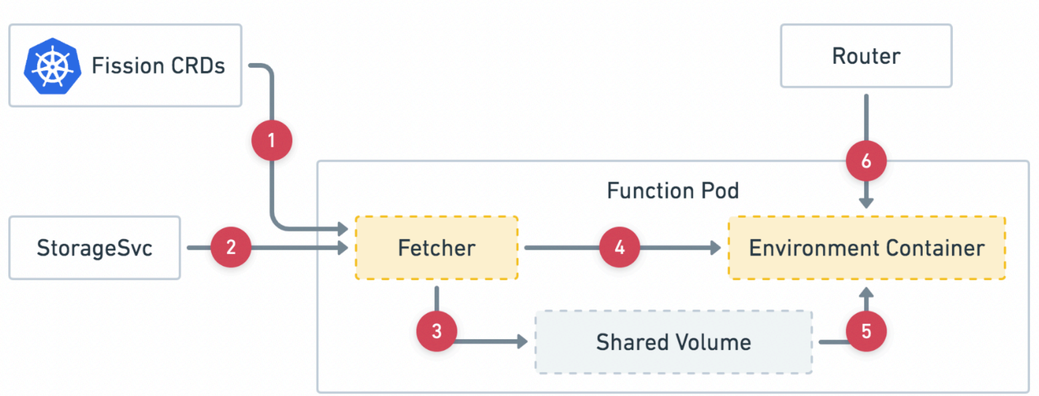
方案

思路

- fission运行函数的逻辑是：构建编程语言环境，然后让环境去运行函数；
- k8s 运行的逻辑是：启动(运行)容器就是运行函数；
- fission顺应k8s运行的逻辑
  - 创建函数：将代码上传，并构建成镜像；
    - 需要编译：将函数编译为wasm后，构建成**镜像**，存储；
    - 不用编译：直接将程序构建成镜像，存储；
  - 运行函数的时候启动容器就行；



方案

启动容器就运行函数

- fission创建package的逻辑
  - = kubectl apply -f xxx.yaml

```
pkg := &fv1.Package{
    ObjectMeta: metav1.ObjectMeta{
        Name:      pkgName,
        Namespace: pkgNamespace,
    },
    Spec: pkgSpec,
    Status: fv
        BuildS
        LastUp                                                    },
    },
}
```

```
var pkgSpec v1.PackageSpec = fv1.PackageSpec{
        Environment: fv1.EnvironmentReference{
            Namespace: envNamespace,
            Name:      envName,
        },
    }
```

-

```
pkgMetadata, err := client.V1().Package().Create(pkg)
```

```
func (c *Package) Create(f *fv1.Package) (*metav1.ObjectMeta, error) {
    err := f.Validate()
    if err != nil : nil, fv1.AggregateValidationErrors("Package", err)

    reqbody, err := json.Marshal(f)
    if err != nil : nil, err

    resp, err := c.client.Create( relativeUrl: "packages",  contentType: "application/json", reqbody)
    if err != nil : nil, err
    defer resp.Body.Close()

    body, err := handleCreateResponse(resp)
    if err != nil : nil, err

    var m metav1.ObjectMeta
    err = json.Unmarshal(body, &m)
    if err != nil : nil, err

    return &m, nil
}
```

方案

启动容器就运行函数

- kubectl run -it --rm --restart=Never wasm-hello --image=leviyanx/wasm-hello-example:v1.0 --annotations="module.wasm.image/variant=compat" /hello.wasm 20000

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: wasm-hello
  annotations:
    module.wasm.image/variant: compat-smart
spec:
  containers:
  - name: wasm-hello-example
    image: leviyanx/wasm-hello-example:v1.0
    imagePullPolicy: IfNotPresent
    command: ["/hello.wasm"]
    args: ["20000"]
```

```
root@VM-0-17-ubuntu:~# kubectl apply -f pod.yaml
pod/wasm-hello created
```

```
root@VM-0-17-ubuntu:~# kubectl logs wasm-hello
hello
20000
```

- kubectl run -it --rm --restart=Never wasi-demo --image=wasmedge/example-wasi:latest --annotations="module.wasm.image/variant=compat-smart" /wasi_example_main.wasm 50000000

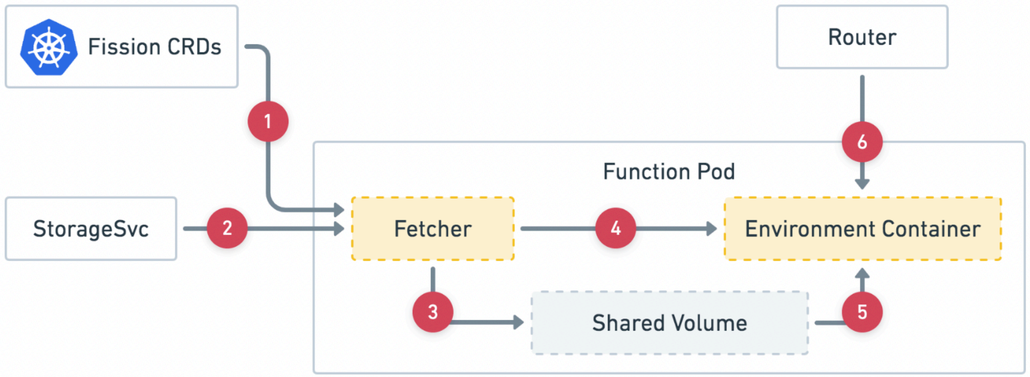
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: wasi-demo
  annotations:
    module.wasm.image/variant: compat-smart
spec:
  containers:
  - name: wasi-demo
    image: wasmedge/example-wasi:latest
    imagePullPolicy: IfNotPresent
    command: ["/wasi_example_main.wasm"]
    args: ["50000000"]
```

```
root@VM-0-17-ubuntu:~# kubectl logs wasi-demo
Random number: -1965498105
Random bytes: [235, 179, 238, 54, 220, 130, 4, 244, 26, 34, 29, 224, 14, 34, 76, 39, 119, 195, 71, 146, 2
24, 178, 114, 25, 202, 173, 177, 144, 33, 62, 141, 243, 62, 198, 115, 234, 158, 91, 125, 5, 237, 133, 37,
 207, 90, 54, 237, 99, 233, 234, 46, 50, 85, 252, 23, 13, 95, 204, 119, 173, 144, 204, 210, 249, 60, 24,
171, 188, 51, 182, 194, 89, 206, 125, 135, 217, 31, 67, 2, 254, 243, 204, 78, 47, 92, 176, 22, 126, 255,
233, 94, 255, 124, 235, 145, 164, 151, 244, 237, 154, 224, 101, 83, 28, 116, 48, 78, 38, 67, 69, 109, 145
, 32, 49, 46, 190, 92, 145, 19, 234, 139, 124, 17, 229, 132, 157, 149, 101]
Printed from wasi: This is from a main function
This is from a main function
The env vars are as follows.
The args are as follows.
/wasi_example_main.wasm
50000000
File content is This is in a file
```

方案

问题

- 函数的存储问题
  - storagesvc存储形式是把文件打个包（zip archive），而现在需要把函数程序放到镜像中；
- 镜像的拉取问题
  - 要让storagesvc能够拉取/存储镜像，提供url；
- 创建函数的问题
  - fission创建函数的时候需要指定environment，而且创建环境必须要env image（builder image是可选的）
  - 与fission运行函数的**核心逻辑**有冲突，需要改动的代码量大；
- ...




```yaml
apiVersion: v1
kind: Pod
metadata:
  name: wasm-hello
  annotations:
    module.wasm.image/variant: compat-smart
spec:
  containers:
  - name: wasm-hello-example
    image: leviyanx/wasm-hello-example:v1.0
    imagePullPolicy: IfNotPresent
    command: ["/hello.wasm"]
    args: ["20000"]
```

参考

- https://github.com/containers/crun/blob/main/docs/wasm-wasi-on-kubernetes.md
- https://github.com/containers/crun/pull/886
- https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/