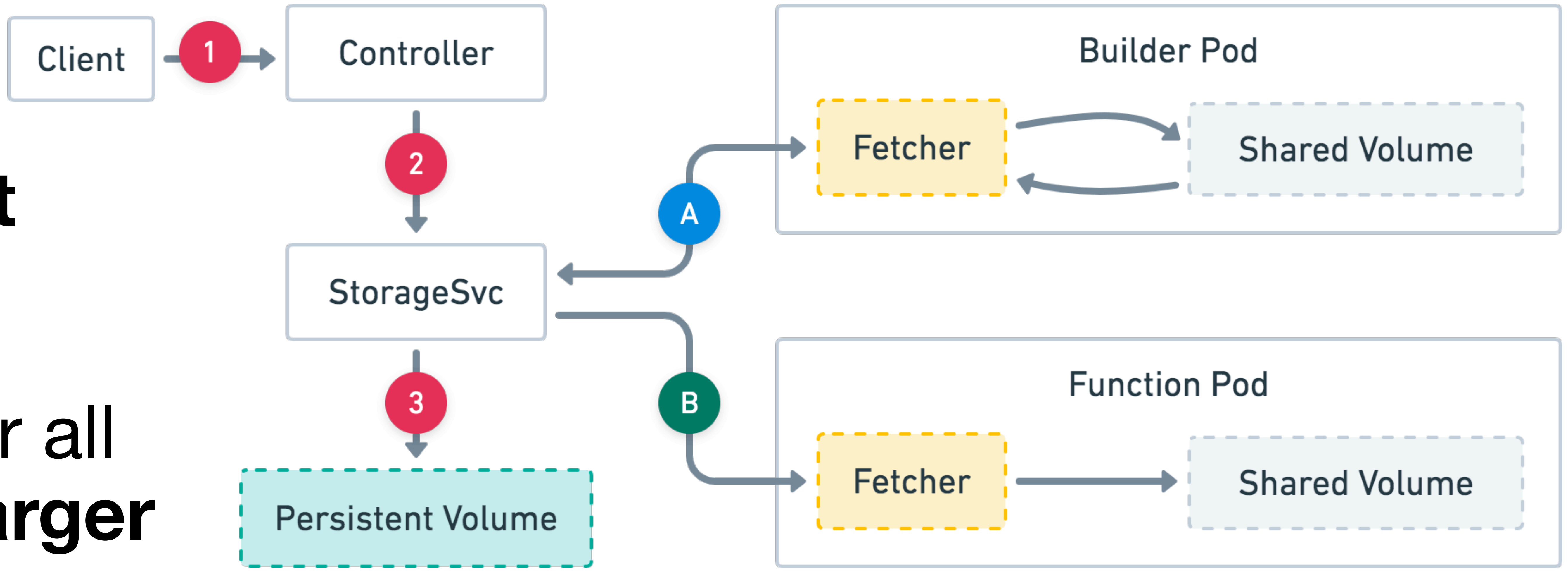
Fission Architecture

StorageSVC

- Home for **source and deployment archives**

- The storage service is the home for all archives of packages **with sizes larger than 256KB**

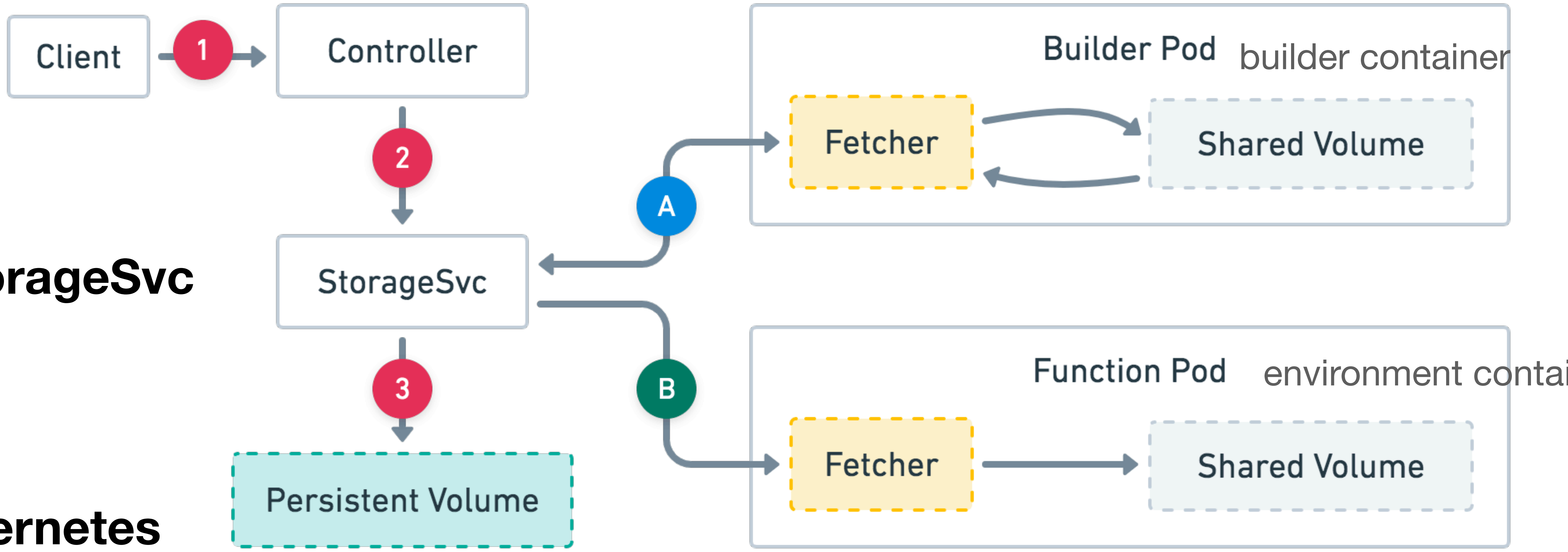
  - https://processon.com/diagraming/62f0c5096376896d38772e90



Fission Architecture

StorageSVC

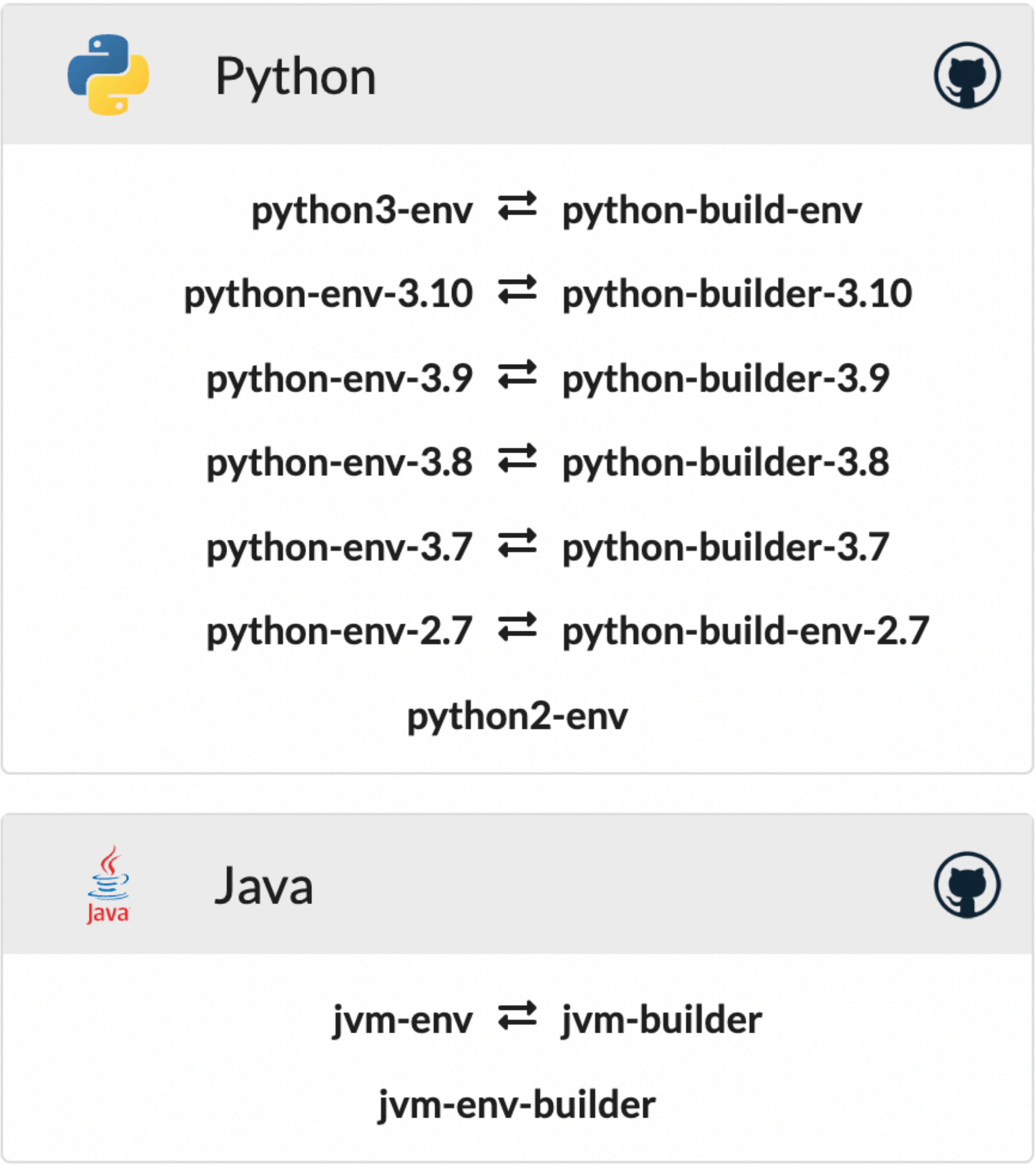
- Workflow
  - Client (CLI or HTTP requests) connects to the **StorageSvc proxy endpoint on Controller**.
  - Controller proxies **requests** to StorageSvc.
  - StorageSvc stores uploaded archives to the **Kubernetes persistent volume**.
  - (A) To build source archive into deployment archive, the Fetcher in **the builder pod** downloads source archive and save to shared volume. Once the **build process** finished, the Fetcher uploads the deployment archive to StorageSvc.
  - (B) The Fetcher inside **function pod** fetches **deployment archive** created in (A) for **environment container** to load in.



Fission Architecture

Builder Pod

- Fetcher
  - **pull source archive** from the StorageSvc (**verify** the checksum of file to ensure the integrity of file)
  - After the build process, it **uploads the deployment archive** to StorageSvc
- Builder Container
  - Builder Container **compiles function source code into executable binary/ files** and is **language-specific**.



Python

python3-env ⇄ python-build-env

python-env-3.10 ⇄ python-builder-3.10

python-env-3.9 ⇄ python-builder-3.9

python-env-3.8 ⇄ python-builder-3.8

python-env-3.7 ⇄ python-builder-3.7

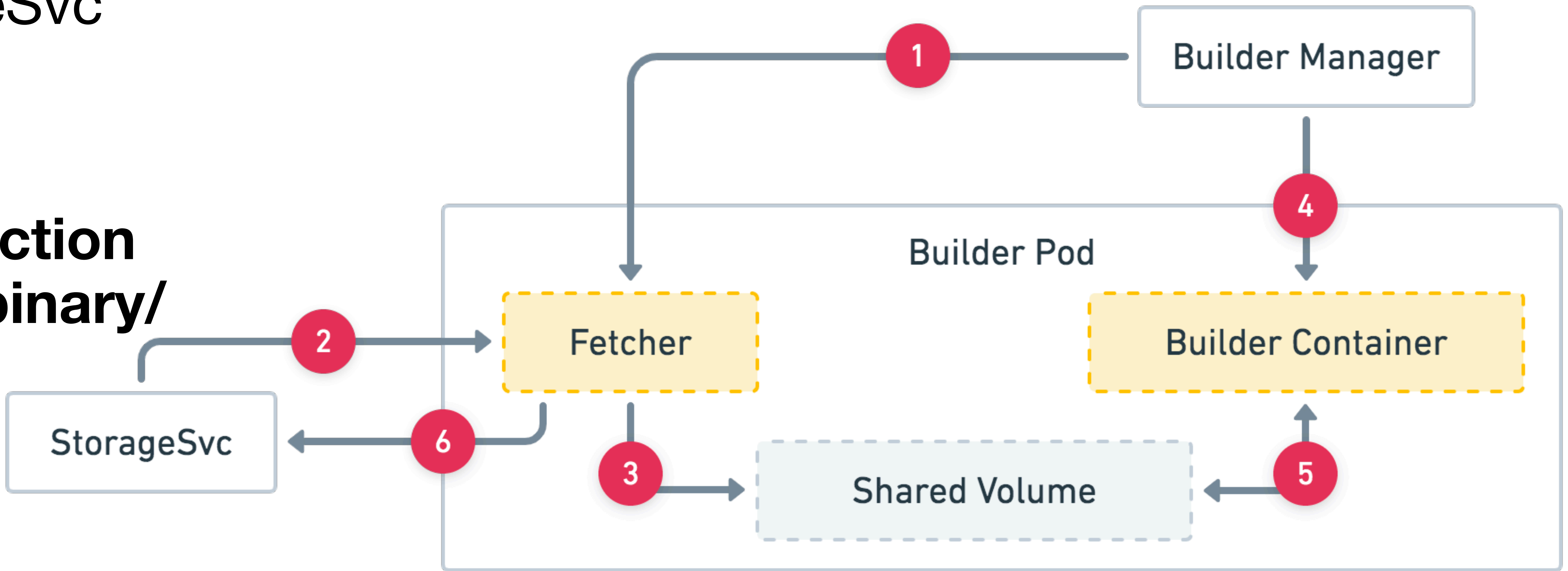
python-env-2.7 ⇄ python-build-env-2.7

python2-env

Java

jvm-env ⇄ jvm-builder

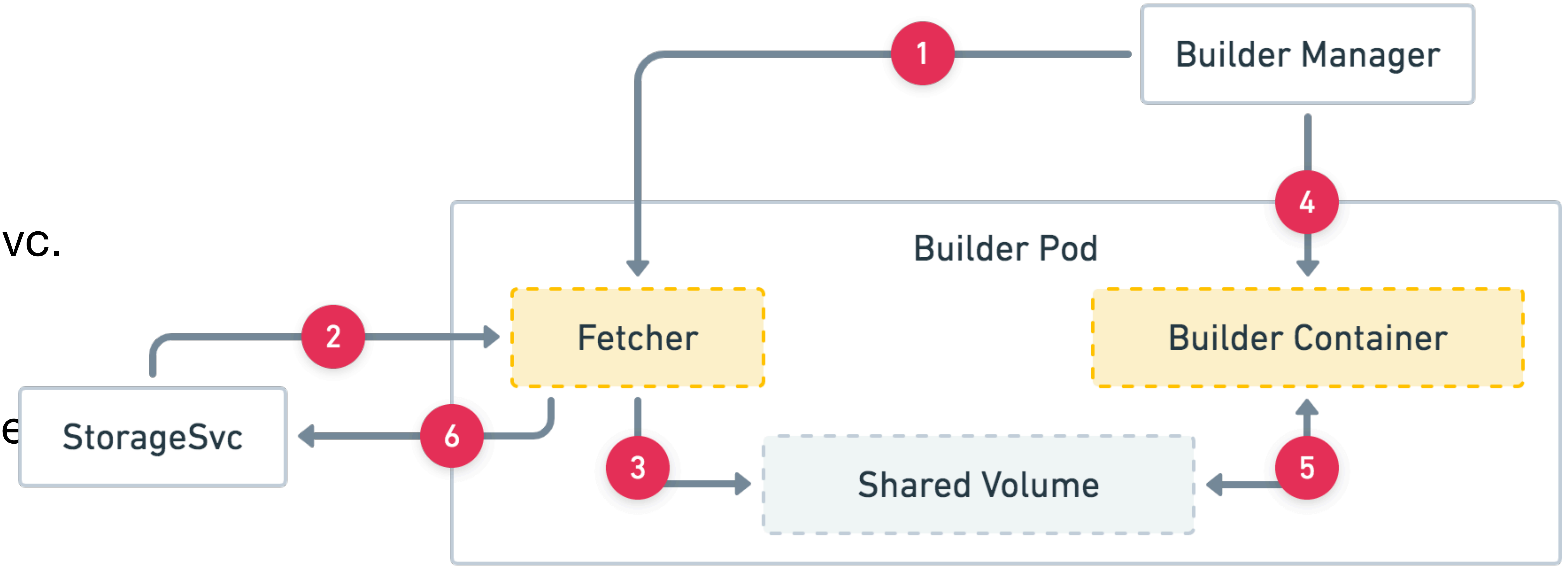
jvm-env-builder



Fission Architecture

Builder Pod

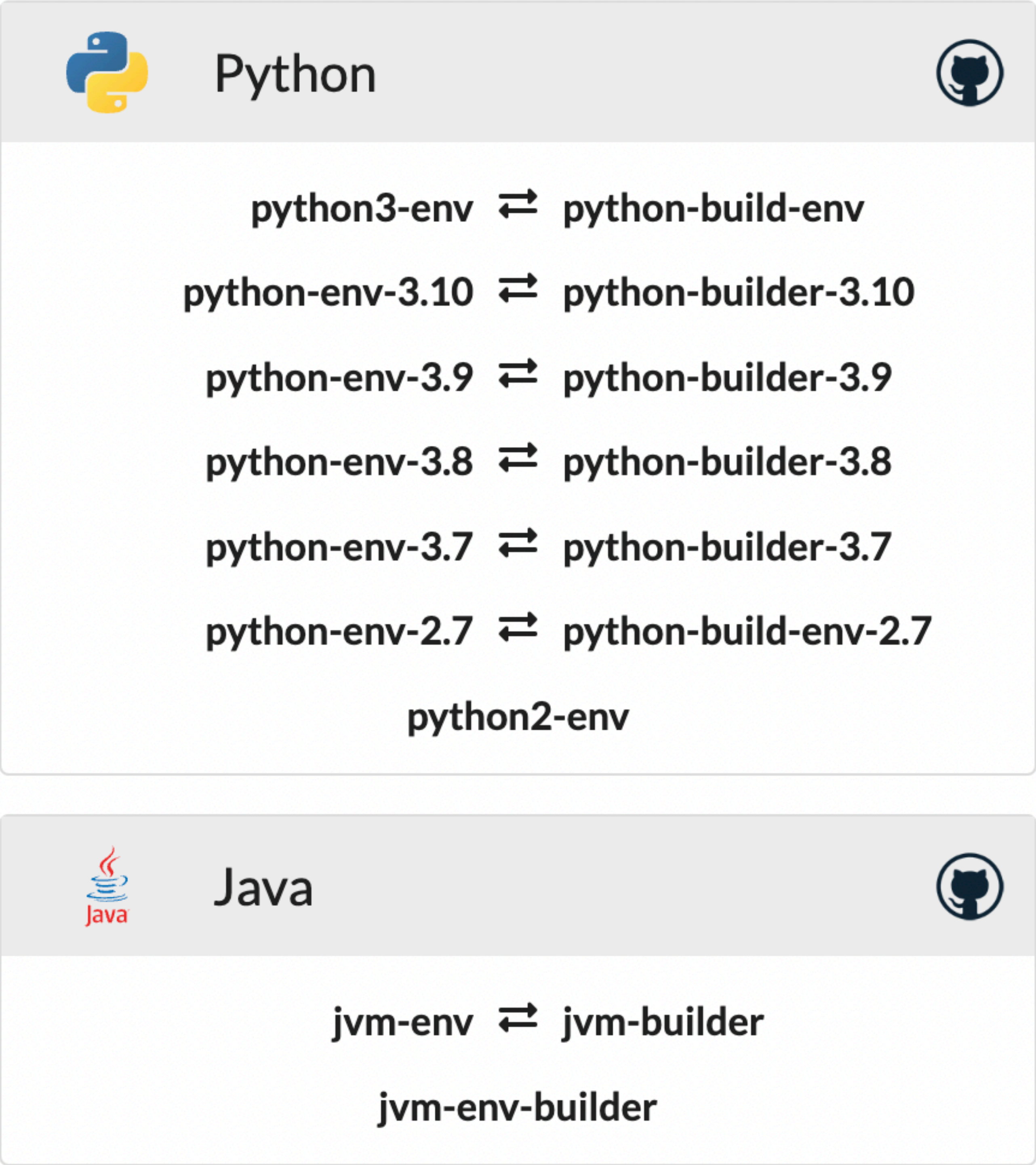
- Workflow
  - Builder Manager asks Fetcher to pull the source archive.
  - Fetcher pulls the source archive from the StorageSvc.
  - Save the archive to the shared volume.
  - Builder Manager sends a build request to the Builde Container to start the build process.
    - build(): 执行命令行命令 ls -a, javac xxx.java
  - Builder Container reads source archive from the volume, **compiles it into deployment archive.** Finally, save the result back to the share volume.
  - Builder Manager asks Fetcher to upload the deployment archive.



Fission Architecture

Function Pod

- Fetcher
  - pull **deployment archive** from the StorageSvc
- Environment Container
  - Environment Container **runs user-defined functions and is language-specific.**
  - Each environment container must contain an HTTP server and a loader for functions.



Python

python3-env ⇄ python-build-env

python-env-3.10 ⇄ python-builder-3.10

python-env-3.9 ⇄ python-builder-3.9

python-env-3.8 ⇄ python-builder-3.8

python-env-3.7 ⇄ python-builder-3.7

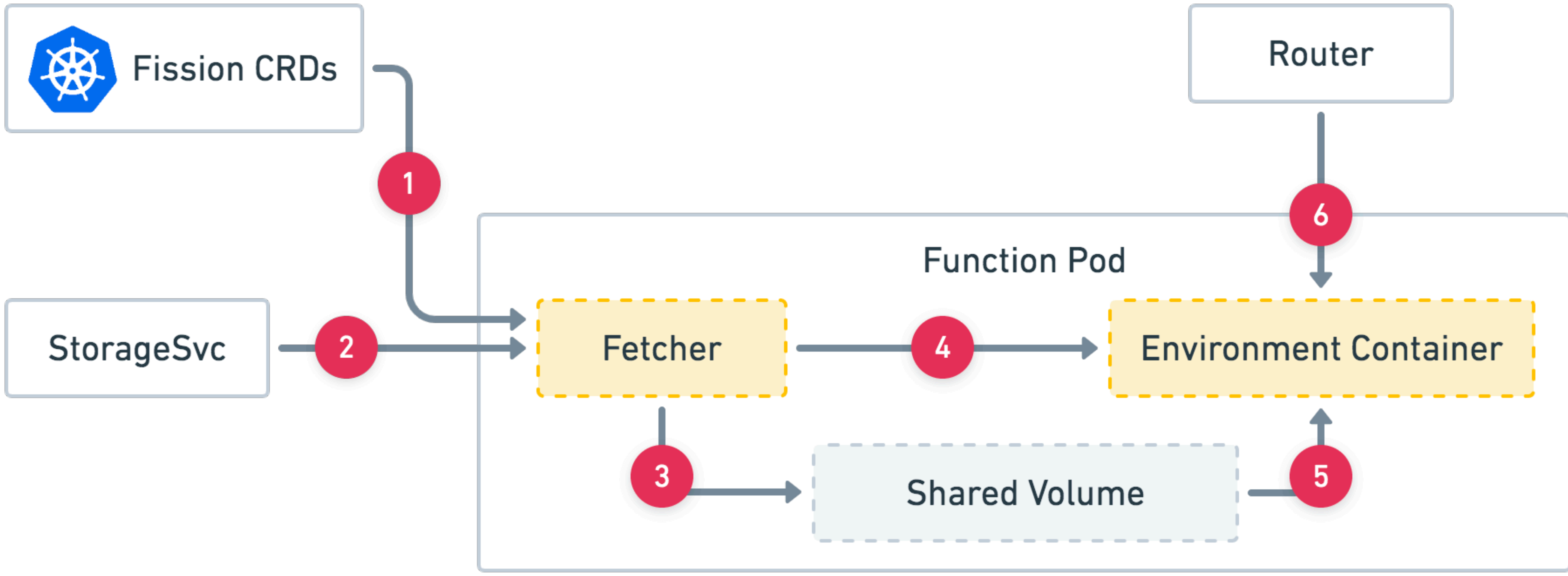
python-env-2.7 ⇄ python-build-env-2.7

python2-env

Java

jvm-env ⇄ jvm-builder

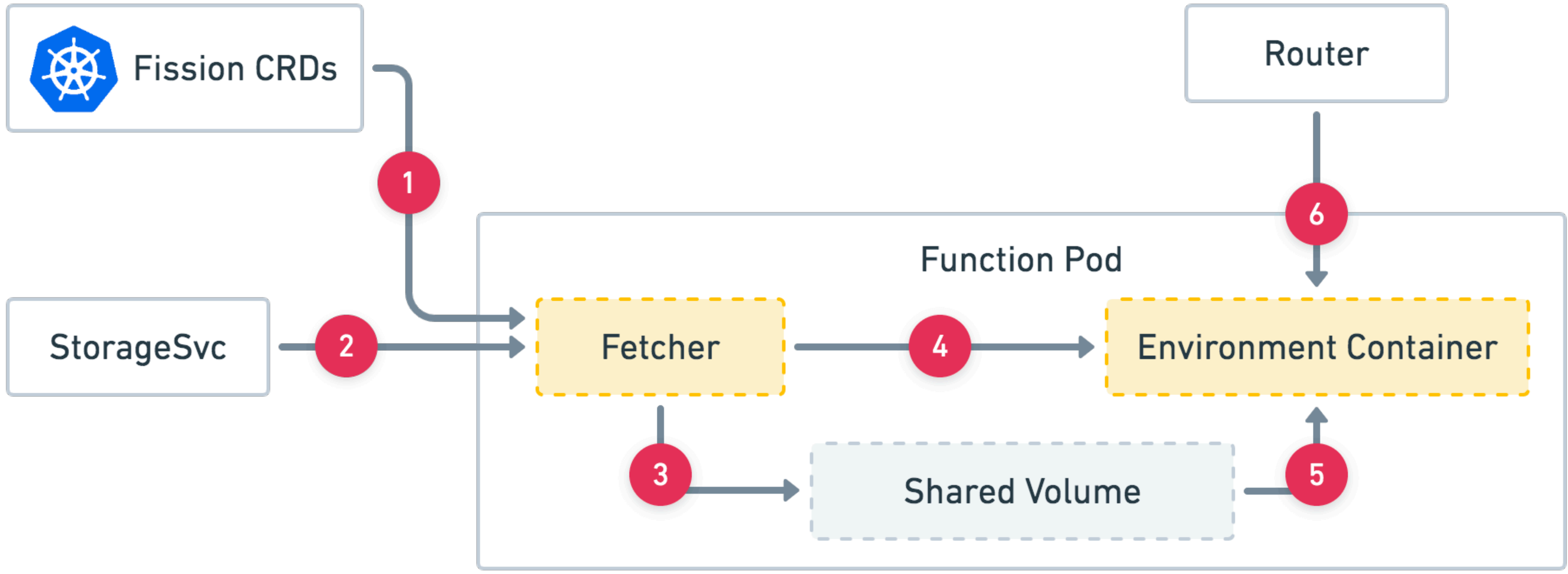
jvm-env-builder



Fission Architecture

Function Pod

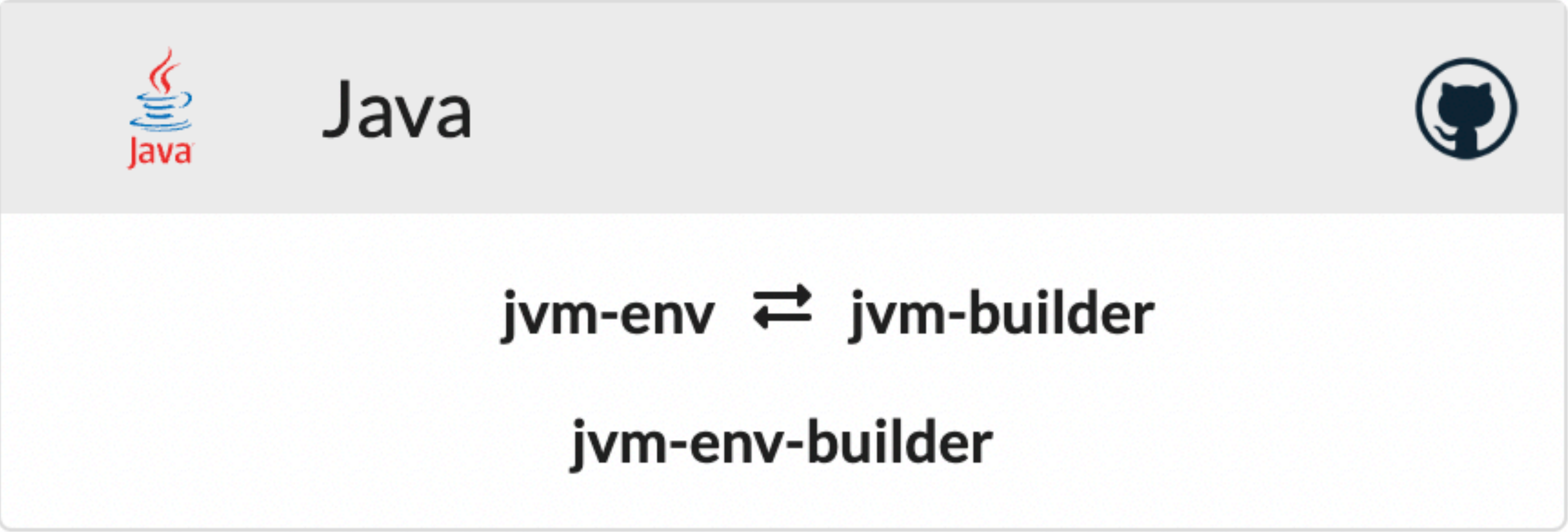
- Workflow
  - Fetcher gets the function information from the CRD.
    - objectMeta
  - Pull the deployment archive from the StorageSvc.
  - Save the archive to the shared volume.
  - Call the specialized endpoint on the environment container to start **function specialization.**
  - Environment Container loads the user function from the volume.
  - Start serving the requests from Router.



Fission Architecture

Function Pod

- fission environment create --name java --image fission/jvm-env --builder fission/jvm-builder

- 

  Java

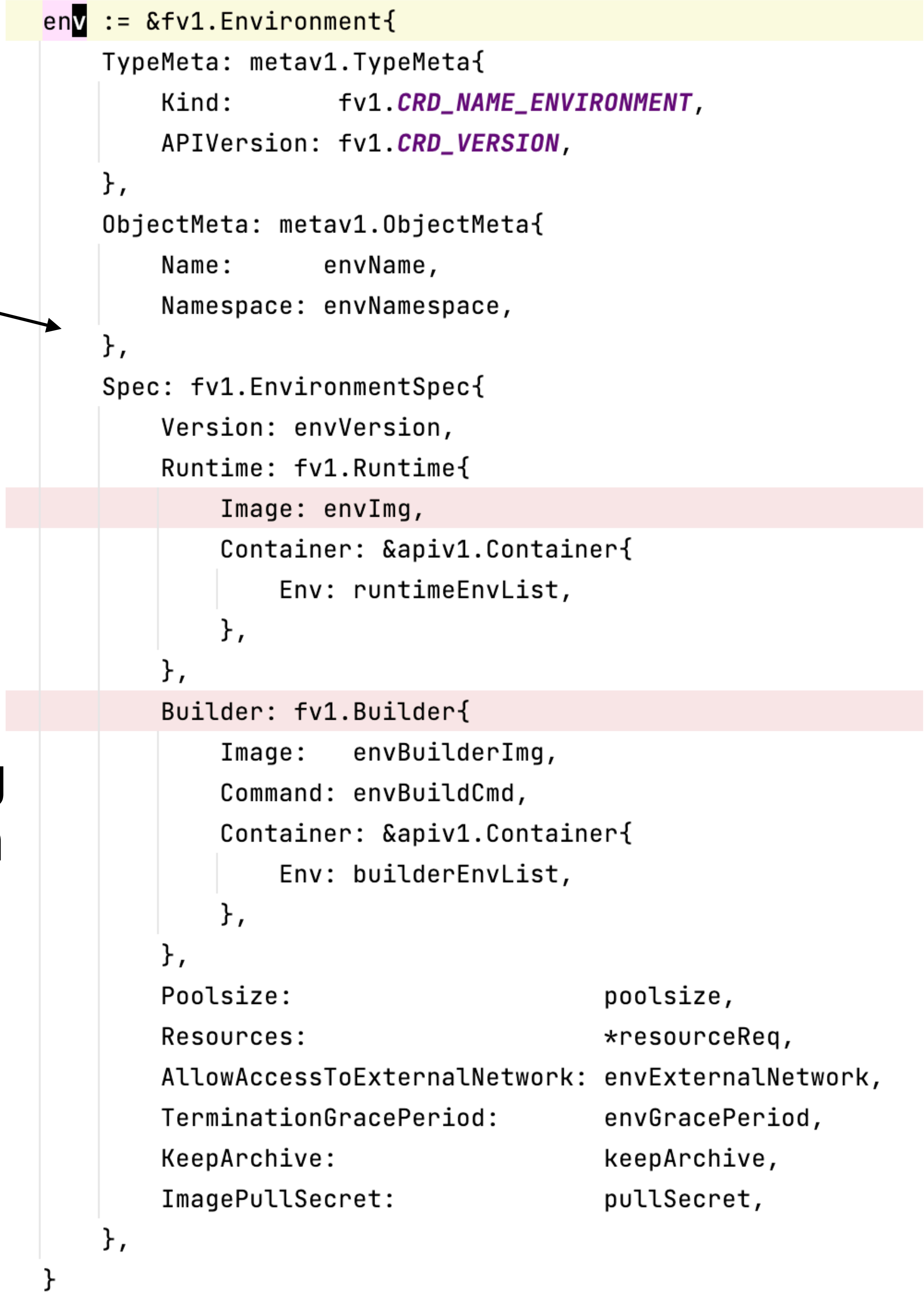
  jvm-env ⇄ jvm-builder

  jvm-env-builder

- It's a Docker image containing **a OpenJDK8 runtime, along with a dynamic loader.** A few dependencies are included in the pom.xml file.

- env的创建过程与package、function的创建过程一致；

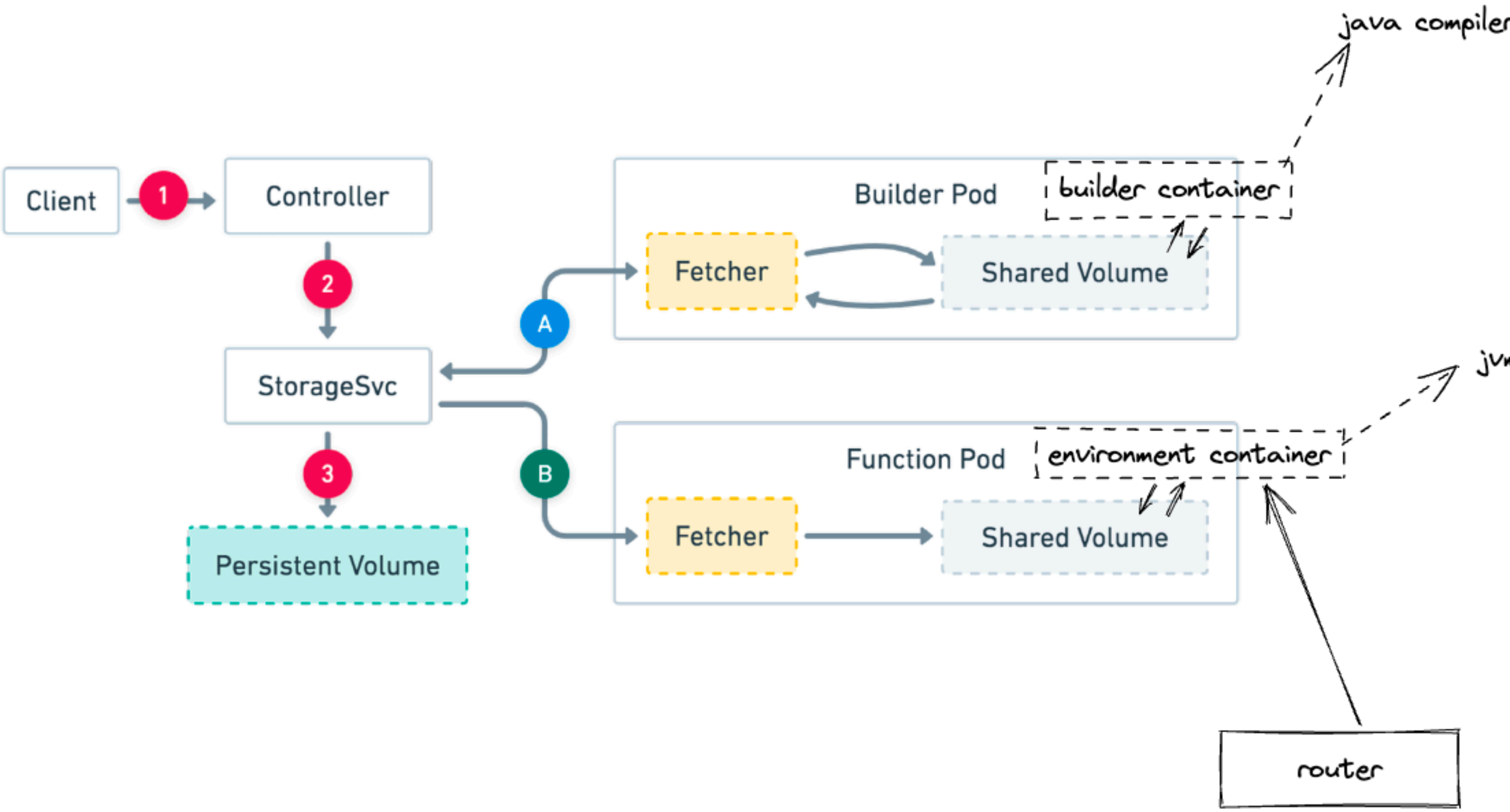
  - 发送http请求；



```
env := &fv1.Environment{
    TypeMeta: metav1.TypeMeta{
        Kind:       fv1.CRD_NAME_ENVIRONMENT,
        APIVersion: fv1.CRD_VERSION,
    },
    ObjectMeta: metav1.ObjectMeta{
        Name:      envName,
        Namespace: envNamespace,
    },
    Spec: fv1.EnvironmentSpec{
        Version: envVersion,
        Runtime: fv1.Runtime{
            Image: envImg,
            Container: &apiv1.Container{
                Env: runtimeEnvList,
            },
        },
        Builder: fv1.Builder{
            Image:   envBuilderImg,
            Command: envBuildCmd,
            Container: &apiv1.Container{
                Env: builderEnvList,
            },
        },
        Poolsize:                     poolsize,
        Resources:                    *resourceReq,
        AllowAccessToExternalNetwork: envExternalNetwork,
        TerminationGracePeriod:       envGracePeriod,
        KeepArchive:                  keepArchive,
        ImagePullSecret:              pullSecret,
    },
}
```

运行过程的推测

- 创建builder container, env container（通过builder image, env image)
  - 提供某个语言的编译、运行环境（类似于在本地搭建好了某个语言的运行环境）
- 以java为例
  - 先交给builder pod 拉取依赖，编译为xxx.class
  - 再交给function pod运行；



参考

- Builder Pod
- Function Pod
- Builder Manager
- StorageSvc
- Environments Store
- Hello World in JVM/Java on Fission