Ingress-Nginx Controllerの Rate limiting機能を用いてDoS攻撃を防御する

NTTコミュニケーションズ株式会社
ソリューションサービス部
増田　和己

2024/12/10

発表者 自己紹介

写真右が発表者
2020年ミャンマー勤務時に撮影

増田　和己
Kazuki Masuda

◆得意分野と現業務

Windows OS/Microsoft Azureを用いたシステム開発やトラブルシューティングを得意とし、現在はKubernetesの開発運用にも従事しています。

本資料_概要

本資料はIngress-Nginx ControllerのRate-Limiting機能を用いて、
Kubernetesクラスタ内のコンテナをDoS攻撃から保護するための手順を記載したものである。

前提

本検証にあたっての前提は以下の通り。

- KubernetesはAKS 1.29.9を用いている(AKS = Azure Kubernetes Service)。
- Ingress-Nginx Controller(OSS製品)を用いている。
- Ingress-Nginx Controllerはバージョン1.11.3である。
- AKSやIngress-Nginx Controllerの構築は完了している。
- Windows 11端末にDocker Desktopをインストールしている。

目次

1. Ingress とは

2. システム構成

3. Rate-limiting機能_利用の流れ
   a. エラーページ表示用イメージを作成
   b. エラーページ表示用イメージをAKSにデプロイ
   c. Ingress Controller(Deployment)の設定
   d. Ingress Controller(ConfigMap)の設定
   e. Ingress Ruleの設定

4. デモ

目次

1. Ingressとは

- Kubernetes内のアプリケーション提供用podに対して、外部からの(HTTP/HTTPS)アクセスを管理するAPIオブジェクトである。

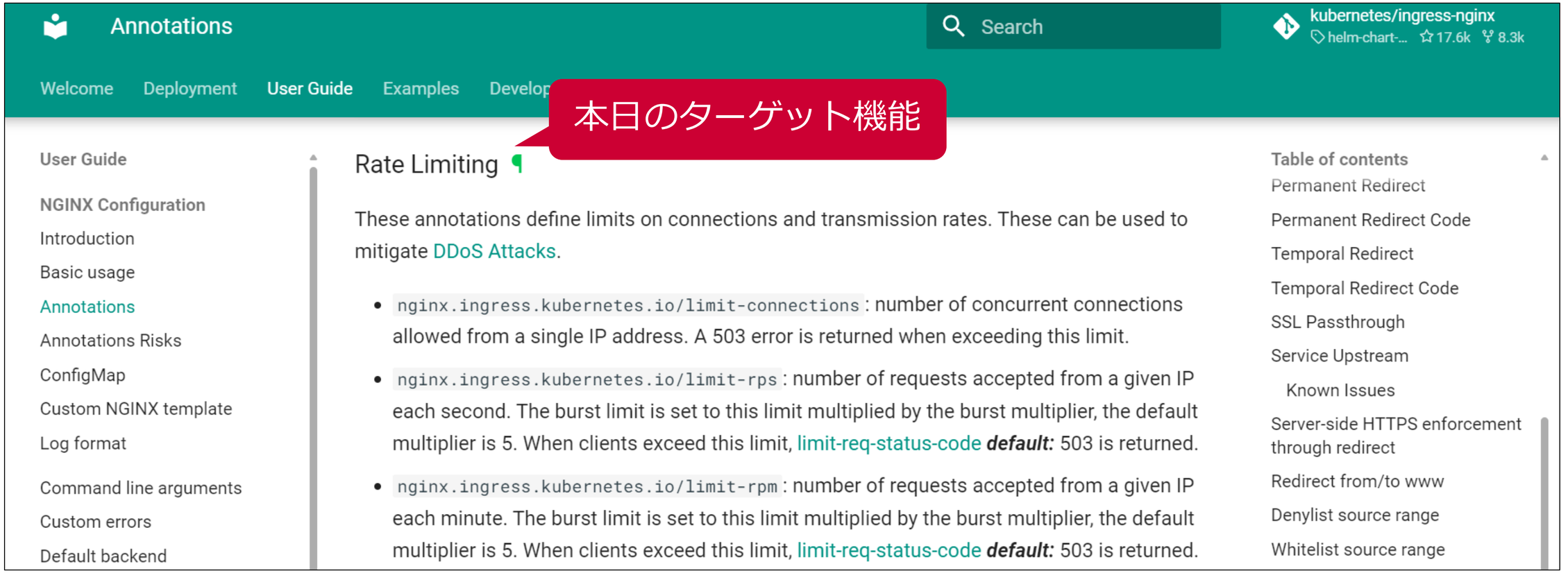
- Ingress RuleとIngress Controllerがあり、Ingress RuleはL7ロードバランサの設定を行う。
  Ingress ControllerはIngress Ruleの設定内容に基づき各種制御する。

- Kubernetes上のL7ロードバランサとして、以下を含めた機能を提供する。
  a. 外部からHTTPS通信を受付する際のSSL終端機能
  b. Ingressとアプリケーション提供用pod間のセッションアフィニティ機能
  c. URLパスベースのルーティング機能
     (例：/authの場合は認証用podに、/appの場合なアプリ用podに通信を振り分ける)
  d. 外部からのアクセス量を制御する機能(Rate Limiting機能)

本日のターゲット機能

1. Ingressとは(Rate Limiting機能)

- 特定IPアドレスからのHTTPリクエスト通信を時間単位で制御する
- 制御された通信はデフォルトでHTTP 503のエラーを返す(※)
  ※本検証ではHTTP 502エラーを返すように変更



※同機能詳細は以下Web参照
Annotations - Ingress-Nginx Controller

目次

1. Ingress とは

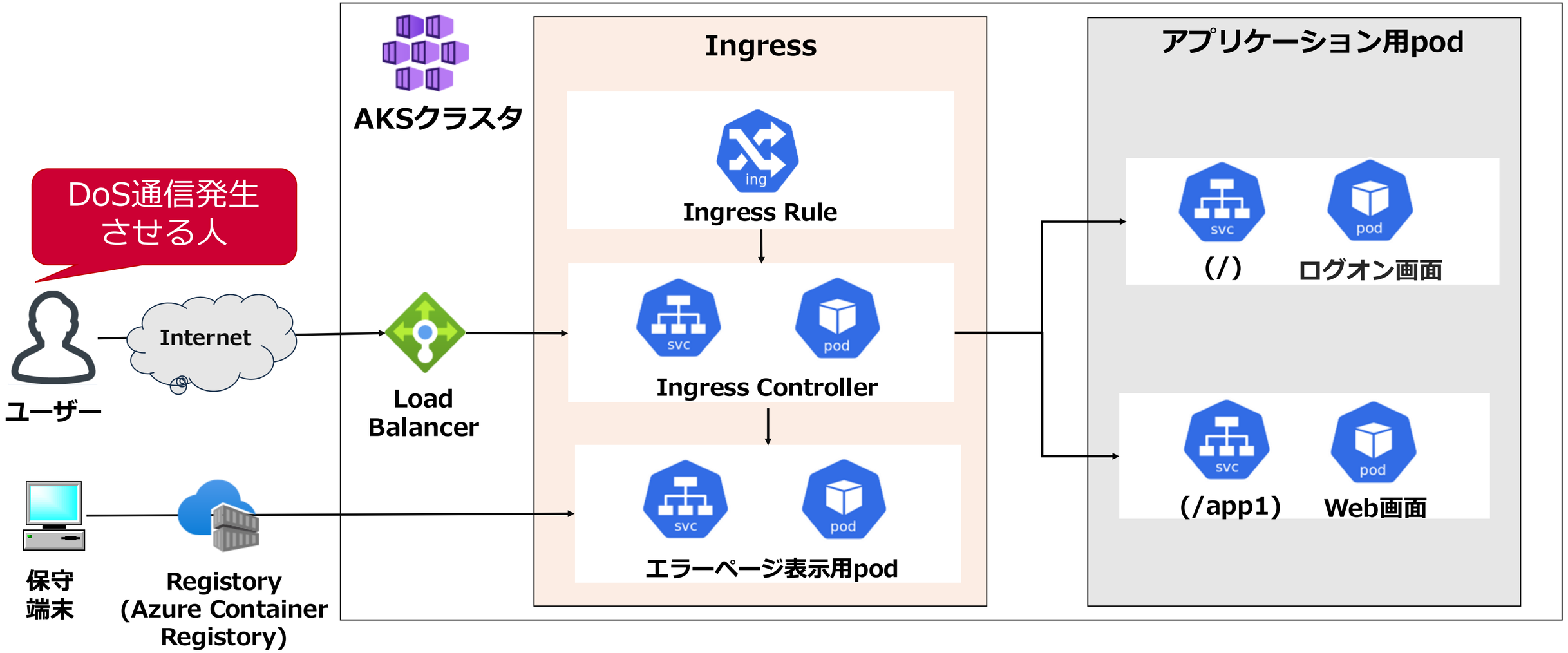
2. システム構成

3. Rate-limiting機能_利用の流れ
   a. エラーページ表示用イメージを作成
   b. エラーページ表示用イメージをAKSにデプロイ
   c. Ingress Controller(Deployment)の設定
   d. Ingress Controller(ConfigMap)の設定
   e. Ingress Ruleの設定

4. デモ

2. システム構成

本検証にあたってのシステム構成は以下のとおり。



目次

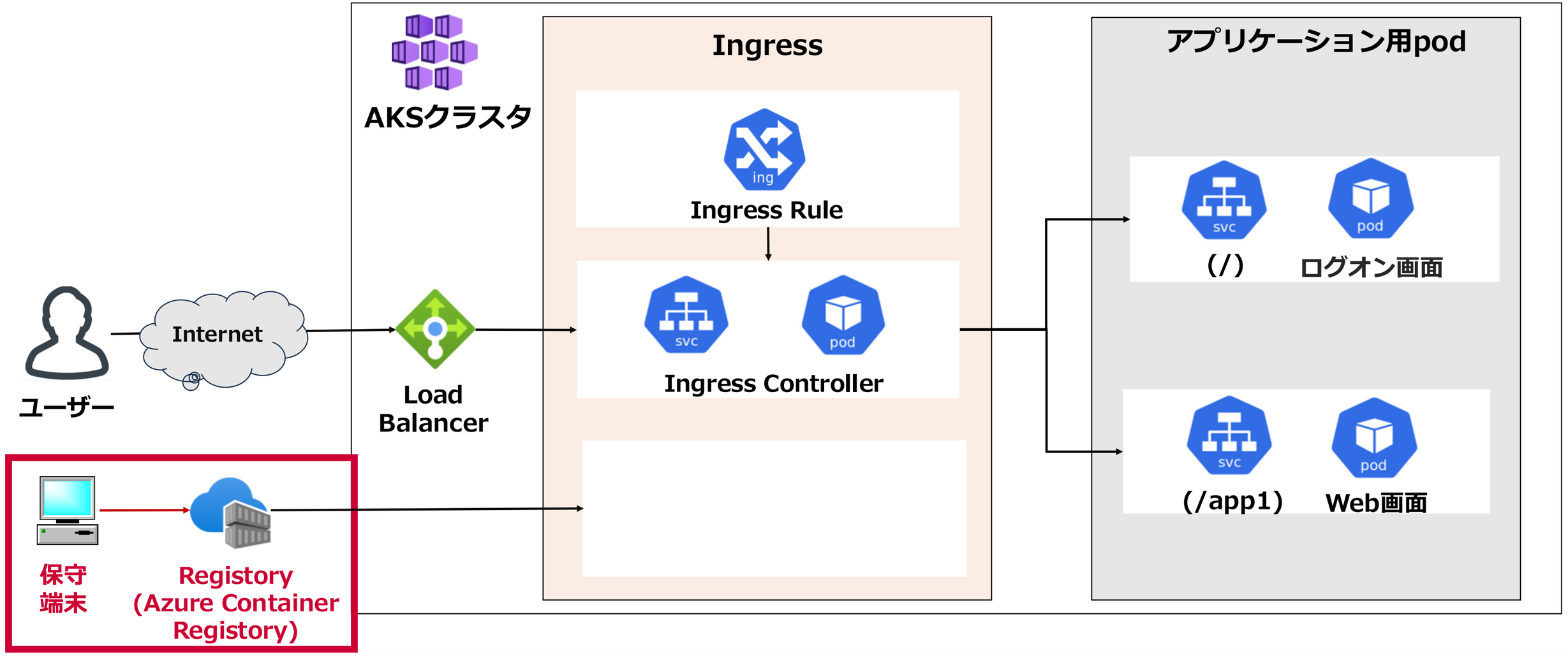
1. Ingress とは

2. システム構成

3. Rate-limiting機能_利用の流れ
   a. エラーページ表示用イメージを作成
   b. エラーページ表示用イメージをAKSにデプロイ
   c. Ingress Controller(Deployment)の設定
   d. Ingress Controller(ConfigMap)の設定
   e. Ingress Ruleの設定

4. デモ

3-a. エラーページ表示用イメージを作成



3-a. エラーページ表示用イメージを作成

保守端末にて、以下2つの資材を同一ディレクトリ階層に保存する。
- 以下内容が記載されたDockerfile
- エラーページ用HTMLファイル(502.html)が格納されたwwwディレクトリ

Dockerfile

```
FROM registry.k8s.io/ingress-nginx/custom-error-pages:v1.0.2@sha256:b2259cf6bfda813548a64bded551b1854cb600c4f095738b49b4c5cdf8ab9d21

COPY ./www /www
```

※イメージ格納元のレジストリ名は以下URLのyaml39行目から抽出
ingress-nginx/docs/examples/customization/custom-errors/custom-default-backend.yaml at main · kubernetes/ingress-nginx · GitHub

3-a(補足). 502.html

3-a. エラーページ表示用イメージを作成

Dockerコマンドでイメージの作成、ならびにACR(Azure Container Registory)へのイメージのプッシュを行う。

cmd

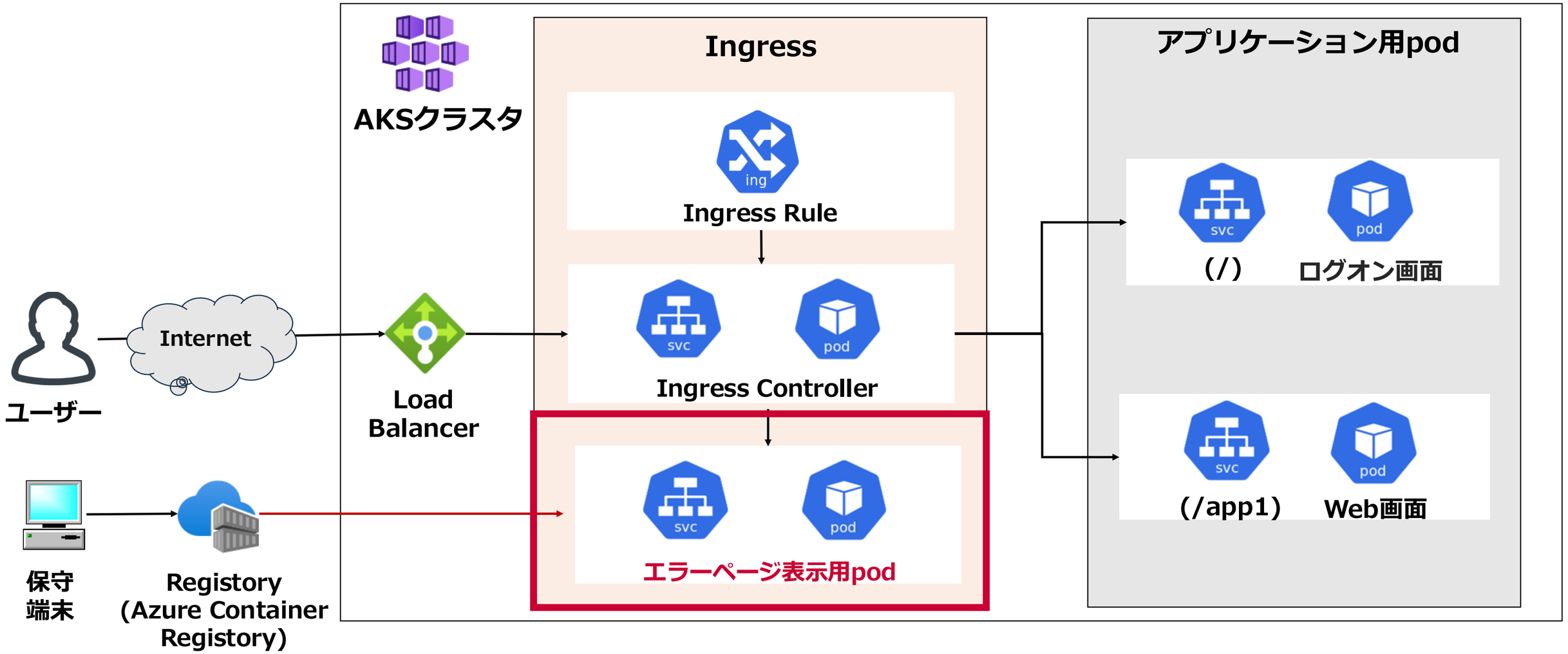
```
# イメージをビルド(※1)
Docker build –t errorimage .

# イメージにタグを割り当て(※2)
Docker tag errorimage xxx.azurecr.io/Kubernetes-ingress-controller/nginx-error:v1.0

# イメージをACRにプッシュ
Docker push xxx.azurecr.io/kubernetes-ingress-controller/nginx-error:v1.0
```

※1:errorimageはイメージ名
※2:acr01.azurecr.ioはACRのホスト名、nginx-error:v1.0はリポジトリ名とタグ

3-b. エラーページ表示用イメージをAKSにデプロイ



3-b. エラーページ表示用イメージをAKSにデプロイ

保守端末にエラーコンテナ用のDeploymentとServiceのマニフェストファイルを用意し、
AKSにデプロイする。

Cmd

```
kubectl apply -f errordeployment.yaml

kubectl apply -f errorservice.yaml
```

※基となるマニフェストファイルは以下から取得
ingress-nginx/docs/examples/customization/custom-errors/custom-default-backend.yaml at main · kubernetes/ingress-nginx · GitHub

3-b. エラーページ表示用イメージをAKSにデプロイ(Deployment)

errordeployment.yaml (サンプル)

```
apiVersion: apps/v1
kind: Deployment

metadata:
  name: nginx-errors
  namespace: ingress-basic
  labels:
    app.kubernetes.io/name: nginx-errors
    app.kubernetes.io/part-of: ingress-nginx

spec:
  replicas: 2
  selector:
    matchLabels:
      app.kubernetes.io/name: nginx-errors
      app.kubernetes.io/part-of: ingress-nginx
  template:
    metadata:
      labels:
        app.kubernetes.io/name: nginx-errors
        app.kubernetes.io/part-of: ingress-nginx
    spec:
      containers:
      - name: nginx-error-server
        image: xxx.azurecr.io/nginx-error:v1.0
        ports:
        - containerPort: 8080
```

15頁にて設定したACRを指定

3-b. エラーページ表示用イメージをAKSにデプロイ(Service)

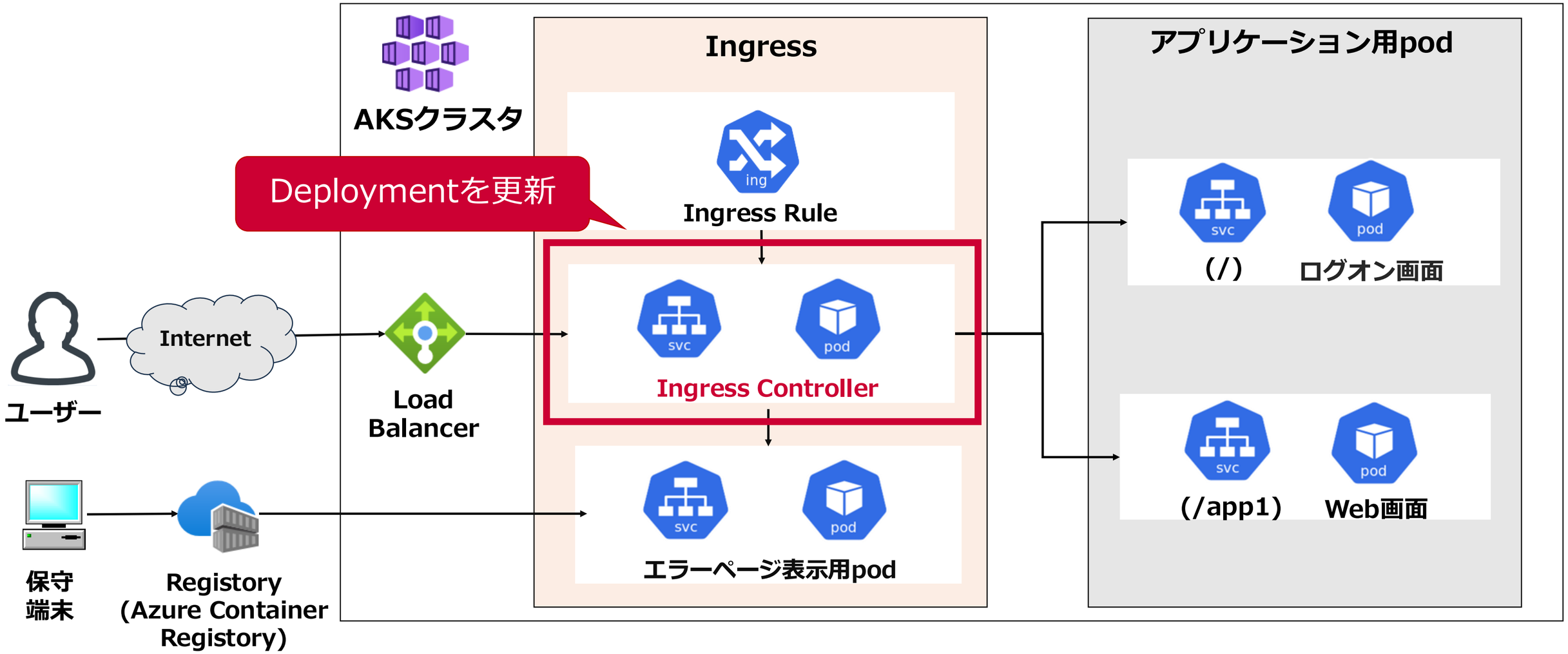
errorservice.yaml (サンプル)

```
apiVersion: v1
kind: Service

metadata:
  name: nginx-errors
  namespace: ingress-basic
  labels:
    app.kubernetes.io/name: nginx-errors
    app.kubernetes.io/part-of: ingress-nginx

spec:
  selector:
    app.kubernetes.io/name: nginx-errors
    app.kubernetes.io/part-of: ingress-nginx
  ports:
  - port: 80
    targetPort: 8080
    name: http
```

3-c. Ingress Controller(Deployment)の設定



3-c. Ingress Controller(Deployment)の設定

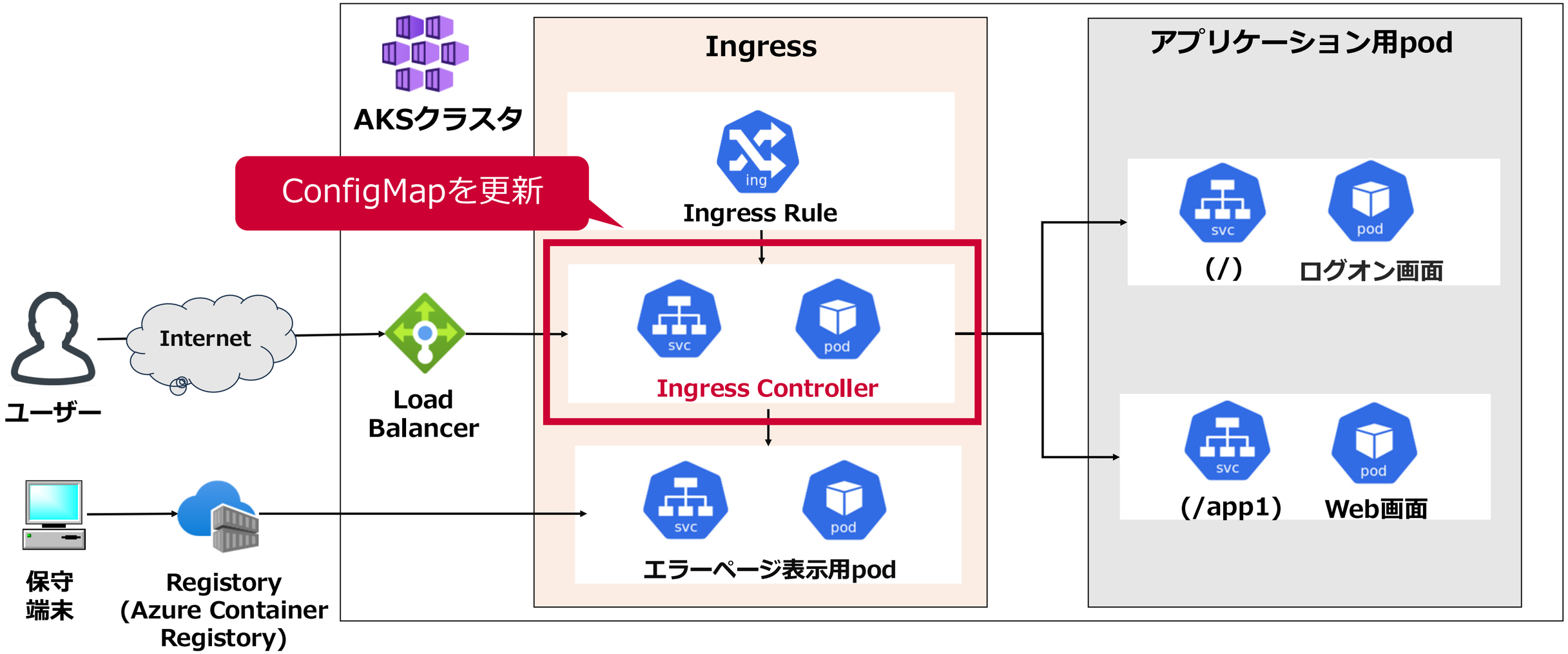
アクセス過多時の通信先を**エラーページ表示用pod**に振り向かせるために、Ingress ControllerのDeployment YAMLのSpec-template-spec-containers-argsセクションに、以下1行を追加する

ingress-nginx-controller_(Deployment.yaml)

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: ingress-nginx-controller
  namespace: ingress-basic
   ~
spec:
  selector:
    matchLabels:
      app.kubernetes.io/component: controller
       ~
  template:
      ~
    spec:
      ~
      containers:
          ~
          args:
            ~
            - '--default-backend-service=$(POD_NAMESPACE)/nginx-errors'
```

1秒間に許容するHTTPリクエスト数を超える場合、
エラーページ表示用podに通信させるための設定

3-d. Ingress Controller(ConfigMap)の設定



3-d. Ingress Controller(ConfigMap)の設定

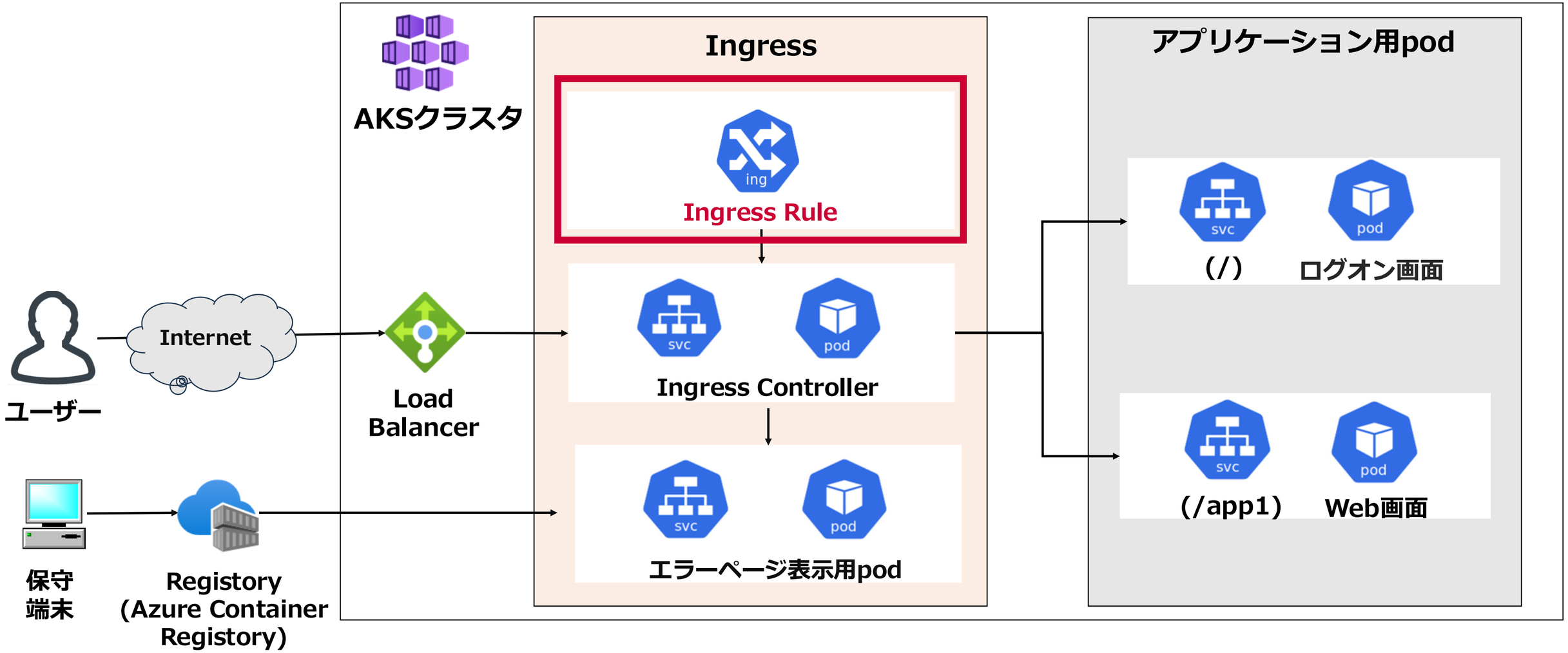
エラーページ表示用podアクセス時のHTTPレスポンスコードを指定するために、
Ingress ControllerのConfigMap YAMLのdataセクションに以下赤字2行を追加する

ingress-nginx-controller (ConfigMap.yaml)

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: ingress-nginx-controller
  namespace: ingress-basic
   ～
data:
  custom-http-errors: '502'
  limit-req-status-code: '502'
```

1秒間に許容するHTTPリクエスト数を超える場合の
HTTPレスポンスコードを指定

3-e. Ingress Ruleの設定



3-e. Ingress Ruleの設定

1秒間に許可するアクセス量を指定するために、Ingress Ruleのannotationセクションに
以下赤字2行を追加する。

ingress-rule_Ingress.yaml

```
kind: Ingress
apiVersion: networking.k8s.io/v1
metadata:
  name: ingress-rule
  ~
  annotations:
      ~
    nginx.ingress.kubernetes.io/limit-rps: '1'
```

1秒間に許容するHTTPリクエスト数を指定

目次

1. Ingress とは

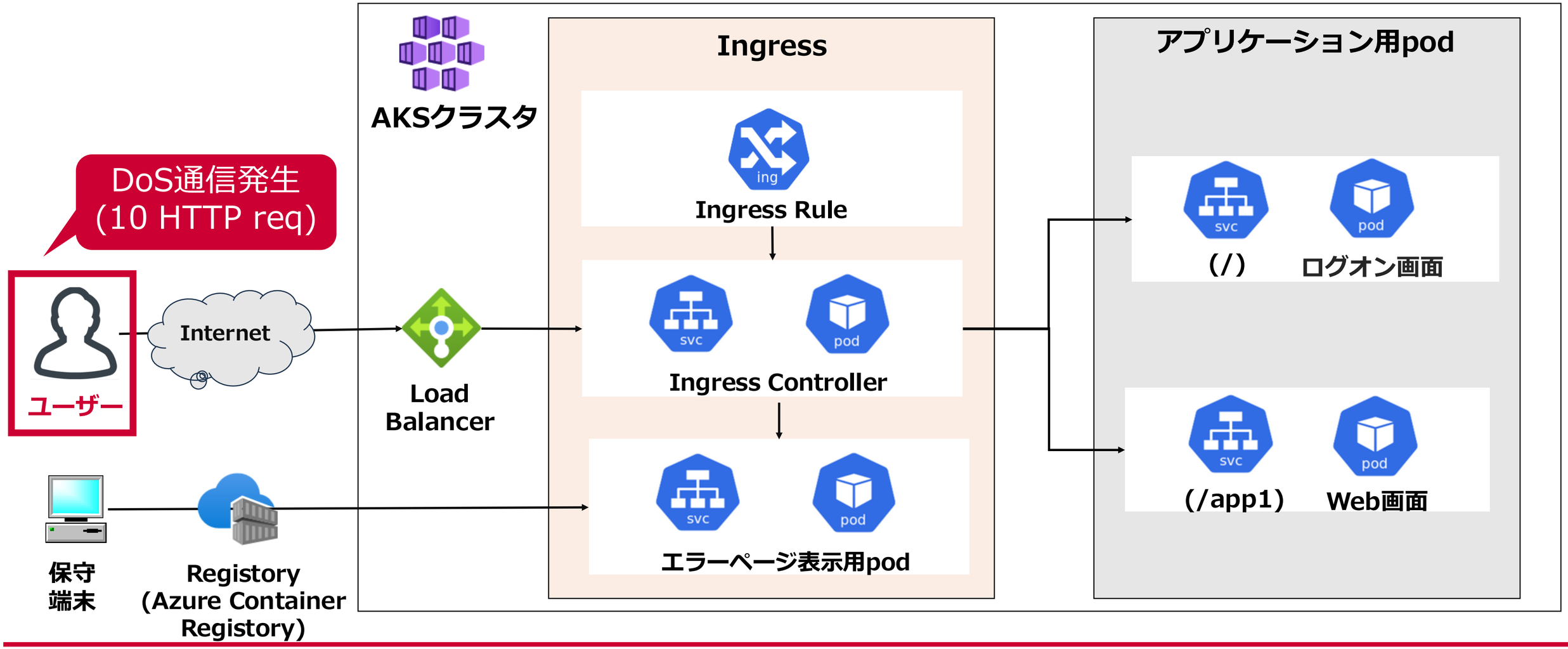
2. システム構成

3. Rate-limiting機能_利用の流れ
   a. エラーページ表示用イメージを作成
   b. エラーページ表示用イメージをAKSにデプロイ
   c. Ingress Controller(Deployment)の設定
   d. Ingress Controller(ConfigMap)の設定
   e. Ingress Ruleの設定

4. デモ

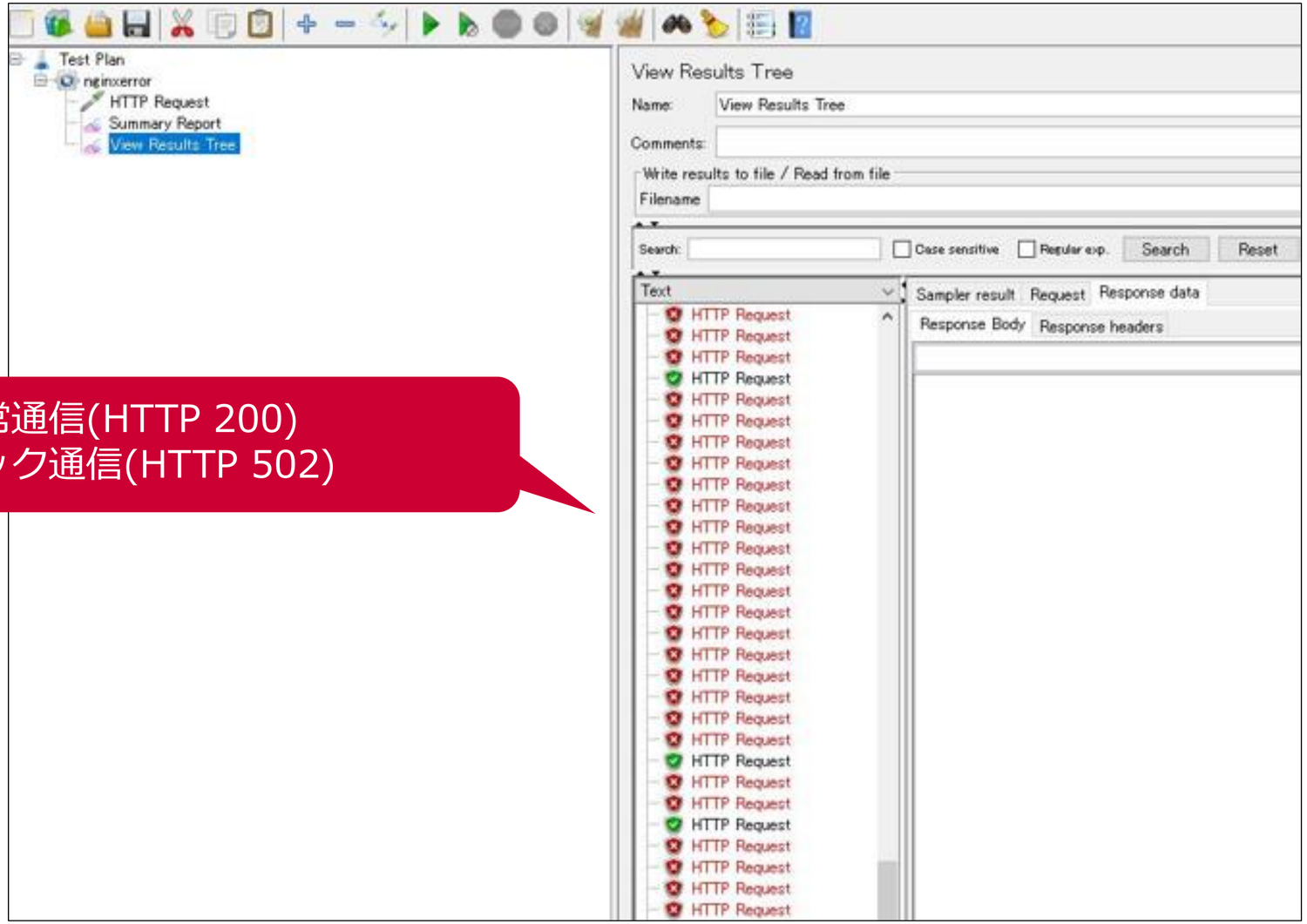
4. デモ

JMETER(通信負荷生成ツール)を用いて、ユーザーから大量アクセス時(※)の挙動を確認する。
※本件では10HTTPリクエスト/秒 、limit-rps=1で実施。



4. デモ

1秒間に10HTTPリクエストを発生させたときの結果(ユーザー側_JMETERで大量アクセス)



緑：正常通信(HTTP 200)
赤：ブロック通信(HTTP 502)

4. デモ

1秒間に10HTTPリクエストを発生させたときの結果(ユーザー側_ブラウザでアクセス)

4. デモ

1秒間に10HTTPリクエストを発生させたときのIngress Controller(サーバー側)のログ

結果　グラフ

LogMessage	TimeGenerated [ローカル時刻]	Computer
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.720	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.718	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.717	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.715	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.713	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.708	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.708	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.707	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.706	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.705	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.702	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.701	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.696	aks-agentpool-32333237-vmss...
2024/12/07 07:38:36 serving custom error response for code 502 and format text/html from file /www/502.html	2024/12/7 16:38:36.695	aks-agentpool-32333237-vmss...

service custom error response for code 502、
格納した502.htmlに関する情報が記録

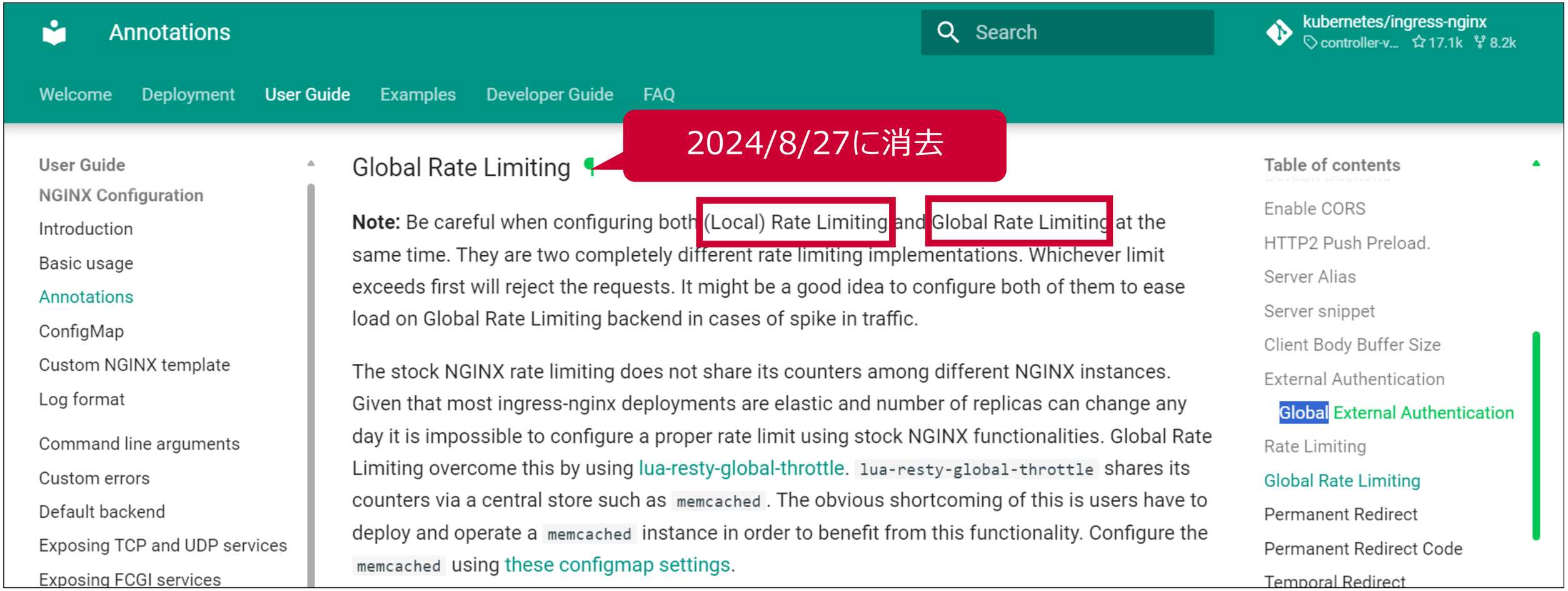
本日お伝えしたかったこと

- Ingress-Nginx ControllerのRate-Limiting機能を用いて、Kubernetesクラスタ内のコンテナをDoS攻撃から保護するための手順を紹介

- Rate-Limiting機能を用いて、ユーザ側とサーバ側(Ingress Controller)で必要以上のHTTPリクエストがブロックされることの確認

補足

DoS攻撃のような特定通信元からのアクセス制御ではなく、Ingress-Nginx Controllerでシステム全体のアクセス量を制御したい場合(サーキットブレーカ機能)、**Global Rate Limiting**機能が必要である。しかし、2024/8/27に同機能の記述が公式サイトから消去された。



ご静聴ありがとうございました