提供するセキュリティ機能

Login flowsのカスタマイズ

Built-inされているフローに加えて、カスタムのフローを定義することができる

例

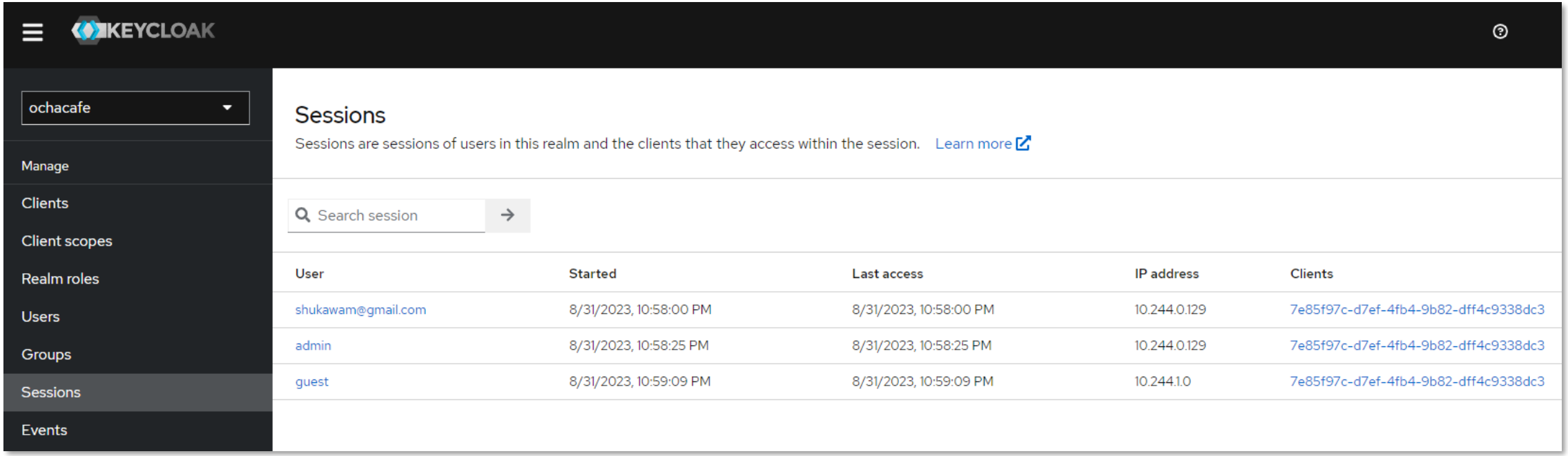
- 任意のユーザーの自己登録
- パスワードの回復
- 電子メールの確認
- パスワードの更新要求
- 2FA／パスワードレス認証を実現する認証フロー



Keycloakが提供するセキュリティ機能

セッション管理

管理者、ユーザー自身でセッションの参照、管理が可能

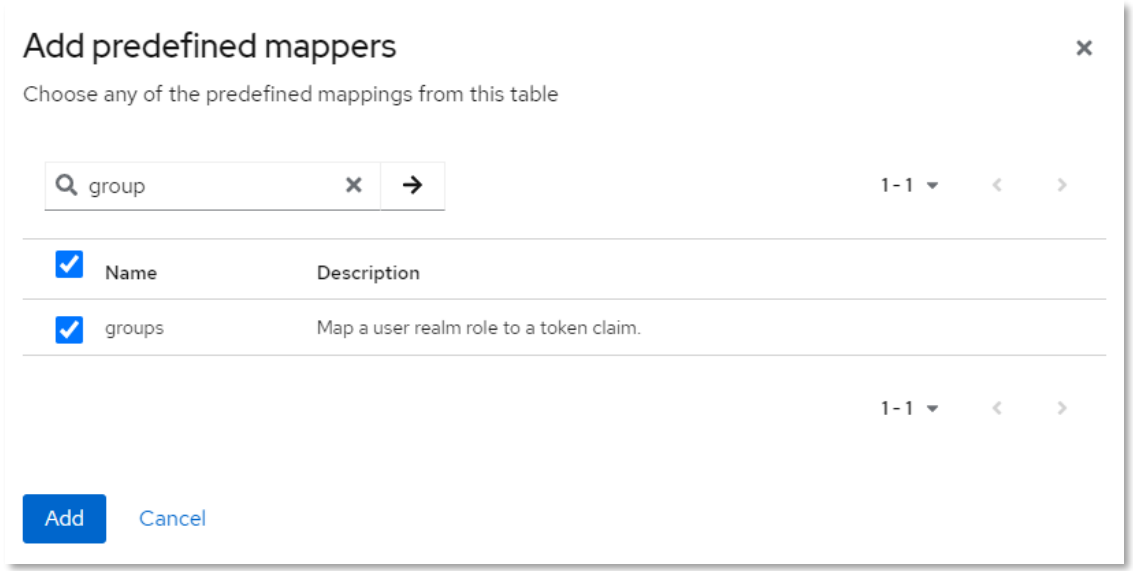


Keycloakが提供するセキュリティ機能

Token mappers

ユーザーの属性やロール、グループなどをKeycloakが払い出すトークンにマッピングできる機能

例：”groups”というclaimが必要だが、Keycloak標準ではトークンに含まれないので、Keycloak内のグループを”groups” claimにマッピングする



Keycloakの設定方法アレコレ

- Keycloakの管理コンソールから設定をする
  - 直感的な操作で設定が行える
  - UIから設定したRealmの設定をエクスポートも可能
- Realmを設定するための構成情報を書き、それを元に設定する
  - 構成情報がコードとして管理できる（= Infrastructure as Code）
  - 設定を投入する手段は様々
    - Keycloak Admin ConsoleにRealm定義（JSON）を取り込む
    - Keycloak Admin REST API – Realms Admin
    - Keycloak Operator
    - mrparkers/terraform-provider-keycloak
    - pulumi/pulumi-keycloak
    - ...

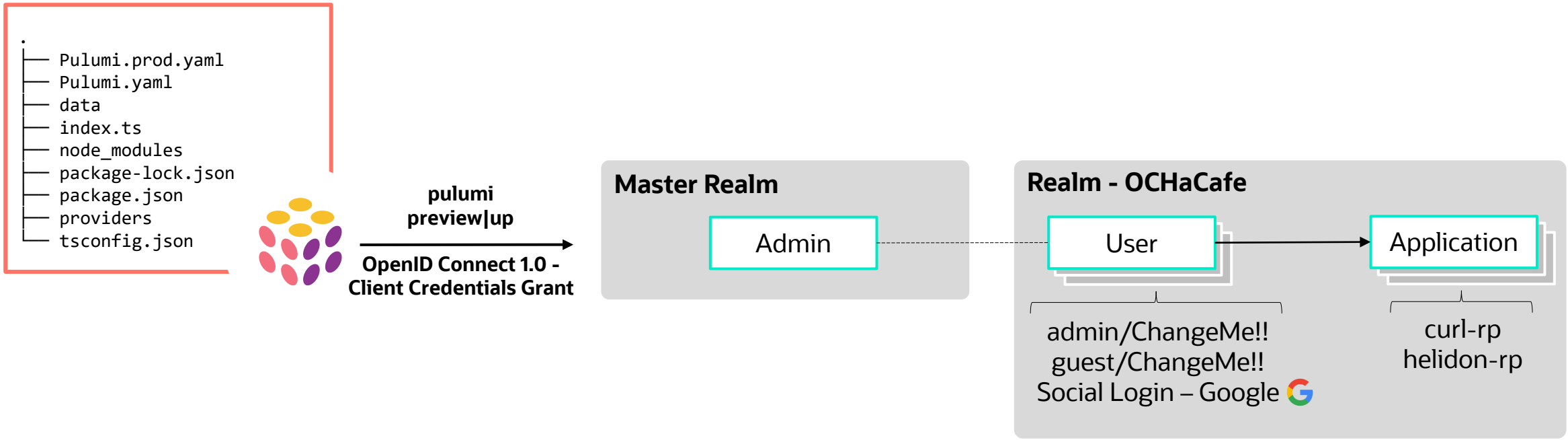
e.g. Pulumiを用いた宣言的なKeycloakのRealm定義

1. Keycloak管理コンソールからPulumi用のクライアントを作成する
    1. 管理者権限を渡しても良いが、OIDC(Client Credentials Grant)ベースで最小限の権限を持ったクライアントを作成する方が安全にIaCツールを扱える
2. 環境変数 or PulumiのConfigにURL, ClientID, ClientSecretを渡す
3. packages/keycloak を参照しながらプロビジョニングしたいリソースを宣言的に定義する
4. `pulumi preview`で作成されるリソースを確認する
5. `pulumi up`でリソースを作成する

Demo

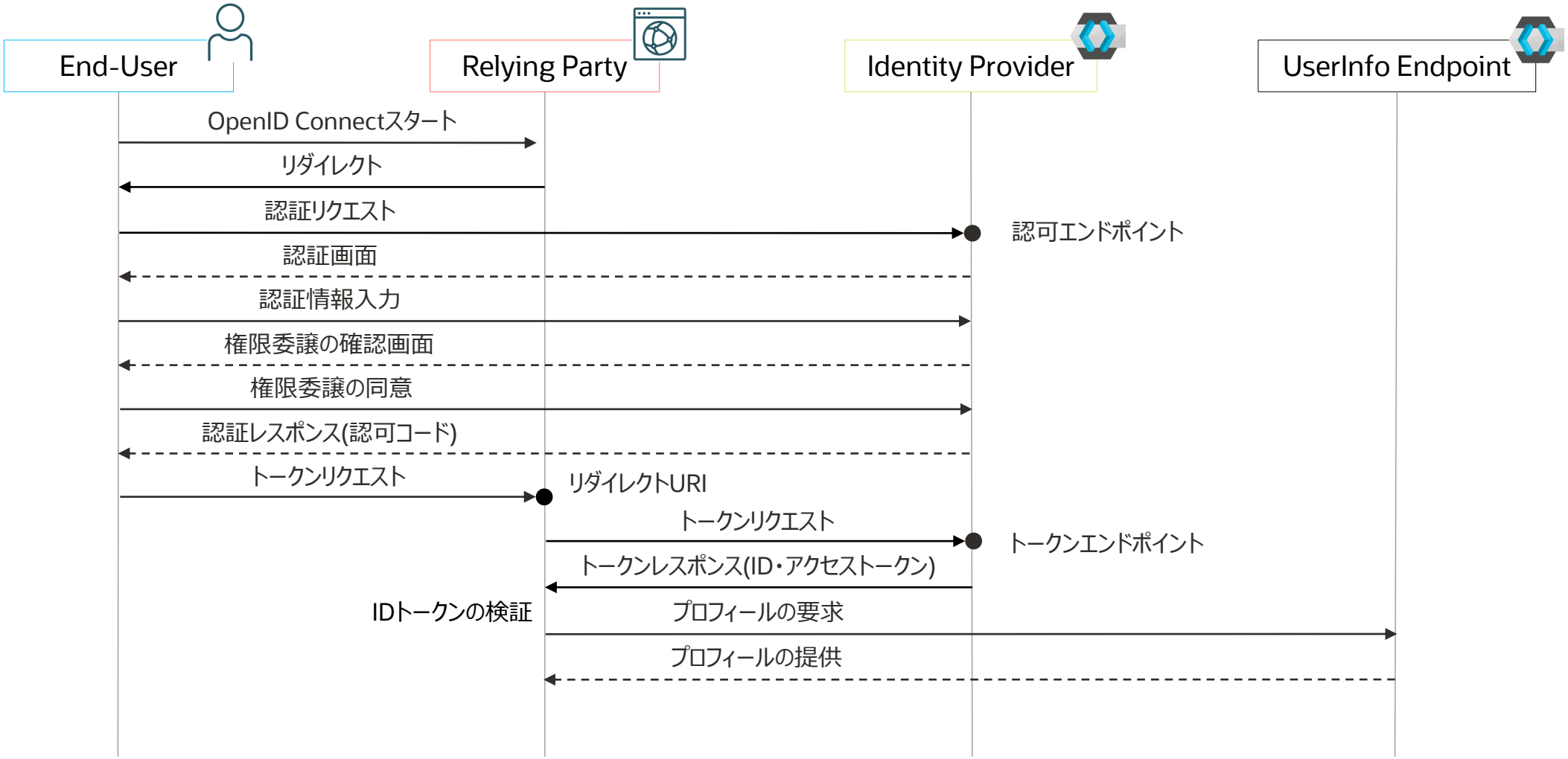
IaCツール（Pulumi）を用いた宣言的なKeycloakのRealm定義

以下の設定を持つKeycloak – Realm(OCHaCafe)を宣言的に定義、構築する



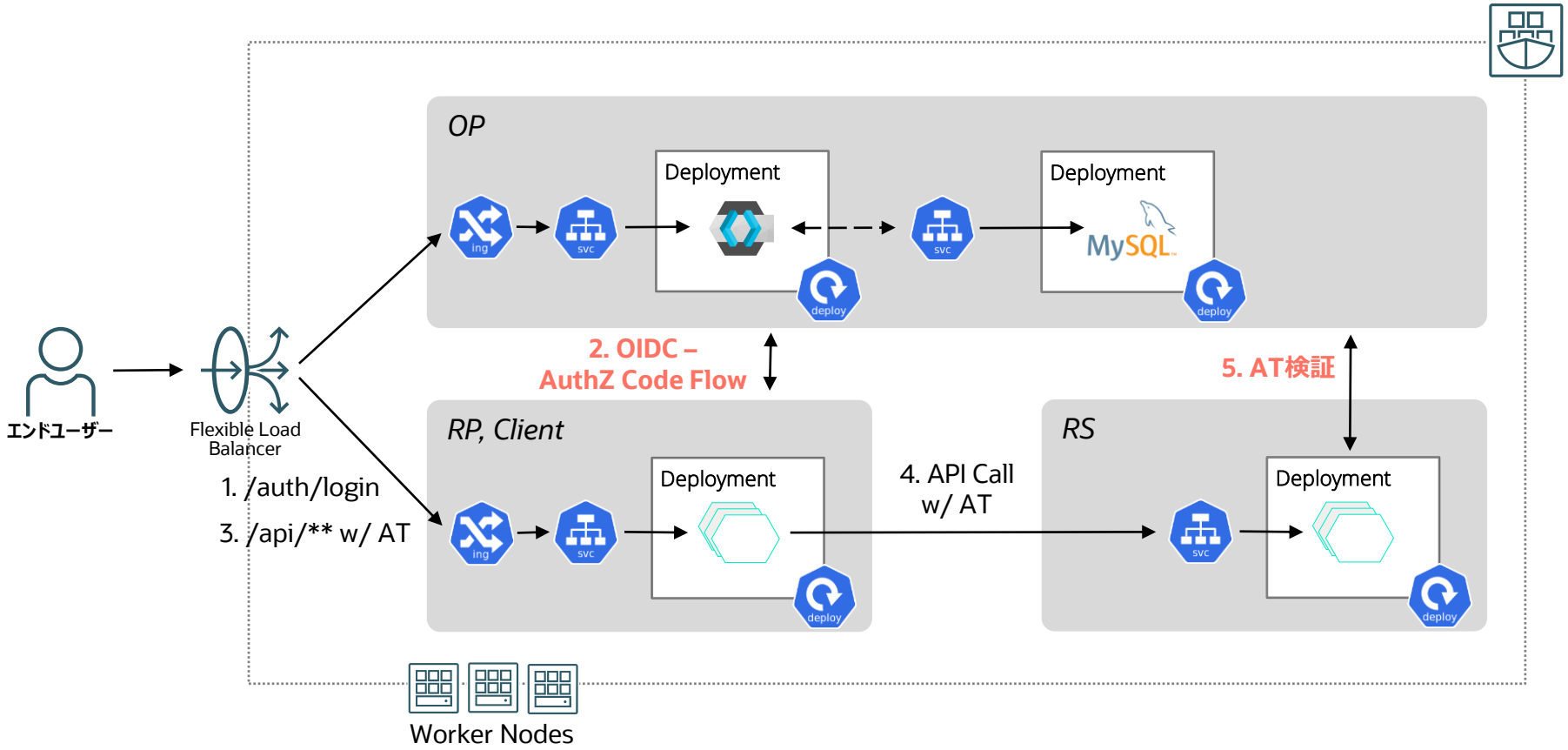
Demo

curl + KeycloakでOpenID Connect - 認可コードフローを理解する

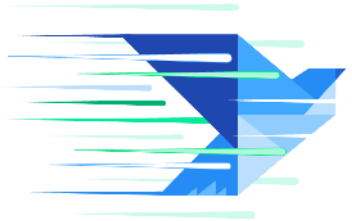


e.g. エンドユーザーの認証連携 ～ APIリソースの保護

OAuth 2.0 + OpenID Connect 1.0



Helidon



クラウドネイティブなマイクロサービスを開発するためのJavaアプリケーションフレームワーク

OracleがホストするOSSプロジェクト

- GitHubでソースコードを公開: https://github.com/oracle/helidon

マイクロサービスアプリケーションが必要とする機能を提供するJavaライブラリの集合体

- Executable JARとして動作し、コンテナ化との相性 ○
- 必要なコンポーネントを追加して拡張することも可能

マイクロサービスの開発・運用を支援する機能を提供

- REST Client, Config, OpenAPI, GraphQL, gRPC, …
- Health, Metrics(OpenMetrics), Tracing(OpenTracing), Fault Tolerance, …

2つのプログラミングモデルを提供

- Helidon MP: 宣言的記法(Java/Jakarta EE開発者フレンドリー)
- Helidon SE: 関数型記法

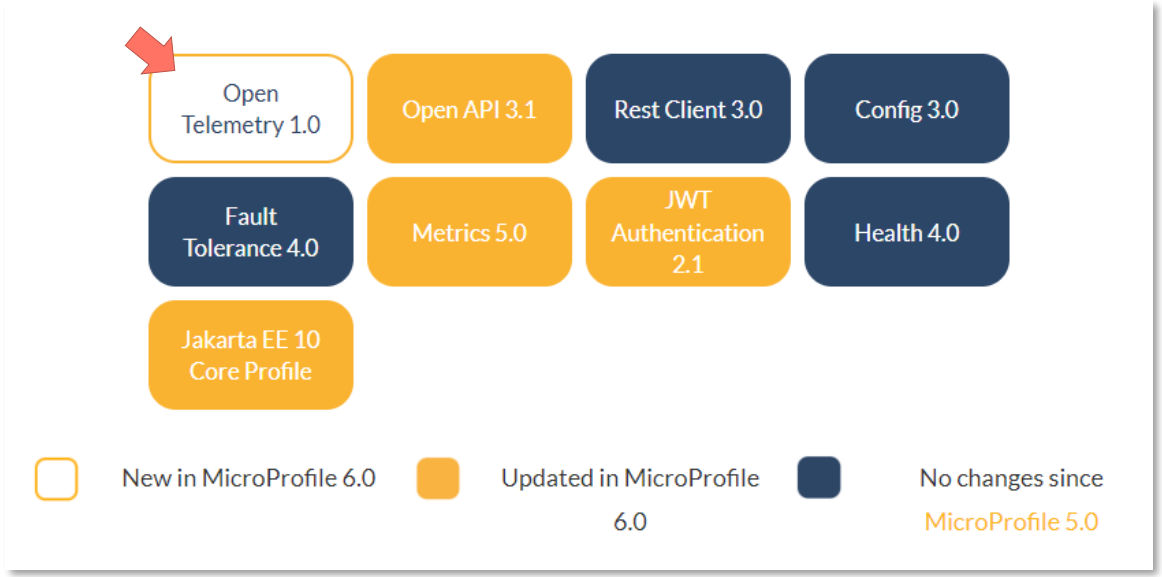
参考: Tracing(分散トレーシング)標準化について

OpenTelemetry

OpenTelemetry(OTel)

- 分散トレーシングやメトリクスなどのデータを収集、処理、エクスポートするためのオープンソースプロジェクト
- 2019年に、OpenCensus, OpenTracingがOpenTelemetryとして統合
- MicroProfile 6.0(Helidon 4系)では、OpenTelemetryに対応！ // お楽しみに！

引用：https://microprofile.io/compatible/5-0/



引用：https://microprofile.io/compatible/6-0/

Helidon – Security Provider

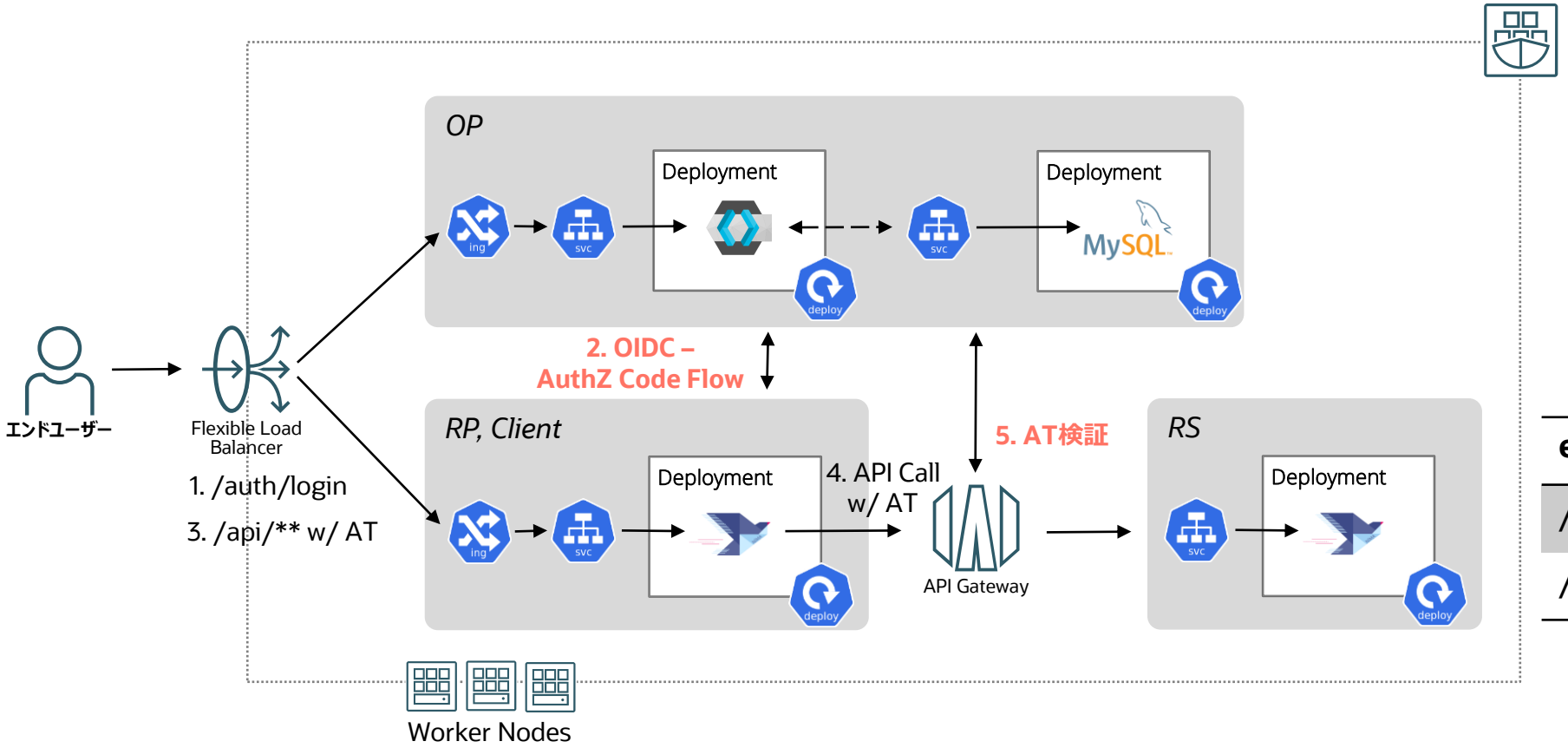
エンドポイント保護のためのプロバイダーを豊富に用意

Provider	Type	Description
OIDC Provider	Authentication	OIDC – 認可コードフローのサポート
HTTP Basic Authentication	Authentication	HTTP Basic認証のサポート
HTTP Digest Authentication	Authentication	HTTP Digest認証のサポート
Header Assertion	Authentication	ヘッダーの値に基づいたユーザーのアサーション
HTTP Signature	Authentication	署名を用いたサービス間通信の保護
IDCS Roles	Role Mapping	認証済みユーザーのロールをIDCSから取得
ABAC Authorization	Authorization	属性ベースの認可制御をサポート
Google Login	Authentication	Googleログインのサポート
JWT Provider	Authentication	JWTの検証機能をサポート

引用: https://helidon.io/docs/v3/#/mp/security/providers

Demo: エンドユーザーの認証連携 ～ APIリソースの保護

OAuth 2.0 + OpenID Connect 1.0 + API Gateway



endpoints	required scopes
/cowsay/say	cowsay:say
/cowsay/think	cowsay:think

Keycloakまとめ

- Open Source Identity and Access Management For Modern Applications and Services
- IAMにおける一般的なユースケースのサポートに加え、高いカスタマイズ性を持つ

参考情報

- OWASP API Security
  - https://owasp.org/API-Security/
- cert-manager
  - https://cert-manager.io/docs/
- Keycloak
  - https://www.keycloak.org/docs/22.0.1/server_development/
- 実践 Keycloak —OpenID Connect、OAuth 2.0を利用したモダンアプリケーションのセキュリティー保護
  - https://learning.oreilly.com/library/view/shi-jian-keycloak-openid/9784814400096/
- デモコード
  - https://github.com/oracle-japan/ochacafe-secure-api

Thank you