部署同步文件程序

将下载到的二进制文件(drsync)复制到/usr/bin/目录下，并给予该命令可执行权限。

将下述内容，保存到/etc/systemd/system/drsync.service中，

```
[Unit]
Description=drsync

[Service]
User=root
PermissionsStartOnly=true
ExecStart=/usr/bin/drsync -W /tmp/test -C 3 -D /tmp/test2
ExecStop=/bin/kill -TERM $MAINPID
Restart=always
RestartSec=15s
TimeoutStartSec=30s

[Install]
WantedBy=multi-user.target
```

参数解释：

```
ExecStart=/usr/bin/drsync -W /tmp/test -C 3 -D /tmp/test2 -debug
这一行为我们启动程序到命令，需要制定几个参数，根据实际情况修改：
-C int  表示要启动几个协程，默认为1个，建议设置为3-5
-W /PATH 要同步的源文件目录，要是绝对路径，必选参数
-D [ip:]/PATH  可以是本机的某个目录，也可以制定一个远程的机器上的目录（要提前配置好基于key 的ssh登陆，
建议将共享存储挂载为本地目录使用，该参数指定为本地挂载点即可。必选参数

－debug  是否显示详细信息
-delete 是否删除目标目录中在源目录中没有的文件，我们的需求不需要制定该参数
-exclude-from /PATH/excludefiles 指定要忽略掉的文件列表，该列表从一个文件里面读取

-exclude-from 所指定的文件内容格式如下（与rsync同名参数一样），一行一个规则：
atomic*tmp
atomic*tmp2
```

执行一下命令，启动服务：

```
systemctl daemon-reload
systemctl restart drsync
```

部署业务服务

首先存储会在两个地方有用到：

启动drsync文件同步服务的时候要指定，上面有提到。

在k8s启动服务的时候也需要指定共享存储的参数信息，我们以glusterfs为例,需要提前在k8s创建glusterfs的endpoints:

```
cat glustrefs-endpoint.json

{
  "kind": "Endpoints",
  "apiVersion": "v1",
  "metadata": {
    "name": "glusterfs-cluster"
  },
  "subsets": [
    {
      "addresses": [
        {
          "ip": "GLUSTERFS_IP1"
        }
      ],
      "ports": [
        {
          "port": 27004
        }
      ]
    },
    {
      "addresses": [
        {
          "ip": "GLUSTERFS_IP2"
        }
      ],
      "ports": [
        {
          "port": 27004
        }
      ]
    },
    {
      "addresses": [
        {
          "ip": "GLUSTERFS_IP3"
        }
      ],
      "ports": [
        {
          "port": 27004
        }
      ]
    }
  ]
}
```

创建endpoint:

```
kubectl create -f glustrefs-endpoint.json
```

创建，glusterfs service:

```
cat glusterfs-service.json

{
  "kind": "Service",
  "apiVersion": "v1",
  "metadata": {
    "name": "glusterfs-cluster"
  },
  "spec": {
    "ports": [
      {"port": 27004}
    ]
  }
}
```

创建service:

```
kubectl create -f glusterfs-service.json
```

注意， glusterfs的endpoint和service我们都创建到default namespace中。

业务服务deployment:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: jenkins
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: jenkins
    spec:
      containers:
        - name: jenkins
          image: nexus.hlp.fun:5000/jenkins:2.99-final
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 8080
          volumeMounts:
```

```
            - mountPath: "/var/gv0"
              name: glusterfsvol
            - name: shared-data
              mountPath: /var/jenkins_home/jobs
        - name: wrapper
          image: nexus.hlp.fun:5000/wrapper:0.0.18 #wrapper镜像
          imagePullPolicy: IfNotPresent
          volumeMounts:
            - mountPath: "/var/gv0"
              name: glusterfsvol
            - name: shared-data
              mountPath: /var/jenkins_home/jobs
          ports:
            - containerPort: 8888
      securityContext:
        runAsUser: 0
      volumes:
      - name: glusterfsvol
        glusterfs:
         endpoints: glusterfs-cluster
         path: jenkins_gv #glusterfs中的volume name
         readOnly: false
      - name: shared-data
        emptyDir: {}
      nodeSelector:
        app: jenkins
      imagePullSecrets:
      - name: go-nexus

---

apiVersion: v1
kind: Service
metadata:
  name: jenkins-svc
spec:
  type: NodePort
  ports:
    - port: 8081
      targetPort: 8080
      protocol: TCP
      nodePort: 32051
      name: jenkins
    - port: 8082
      targetPort: 8888
      protocol: TCP
      nodePort: 32052
      name: wrapper
```

```
  selector:
    name: jenkins
```

创建业务服务

```
kubectl create -f jenkins.yml
```

注意，若是存储有修改，就需要修改相关的配置信息。