Fluid ThinRuntime的插件开发流程和演示demo

Fluid已经通过Runtime Plugin的方式，扩展兼容多种分布式缓存引擎，包括阿里云EMR的JindoFS，开源的Alluxio，Juicefs等缓存引擎等。这极大地方便了用户通过上述缓存系统与底层存储系统交互，打通数据访问链路。

我们发现，除了上述已经集成的缓存引擎外，用户在一些云场景下存在访问自建存储系统的需求。为满足这个需求，用户须自行完成自建存储系统与Kubernetes环境的对接工作。这就需要用户根据自建存储系统的特性，设计实现自定义的CSI插件或编写Runtime Controller与Fluid对接。以上这些方式都要求用户具备Kubernetes的相关知识，增加了存储提供者的工作成本。
因此，Fluid ThinRuntime被提出以支持通用存储系统快速接入，满足用户使用Fluid访问通用存储系统数据需求。与Fluid对接后，用户可复用Fluid已有的通用能力，如：

- 通用化CSI Plugin实现
- 通用化Serverless场景适用的Fuse Sidecar架构支持
- 数据集抽象管理
- ...

基于ThinRuntime，用户可以通过编写FUSE客户端的方式，将自建存储系统对接到Fluid，具体开发指南见社区。如果有多个存储系统，或者存储系统版本出现不兼容的情况，Fluid也可以很好的通过ThinRuntime的方式，为用户提供统一的Kubernetes标准PVC接入。

下面以开源MinIO为例，展示如何通过Fluid ThinRuntime机制实现第三方存储接入及插件接入开发流程。

Fluid将会把ThinRuntime中FUSE所需的运行参数、Dataset中描述数据路径的挂载点等参数传入到ThinRuntime FUSE Pod容器中。在容器内部，需要执行参数解析脚本，并将解析完的运行时参数传递给FUSE客户端程序，由客户端程序完成Fuse文件系统在容器内的挂载。

因此，使用ThinRuntime CRD描述存储系统时，需要使用**特制的容器镜像**，镜像中需要包括以下两个程序：

- FUSE客户端程序所需的运行时参数解析脚本
- FUSE客户端程序

1. 编写参数解析脚本

对于FUSE客户端程序，在本示例中选择S3协议兼容的goofys客户端连接并挂载minio存储系统。

对于运行时所需的参数解析脚本，定义如下python脚本 `fluid-config-parse.py`：

fluid-config-parse.py Bash 复制代码

```
import json

with open("/etc/fluid/config.json", "r") as f:
    lines = f.readlines()

rawStr = lines[0]
print(rawStr)


script = """
#!/bin/sh
set -ex
export AWS_ACCESS_KEY_ID=`cat $akId`
export AWS_SECRET_ACCESS_KEY=`cat $akSecret`

mkdir -p $targetPath

exec goofys -f --endpoint "$url" "$bucket" $targetPath
"""

obj = json.loads(rawStr)

with open("mount-minio.sh", "w") as f:
    f.write("targetPath=\"%s\"\n" % obj['targetPath'])
    f.write("url=\"%s\"\n" % obj['mounts'][0]['options']['minio-url'])
    if obj['mounts'][0]['mountPoint'].startswith("minio://"):
      f.write("bucket=\"%s\"\n" % obj['mounts'][0]['mountPoint'][len("minio://"):])
    else:
      f.write("bucket=\"%s\"\n" % obj['mounts'][0]['mountPoint'])
    f.write("akId=\"%s\"\n" % obj['mounts'][0]['options']['minio-access-key'])
    f.write("akSecret=\"%s\"\n" % obj['mounts'][0]['options']['minio-access-secret'])

    f.write(script)
```

- 读取 `/etc/fluid/config.json` 文件中的json字符串，Fluid会将Fuse客户端挂载所需的参数存储并挂载到Fuse容器的 `/etc/fluid/config.json` 文件。示例如下：

/etc/fluid/config.json Bash 复制代码

```
{
    "mounts": [
        {
            "mountPoint": "minio://my-first-bucket",
            "options": {
                "minio-access-key": "/etc/fluid/secrets/minio-secret/minio-access-key",
                "minio-access-secret": "/etc/fluid/secrets/minio-secret/minio-access-secret",
                "minio-url": "http://minio:9000"
            },
            "name": "minio"
        }
    ],
    "targetPath": "/runtime-mnt/thin/default/minio-demo/thin-fuse"
}
```

- 解析json字符串，从中提取Fuse客户端挂载所需的参数。例如，上述示例中的 `url` 、 `bucket` 、 `minio-access-key` 、 `minio-access-secret` 等参数。
- 提取出所需参数后，输出挂载脚本到文件 `mount-minio.sh` 。

> 注意：在Fluid中， `/etc/fluid/config.json` 文件中仅会提供各个加密参数具体值的存储路径，因此需要参数解析脚本额外执行文件读取操作（例如：上述示例中的"export AWS_ACCESS_KEY_ID=`cat $akId`"）。

上述参数解析脚本将生成挂载脚本，如下：

mount-minio.sh Bash 复制代码

```
targetPath="/runtime-mnt/thin/default/minio-demo/thin-fuse"
url="http://minio:9000"
bucket="my-first-bucket"
akId="/etc/fluid/secrets/minio-secret/minio-access-key"
akSecret="/etc/fluid/secrets/minio-secret/minio-access-secret"

#!/bin/sh
set -ex
export AWS_ACCESS_KEY_ID=`cat $akId`
export AWS_SECRET_ACCESS_KEY=`cat $akSecret`

mkdir -p $targetPath

exec goofys -f --endpoint "$url" "$bucket" $targetPath
```

2. 制作FUSE容器镜像

接着，使用如下Dockerfile制作镜像，这里我们直接选择包含 `goofys` 客户端程序的镜像（i.e. `cloudposse/goofys` ）作为Dockerfile的基镜像：

dockerfile Bash 复制代码

```
FROM cloudposse/goofys

RUN apk add python3 bash

COPY ./fluid-config-parse.py /fluid-config-parse.py
```

使用以下命令构建并推送镜像到镜像仓库：

Bash 复制代码

```
$ IMG_REPO=<your image repo>

$ docker build -t $IMG_REPO/fluid-minio-goofys:demo .

$ docker push $IMG_REPO/fluid-minio-goofys:demo
```

3. 开发和部署MinIO的ThinRuntimeProfile

在创建Fluid Dataset和ThinRuntime 挂载Minio存储系统前，首先需要开发ThinRuntimeProfile CR资源。ThinRuntimeProfile是一种Kubernetes集群级别的 Fluid CRD资源，它描述了一类需要与Fluid对接的存储系统的基础配置（例如：容器、计算资源描述信息等）。集群管理员需提前在集群中定义若干ThinRuntimeProfile CR资源，在这之后，集群用户需要显示声明引用一个ThinRuntimeProfile CR来创建ThinRuntime，从而完成对应存储系统的挂载。

以下为MinIO存储系统的ThinRuntimeProfile CR示例（ `profile.yaml` ）：

```Bash
apiVersion: data.fluid.io/v1alpha1
kind: ThinRuntimeProfile
metadata:
  name: minio
spec:
  fileSystemType: fuse
  fuse:
    image: $IMG_REPO/fluid-minio-goofys
    imageTag: demo
    imagePullPolicy: IfNotPresent
    command:
    - sh
    - -c
    - "python3 /fluid-config-parse.py && chmod u+x ./mount-minio.sh && ./mount-minio.sh"
```

在上述示例中：

- `fileSystemType` 描述了ThinRuntime FUSE所挂载的文件系统类型(fsType)。需要根据使用的存储系统Fuse客户端程序填写，例如，goofys挂载的挂载点fsType为fuse, s3fs挂载的挂载点fsType为fuse.s3fs)
- `fuse` 描述了ThinRuntime FUSE的容器信息，包括镜像信息（`image`、`imageTag`、`imagePullPolicy`）以及容器启动命令(`command`)等。

4. demo

部署Minio存储：

minio.yaml Bash 复制代码

```yaml
apiVersion: v1
kind: Service
metadata:
  name: minio
spec:
  type: ClusterIP
  ports:
    - port: 9000
      targetPort: 9000
      protocol: TCP
  selector:
    app: minio
---
apiVersion: apps/v1 #  for k8s versions before 1.9.0 use apps/v1beta2  and before 1.8.0 use extensions/v1beta1
kind: Deployment
metadata:
  # This name uniquely identifies the Deployment
  name: minio
spec:
  selector:
    matchLabels:
      app: minio
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        # Label is used as selector in the service.
        app: minio
    spec:
      containers:
      - name: minio
        # Pulls the default Minio image from Docker Hub
        image: bitnami/minio
        env:
        # Minio access key and secret key
        - name: MINIO_ROOT_USER
          value: "minioadmin"
        - name: MINIO_ROOT_PASSWORD
          value: "minioadmin"
        - name: MINIO_DEFAULT_BUCKETS
          value: "my-first-bucket:public"
        ports:
        - containerPort: 9000
```

```
45              hostPort: 9000
```

部署成功后，Kubernetes集群内的其他Pod即可通过http://minio:9000的Minio API端点访问Minio存储系统中的数据。上述YAML配置中，我们设置Minio的用户名与密码均为minioadmin，并在启动Minio存储时默认创建一个名为my-first-bucket的存储桶，在接下来的示例中，我们将会访问my-first-bucket这个存储桶中的数据。在执行以下步骤前，首先执行以下命令，在my-first-bucket中存储示例文件：

Bash 复制代码

```
$ kubectl exec -it minio-69c555f4cf-np59j -- bash -c "echo fluid-minio-test > testfile"

$ kubectl exec -it minio-69c555f4cf-np59j -- bash -c "mc cp ./testfile local/my-first-bucket/"

$ kubectl exec -it  minio-69c555f4cf-np59j -- bash -c "mc cat local/my-first-bucket/testfile"
fluid-minio-test
```

Bash 复制代码

```
$ kubectl apply -f minio.yaml
```

部署上述ThinRuntimeProfile：

Bash 复制代码

```
$ kubectl apply -f profile.yaml
```

创建访问minio所需的凭证Secret：

Bash 复制代码

```
$ kubectl create secret generic minio-secret \

  --from-literal=minio-access-key=minioadmin \
  --from-literal=minio-access-secret=minioadmin
```

创建Dataset和ThinRuntime：

```Bash
apiVersion: data.fluid.io/v1alpha1
kind: Dataset
metadata:
  name: minio-demo
spec:
  mounts:
  - mountPoint: minio://my-first-bucket   # minio://<bucket name>
    name: minio
    options:
      minio-url: http://minio:9000  # minio service <url>:<port>
    encryptOptions:
      - name: minio-access-key
        valueFrom:
          secretKeyRef:
            name: minio-secret
            key: minio-access-key
      - name: minio-access-secret
        valueFrom:
          secretKeyRef:
            name: minio-secret
            key: minio-access-secret
---
apiVersion: data.fluid.io/v1alpha1
kind: ThinRuntime
metadata:
  name: minio-demo
spec:
  profileName: minio
```

- `Dataset.spec.mounts[*].mountPoint` 指定所需访问的数据桶(e.g. `my-frist-bucket`)
- `Dataset.spec.mounts[*].options.minio-url` 指定minio在集群可访问的URL（e.g. `http://minio:9000`）
- `ThinRuntime.spec.profileName` 指定已创建的ThinRuntimeProfile（e.g. `minio-profile`）

```Bash
$ kubectl apply -f dataset.yaml
```

数据访问应用示例

在Serverless场景下，用户只需要在应用podSpec或者podTemplateSpec中的label中配置 serverless.fluid.io/inject: "true" ，便可为Serverless应用提供数据访问能力。

使用以下命令创建数据访问应用示例：

Bash 复制代码

```
apiVersion: v1
kind: Pod
metadata:
  name: test-minio
  labels:
    serverless.fluid.io/inject: "true"
spec:
  restartPolicy: Never
  containers:
  - name: app
    image: nginx:latest
    command: ["bash"]
    args:
    - -c
    - ls -lh /data && cat /data/testfile && sleep 180
    volumeMounts:
    - mountPath: /data
      name: data-vol
  volumes:
  - name: data-vol
    persistentVolumeClaim:
      claimName: minio-demo
```

Bash 复制代码

```
$ kubectl create -f pod.yaml

$ kubectl logs test-minio -c app
total 512
-rw-r--r-- 1 root root 6 Aug 15 12:32 testfile
fluid-minio-test
```