

Side Car With Openshift

Draft

Spike Spiegel

Auguste 2018

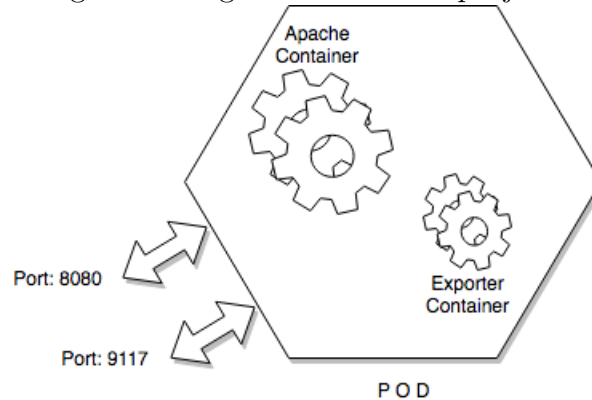
Contents

1	The Project	2
1.1	schema	2
1.2	Status On Apache	2
1.3	Secret Access	3
1.4	New Project	4
1.5	New Build	4
1.6	New Application	5
2	The Side Car	5
2.1	Export	5
2.2	ImageStream	6
2.3	BuildConfig	6
2.4	DeploymentConfig	6
2.5	Service	6
3	Geek Method	7
3.1	DeploymentConfig	7
3.2	Service	8

1 The Project

1.1 schema

Figure 1: diagram of different projects



A sidecar architecture is based on 2 containers, the main application and its addon. The objectif consists of only one process per container. In our case, the main application is the apache server and the addon is the exporter-apache. The containers share the same memory space, the same network stack, easily to deploy and manage the access from exporter-apache to apache server.

1.2 Status On Apache

status.conf

```
<Location /server-status>  
SetHandler server-status  
Order deny,allow  
Allow from all  
</Location> ExtendedStatus On
```

and *Dockerfile*

```
FROM ubuntu:latest
USER root
...
RUN a2enmod status
COPY status.conf /etc/apache2/mods-enabled/
EXPOSE 8080
USER 1001
CMD ["/usr/sbin/apache2ctl", "-DFOREGROUND"]
```

1.3 Secret Access

We firstly define our *secret file*. If the access is based on a login/password.

```
apiVersion: v1
kind: Secret
metadata:
name: github-secret
namespace: sidecar
type: kubernetes.io/basic-auth
data:
username: c3Bpa2U=
password: dmFsZW50aW5l
```

username and *password* are defined with the command

```
$ echo -n 'spike' | base64
c3Bpa2U=
$ echo -n 'valentine' | base64
dmFsZW50aW5l
```

and we run

```
$ oc create -f gitlab-secret.yaml
```

or if we use a *ssh key*, we generate the *key*, create the secret, and link the secret to the right project

```
$ ssh-keygen -C "openshift-source-builder/repo@github" \
-f repo-at-github -N ''
$ oc secrets new-sshauth repo-at-github \
--ssh-privatekey=repo-at-github
$ oc secrets link builder repo-at-github
```

When we create a new application based on this repository, it doesn't work. We have to set the new *build*

```
$ oc set build-secret --source bc/mysite repo-at-github
```

In case of our *GITLAB*, we have to use the login and password based on the *Deploy Tokens*.

1.4 New Project

Firstly, we create a new project

```
$ oc new-project sidecar \
--display-name='Side Car Project' \
--description='Side Car Project'
```

1.5 New Build

To obtain our image, we firstly

```
$ oc new-build http://192.168.0.8:8880/spike/faye.git \
--name faye
```

But to resolve the issue based on the credential, we'll attribute the *login/password* defined before and restart the build process.

```
$ oc set build-secret --source bc/faye github-secret
$ oc start-build faye
```

or directly

```
$ oc new-build http://192.168.0.8:8880/spike/faye.git \
--source-secret github-secret
--name faye
```

1.6 New Application

It's time to create our application

```
$ oc new-app faye \
--name fayeapp
$ oc status
$ oc expose service faye
$ oc get pod
$ oc get all name --selector app=cdnselect
```

2 The Side Car

2.1 Export

We firstly export our *project*.

```
$ oc get --export is,bc,dc,svc -o yaml > export.yaml
```

2.2 ImageStream

We delete *resourceVersion*, *selfLink* and *uid*. In status, we keep *dockerImageRepository* (set to "")

2.3 BuildConfig

We delete *resourceVersion*, *selfLink* and *uid*. We delete in *spec.triggers.imageChange* *lastTriggeredImageID*

2.4 DeploymentConfig

We replace *spec.template.spec.containers.image* by *faye* in the first container
We add in *spec.template.spec.container*

```
- name: apache-exporter
  image: previousnext/apache-exporter
  command: [ "apache_exporter", \
    "-scrape_uri", \
    "http://127.0.0.1:8080/server-status/?auto" ]
  ports:
    - containerPort: 9117
```

2.5 Service

We delete *resourceVersion*, *selfLink* and *uid*.
We add in *spec.ports*

```
- name: 9117-tcp
  port: 9117
```

```
protocol: TCP
targetPort: 9117
```

3 Geek Method

An other solution consists to create the application without the sidecar, and at the end, we modify the *DeploymentConfig* and *Service*.

3.1 DeploymentConfig

The *DeploymentConfig* to add the *exporter-apache* image

```
$ oc get dc
$ oc edit dc/faye
```

```
spec:
  containers:
    - command:
      - apache_exporter
      - '-scrape_uri'
      - 'http://127.0.0.1:8080/server-status/?auto'
      image: previousnext/apache-exporter
      imagePullPolicy: Always
      name: apache-exporter
      ports:
        - containerPort: 9117
          protocol: TCP
    - image: 172.30.220.103:5000/.....
      imagePullPolicy: Always
      name: faye
      ports:
        - containerPort: 8080
```

```
protocol: TCP
resources: {}
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
```

The modification of the *DeploymentConfig* will be followed by a new *build* of the image.

3.2 Service

In the common case, we don't have to modify the service, but in this case, we must access to the exporter service through the port *9117*.

```
$ oc getsvc
$ oc edit svc/faye
```

```
spec:
  ports:
    - name: 8080-tcp
      port: 8080
      protocol: TCP
      targetPort: 8080
    - name: 9117-tcp
      port: 9117
      protocol: TCP
      targetPort: 9117
```