Install SSO

مستند نصب SSO با Keycloak

این مستند مراحل ایجاد و استقرار یک سرویس Single Sign-On (SSO) با استفاده از Keycloak ورژن 23 را توضیح می‌دهد. مراحل شامل ساخت Docker Image، بیلد و پوش به Docker Registry، و دیپلوی روی Kubernetes است.

مراحل

1. ساخت Docker Image

برای ایجاد Docker Image، فایل `Dockerfile` زیر استفاده می‌شود:

```
FROM quay.io/keycloak/keycloak:23.0.3 as builder

# Enable health and metrics support
ENV KC_HEALTH_ENABLED=true
ENV KC_METRICS_ENABLED=true

# Configure a database vendor
ENV KC_DB=postgres

WORKDIR /opt/keycloak
# for demonstration purposes only, please make sure to use proper certificates in production instead
RUN keytool -genkeypair -storepass password -storetype PKCS12 -keyalg RSA -keysize 2048 -dname "CN=server" -alias server -ext "SAN:c=DNS:localhost,IP:127.0.0.1" -keystore
```

```
conf/server.keystore

ADD --chown=keycloak:keycloak ./com.mavaratech.login-natcode-jar-with-dependencies.jar
/opt/keycloak/providers/com.mavaratech.login-natcode-jar-with-dependencies.jar

RUN /opt/keycloak/bin/kc.sh build

FROM quay.io/keycloak/keycloak:23.0.3

COPY --from=builder /opt/keycloak/ /opt/keycloak/

# change these values to point to a running postgres instance
ENV KC_DB=postgres
#ENV KC_DB_URL=<DBURL>
#ENV KC_DB_USERNAME=<DBUSERNAME>
#ENV KC_DB_PASSWORD=<DBPASSWORD>
ENV KC_HOSTNAME=localhost
ENV KC_HOSTNAME_STRICT_HTTPS=false
ENV TZ=Asia/Tehran
ENTRYPOINT ["/opt/keycloak/bin/kc.sh", "start", "--debug", "--spi-theme-static-max-age=-1","--
spi-theme-cache-themes=false", "--spi-theme-cache-templates=false"]
```

در این فایل، پلاگین جاوا کامپایل شده به دایرکتوری `/opt/keycloak/providers/` اضافه شده است.

دستور بیلد:

```
docker build -t keycloak-custom:23 .
```

توضیح: این دستور Docker Image را با نام `keycloak-custom` و نسخه `23` از محتوای تعریف‌شده در `Dockerfile` می‌سازد.

---

2. پوش Docker Image به Docker Registry

پوش می‌کنیم `192.168.30.26:5000` روی سرور Docker ریجستری به Docker Image ساخته‌شده را:

دستورات:

```
docker tag keycloak-custom:23 192.168.30.26:5000/keycloak-custom:23
```

توضیح: این دستور یک تگ جدید برای Docker Image ساخته‌شده ایجاد می‌کند تا برای پوش کردن به رجیستری آماده شود.

```
docker push 192.168.30.26:5000/keycloak-custom:23
```

توضیح: این دستور Docker Image را به رجیستری مشخص‌شده (`192.168.30.26:5000`) ارسال می‌کند.

---

3. دیپلوی روی Kubernetes

برای استقرار Keycloak، از فایل `keycloak-deploy.yaml` زیر استفاده می‌شود:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: keycloak-deployment
  namespace: nioc
  labels:
    app: keycloak
spec:
  replicas: 1
  selector:
    matchLabels:
      app: keycloak
  template:
    metadata:
      labels:
        app: keycloak
    spec:
      containers:
        - name: keycloak
          image: image
          volumeMounts:
            - mountPath: /opt/keycloak/themes
              name: files-vol
```

```
          subPath: keycloak/themes
      env:
        - name: KC_DB_URL
          valueFrom:
            configMapKeyRef:
              name: integration-configs
              key: keycloak-db-url
        - name: KC_DB_USERNAME
          valueFrom:
            configMapKeyRef:
              name: integration-configs
              key: keycloak-db-username
        - name: KC_DB_PASSWORD
          valueFrom:
            configMapKeyRef:
              name: integration-configs
              key: keycloak-db-password
        - name: KC_HOSTNAME
          valueFrom:
            configMapKeyRef:
              name: integration-configs
              key: keycloak-db-hostname
        - name: KEYCLOAK_ADMIN
          valueFrom:
            configMapKeyRef:
              name: integration-configs
              key: keycloak-admin-user
        - name: KEYCLOAK_ADMIN_PASSWORD
          valueFrom:
            configMapKeyRef:
              name: integration-configs
              key: keycloak-admin-password
        - name: KC_HOSTNAME_STRICT_HTTPS
          value: "false"
        - name: KC_HOSTNAME_STRICT
          value: "false"
        - name: DEBUG_PORT
          value: "*:8787"
  volumes:
    - name: files-vol
```

```
          persistentVolumeClaim:
            claimName: pvc-files


---
apiVersion: v1
kind: Service
metadata:
  name: keycloak-service
  namespace: nioc
spec:
  selector:
    app: keycloak
  ports:
    - name: app
      protocol: TCP
      port: 8443
      targetPort: 8443
    - name: debug
      protocol: TCP
      port: 8787
      targetPort: 8787
```

دستور دیپلوی:

```
kubectl apply -f keycloak-deploy.yaml
```

توضیح: این دستور فایل `keycloak-deploy.yaml` را روی کلاستر Kubernetes اعمال کرده و دیپلوی Keycloak را آغاز می‌کند.

---

4. تنظیم Ingress برای ارائه SSO

با استفاده از فایل `ingress.yaml` زیر، سرویس SSO روی ساب‌دامین ارائه می‌شود:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
```

```
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/backend-protocol: HTTPS
    nginx.ingress.kubernetes.io/keep-alive-requests: "10000"
    nginx.ingress.kubernetes.io/large-client-header-buffers: 8 512k
    nginx.ingress.kubernetes.io/max-worker-connections: "100000"
    nginx.ingress.kubernetes.io/proxy-body-size: 8m
    nginx.ingress.kubernetes.io/proxy-buffer-size: 256k
    nginx.ingress.kubernetes.io/proxy-buffering: "on"
    nginx.ingress.kubernetes.io/proxy-buffers: 4 512k
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/upstream-keepalive-connections: "2000"
    nginx.ingress.kubernetes.io/use-forwarded-headers: "true"
    nginx.ingress.kubernetes.io/use-proxy-protocol: "true"
    nginx.org/server-snippets: |
      listen 80 443 backlog=4096;
  name: integration-ingress-ssl
  namespace: nioc
spec:
  rules:
  - host: mysso.nioc.ir
    http:
      paths:
      - backend:
          service:
            name: keycloak-service
            port:
              number: 8443
        path: /
        pathType: Prefix
```

دستور اعمال:

```
kubectl apply -f ingress.yaml
```

توضیح: این دستور فایل `ingress.yaml` را روی کلاستر Kubernetes اعمال کرده و تنظیمات Ingress برای دسترسی به سرویس SSO را ایجاد می‌کند.

نتیجه

پس از اجرای مراحل بالا، سرویس SSO روی ساب‌دامین `mysso.nioc.ir` در دسترس خواهد بود. اگر مشکلی در هر مرحله پیش آمد، از لاگ‌های مرتبط برای عیب‌یابی استفاده کنید.

دستورات مفید برای بررسی:

بررسی وضعیت پادها:

```
kubectl get pods -n nioc
```

توضیح: این دستور وضعیت پادهای مستقرشده در namespace `nioc` را نمایش می‌دهد.

بررسی لاگ‌های پاد:

```
kubectl logs [POD_NAME] -n nioc
```

توضیح: این دستور لاگ‌های یک پاد مشخص را برای بررسی خطا یا مشکلات نمایش می‌دهد.

Revision #1
Created 2025-01-08 12:38:19 UTC by sh.alavian
Updated 2025-01-08 12:49:18 UTC by sh.alavian