POD详解三

volume

volume， 中文称为卷， 在k8s中我们经常会使用各种各样的volume类型。 其实抛开k8s，我们单纯在日程应用docker的时候，相信也已经有很多同学体验过使用-v这个参数来实现宿主机和容器之间的容器挂载。 这是一种我们最常见的volume，专门用来实现数据持久化的目的， 也被我们常称为数据卷。 比如在k8s中我们也有这种volume类型。 如下：

```
volumeMounts:
- mountPath: "/dev/shm"
name: "dshm"
volumes:
- name: "dshm"
hostPath:
path: "/dev/shm"
```

以上是我再利用k8s搭建分布式UI自动化架构中，针对chrome node的yaml文件中的一个小片段。
在一个POD中，可以使用volumes声明一个卷。 并且我们有很多种卷类型可以选择。
比如用于在POD中共享文件的emptyDir，用于将数据持久化到宿主机中的hostPath，
以及专门对接分布式存储系统的PV(PV这块略复杂，我之后有时间再详解)。 以及我们还有一些特别的类型，比如config map和secret。
而上面demo中我们使用的就是将宿主机文件目录挂载到容器中的hostPath模式。 这个的效果跟在docker中使用-v是一样的。
声明了volumes后就可以在容器定义中，使用volumeMounts来把卷挂载到容器中了。 那么在之前POD详解一种我们说过POD的side car模式是可以让POD中的目录共享文件的。 那么实现了这种方式的卷就是emptyDir。
EmptyDir类型的volume创建于pod被调度到某个宿主机上的时候，而同一个pod内的容器都能读写EmptyDir中的同一个文件。一旦这个pod离开了这个宿主机，EmptyDirr中的数据就会被永久删除。所以目前EmptyDir类型的volume主要用作临时空间，比如Web服务器写日志或者tmp文件需要的临时目录。yaml示例如下：

```
apiVersion: v1
kind: Pod
metadata:
labels:
name: test-emptypath
role: master
name: test-emptypath
spec:
containers:
- name: test-emptypath
image: registry:5000/back_demon:1.0
volumeMounts:
- name: log-storage
mountPath: /home/laizy/test/
command:
- /run.sh
volumes:
- name: log-storage
emptyDir: {}
```

为了简单期间， 上面并没有在一个POD中定义两个容器。
但是如果大家试一试启动第二个容器的话，会发现两个容器都可以使用名叫log-storage的emptyDir。

config map

config map是一种特殊的volume， 它的目的解决POD的配置管理问题。 在现实的工作环境中，我们的每一个服务都有复杂的配置文件。
比如java常用的properties文件，python常用的ini文件以及很多人特别偏好的json和xml文件。
那么这些配置一般是以文件的形式供应用程序读取。 那么按照以往在docker上的习惯，我们可能会在项目中设置一个配置模板，
然后容器运行时在外界设置非常多的环境变量。 在容器内通过sed命令来修改配置文件。 但这种方式太挫也太麻烦了，
也许有些同学会想到实现生成配置文件然后通过挂载的方式挂载到容器中使用。 那么k8s就为我们专门提供了config map中特殊的资源类型来实现这个目的。 首先，我们看看如何创建一个config map。

```
apiVersion: v1
data:
config.json: |+
{
"jenkins":{
"job_name": "340UI",
"params": {
"host":"gateway.autoui340cdh.autoui.4pd.io",
"db_port": "30317",
"branch": "release/3.4",
"thread": "10",
"deletedata": "false"
}
```

```
},
"monitor": {
"kubeconfig_path": "configs/autouicluster",
"namespaces": [
"autoui340cdh"
],
"pushgateway": "172.27.128.190:30076",
"emails": [
"sungaofei@4paradigm.com"
],
"blacklist": [
"kafka-exporter"
]
}
}

autouicluster: |
apiVersion: v1
clusters:
- cluster:
certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUR2akNDQXFhZ0F3SUJBZ0lVWDg0SFB0Uzgr
UnRiTTV6UXN5ZTFIVE9kVTQwd0RRWUpLb1pJaHZjTkFRRUwKQlFBd1pURUxNQWtHQTFVRUJoTUNRMDR4RURBT0JnTlZCQWdUQjBKbGFV
cHBibWN4RURBT0JnTlZCQWNUQjBKbAphVXBwYm1jeEREQUtCZ05WQkFvVEEyczRjekVQTUEwR0ExVUVDeE1HVTNsemRHVnRNUk13RVFZ
RFZRUURFd3ByCmRXSmxjbTVsZEdWek1CNFhEVEU1TURFeE1EQTNNemN3TUZvWERUSTBNREV3T1RBM016Y3dNRm93WlRFTE1Ba0cKQTFV
RUJoTUNRMDR4RURBT0JnTlZCQWdUQjBKbGFVcHBibWN4RURBT0JnTlZCQWNUQjBKbGFVcHBibWN4RERBSwpCZ05WQkFvVEEyczRjekVQ
TUEwR0ExVUVDeE1HVTNsemRHVnRNUk13RVFZRFZRUURFd3ByZFdKbGNtNWxkR1Z6Ck1JSUJJakFOQmdrcWhraUc5dzBCQVFFRkFBT0NB
UThBTUlJQkNnS0NBUUVBclUrTWVJaTlUcnN3dWNRSHVRY1cKVmZxaTZ4WXQ3S0UxQmhubjI4aVFNeEI3TGZDamdkZUowMGdPQ3RJUFBY
YmNYK0dTU1VCRE5FeEcvNmVEbmFkZgpyY2NOeU5IY2p0anhCbmd2amYwRmNuTWMwK2ZBcFg0cGUxd1lSUzhscXNrWVZVdzNOT0ZvR2Vq
RExQZWxEanR4Clg1dUFjeUpaMEtuM25WOFFvWnNIaGJ6enk4Q0hOVG5Zb0ZTWmJrVGxBb0YxWGl1ajd4Q2FRa2ZIMTJxdklNQUYKUDNU
cVR1eUtYZ1pkTktUdC8rZXdiNWJRRzVnQlZJTldsUXNER0xrYUdjZWNGMVlJWDRSb1E3TjZhdWxPdVpHQQpMWlkyOVJZcUJUSEpaanBI
VkhoMXJpenhGeWROdDM0MTJtVm1Cem1FSHZIV0ErGh1Mnkzc2FCOHBORktyMFl4CnJRSURBUUFCbzJZd1pEQU9CZ05WSFE4QkFmOEVC
QU1DQVFZd0VnWURWUjBUQVFIL0JBZ3dCZ0VCL3dJQkFqQWQKQmdOVkhRNEVGZ1FVc0pPdmI1aEFWNSt1N3h4RFZMaVR0bWtvRU5Fd0h3
WURWUjBqQkJnd0ZvQVVzSk92YjVoQQpWNSt1N3h4RFZMaVR0bWtvRU5Fd0RRWUpLb1pJaHZjTkFRRUxCUUFEZ2dFQkFKS25FdUNIT0c5
SU8xZkQyVTZzClFhbEp2UnMyVi9Vd3lCMm1FUzBkQ3gyWk9uYjYxM0VIdGJQNUhYU290ZGE2dE0vZGlaYWRnLzJSWlowRHdtbDMKcGUx
VDB1WEdjN1FPOGRvNUVIWFBtSHZBREUyUVRmT250RUJ5Z0RJdXR1M0M2aE52b01WM3RkWi82M21GOWFXQwp0MjBWVlkyQ0JvbkFyYjJu
eUVwS1VPMUdSbUlwL1VSNm1LejFibHNZaW5aUm5nanZsRWJFTWIwM3NZQTFIeno3CnEyazV6dnE1WTNla0cwV20vRFgwNmRCdmRzQ1JL
RFdQaDZWZUMvUjBtU25QSEVYcFNkaVBBeCsrT1JKOXlXd2cKQjRiZGxQT2lzbWlWZVk1TS94SVBtMFJ4S09GdDdkNEUzZG1oa0t0bzZs
T1hReDEzVmpQRmdNWURjNDcwRGZ3ZQpFZGc9Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0K
server: https://172.27.128.190:6443
name: kubernetes
contexts:
- context:
cluster: kubernetes
user: admin
name: kubernetes
current-context: kubernetes
kind: Config
preferences: {}
users:
- name: admin
user:
as-user-extra: {}
client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUQzekNDQXNlZ0F3SUJBZ0lVZmdJS3Y2VzlCZnV
wYmpmaVd0N253RXFrODgwd0RRWUpLb1pJaHZjTkFRRUwKQlFBd1pURUxNQWtHQTFVRUJoTUNRMDR4RURBT0JnTlZCQWdUQjBKbGFVcHB
ibWN4RURBT0JnTlZCQWNUQjBKbAphVXBwYm1jeEREQUtCZ05WQkFvVEEyczRjekVQTUEwR0ExVUVDeE1HVTNsemRHVnRNUk13RVFZRFZ
RUURFd3ByCmRXSmxjbTVsZEdWek1DQVhEVEU1TURFeE1EQTNNemN3TUZvWUR6SXhNVGd4TWpFM01EY3pOekF3V2pCck1Rc3cKQ1FZRFZ
RUUdFd0pEVGpFUU1BNEdBMVVFQ0JNSFFtVnBTbWx1WnpFUU1BNEdBMVVFQnhNSFFtVnBTbWx1WnpFWApNQlVHQTFVRUNoTU9jM2x6ZEd
WdE9tMWhjM1JsY25NeER6QU5CZ05WQkFzVEJsTjVjM1JsYlRFT01Bd0dBMVVFCkF4TUZZV1J0YVc0d2dnRWlNQTBHQ1NxR1NJYjNEUUV
CQVFVQUE0SUJEd0F3Z2dFS0FvSUJBUURycitNN1c5VGIKNU1sWXgxdEpRMGc4T3hFUG1nVmU1QkovU0JEK1AveVJETHZ0RVREbDF6Q2Z
0eHV5VmJQUmlySm02R2s2YmJKTAo4ZFhIaUc3bFpSOFJ4SGxIVXlFSi80SDhST0cwWWJSZVhOaTdBRG0xUC9oVFNxNjV0Nk1YT2dISXE
0eDJUN0RRCmxhbndHa3dTQWx6MGhJbG9ZZU03NDVjclhqMlRRbExsb0piRUlNL1EvQkNUaTN5TWYwNTJHclFLMXJheS9wdUcKc2lydkx
vNkdKMUtNR1JIV1VnR29kTVg2Uy8wdlBjb2tGN2Qvcyt6dDg4YnIxMmZjQXBPWW5yN21tamNXYkllNApLVzd4eE1pdFFCRW11eUl6OTN
JZnhKQ01VeFJFRU14VGxoL1RtNTA4RGJEV2RxQVEwWUtVdGw1eTdYR1licGc5CkRHUWVjSXZpdGZiM0FnTUJBQUdqZnpCOU1BNEdBMVV
kRHdFQi93UUVBd0lGb0RBZEJnTlZIU1VFRmpBVUJnZ3IKQmdFRkJRY0RBUVlJS3dZQkJRVUhBd0l3REFZRFZSMFRBUUgvQkFJd0FEQWR
CZ05WSFE0RUZnUVVhdXdMdzh2Rwp0cUh6S21ZNE5CY2pYSUdQRDlNd0h3WURWUjBqQkJnd0ZvQVVzSk92YjVoQVY1K3U3eHhEVkxpVHR
ta29FTkV3CkRRWUpLb1pJaHZjTkFRRUxCUUFEZ2dFQkFIWjduNkZ1QzhER3RsdzQxRjdDWXFzL0M4Um1acm12Lys1cldCaU8KYnZtWVN
xejAwdG9VR0tHMXNkZVk0UTRQWkFnRVhCUGduRjZTMzltRVVaTmVmRVdVdlpRMVRmTDlTRjYxVlRUZgpGZDdHRURGekIxQ0krTGVrY2p
xNDdBTTNITlMzVWN3bVI5Y0lrRnRjbnZ1dS92T1kxeGRZenVPQXkyOUgyck9vCmtmS3JNNkhlTXZscEdyMWN2VmhVbm5HVGpuVlNVWW9
tYVF3YU1aZUZSSUMyK0hlK1Y3OWlyTTNERFNYZGgxUE4KQlpocUxHbDBDRFVrQTZidW9ieFh4SWo1aVRLd2ZBOEh2OXMvQTlwRzJuQ1J
LaVdDQ01tU1N0UklUVTJpeUlyVQpiWkFBOStVL0dXSUxJNm5kcmhpcjBkNkEyaUVCTTB4MmlOTWVmSngvUzRZbW4wND0KLS0tLS1FTkQ
gQ0VSVElGSUNBVEUtLS0tLQo=
client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBNjYvak8xdlUyK1RKV01kYlN
VTklQRHNSRDVvRlh1UVNmMGdRL2ovOGtReTc3UkV3CjVkY3duN2Nic2xXejBZcXladWhwT20yeVMvSFZ4NGh1NVdVZkVjUjVSMU1oQ2Y
rQi9FVGh0R0cwWGx6WXV3QTUKdFQvNFUwcXV1YmVqRnpvQnlLdU1kayt3MEpXcDhCcE1FZ0pjOUlTSmFHSGpPK09YSzE0OWswSlM1YUN
XeENEUAowUHdRazR0OGpIOU9kaHEwQ3RhMnN2NmJocklxN3k2T2hpZFNqQmtSMWxJQnFIVEYra3Y5THozS0pCZTNmN1BzCjdmUEc2OWR
uM0FLVG1KNis1cG8zRm15SHVDbHU4Y1RJclVBUkpyc2lNL2R5SDhTUWpGTVVSQkRNVTVZZjA1dWQKUEEydzFuYWdFTkdDbExaZWN1MXh
tRzZZUFF4a0huQ0w0clgyOXdJREFRQUJBb0lCQVFDb1B5YzNlSmE3WXRkWgpTUGNobGFZN1dPOFU5QjVoWHU3VmJkeXpvM25wRWU0Vmp
```

mQWFJMFBTd0NSRmFtaXpiUTl0NXZzM2VwZU5IMVk4CjJtaFAyYUFVVHUxRXZWTVlrQTE2eUxGVzAyaXU5QmpEWmFYTWZaNEgxNGhqaTN
RaFlJZGxlUkVNWkZjVWo4S0EKWkVWcGxjWkZ4MWRQN0pFS2I2MjZoOHZ0RTZ1WnYyc3A3aFFEU0E0bERTT2lzSFJCMU0yUlNuTk5Hdnl
FaHM3MQpkT2ZBcXlTN3BLa0JpNEZFQW9sa3hYbGtGVHA5QmxFSlpRcTR1N01xQitCZFV5cFlDc3pnNXdzVzVtNnhmK0RkCmRMZWlwUk0
ydXhiZFFOd3NtdmpJaDhzMUlNbkF0a2hEb3FzOU9CL3ZIY1RlbjAwL2dUT2ZJMENEVjFQTGcycmcKSmhKTnhsZ0JBb0dCQVBOT3hEdEJ
pZ1RRVEFDVXZQNmhudmdEYzBVd3RhQW5oMGg3SU1ZbmRvN25KL0xQT1h0MwpwOU15Nzk0dGZtWXRHTEJOcEIvR1B5Z1c5Vk52MUpkalA
xTjNKU3B1cjh6Q1RVTkxmeHdoQmVxVnNpQ0NPNWhWCjhCb0FjV0ZmVUhtbUFySjF5cmVUbW9KVkZoTnoralpnMThmVkFrNWdXaERqRTl
IdnVtZkdmRkt4QW9HQkFQZjcKV2lQZjFzOUgwNUxSMDdKUURtTGVTSFBVMnZiL01XNk9URTlCaFNKV2NjdlI3VzcrQUFGRnM5dUtQMlg
zVjQyVwpuL25SaFJvL1gyVjNkRldmNVA3N212c24vZForNHJsU2RIeHhHckFMaDFzMnJwUXk4S0FHek8yK3NEZzZ0VmtXCmZVcXhYSnZ
ucGs1THZ1UzVOL1lMd2YySk1vVVBRQkRER3hNRVJyNG5Bb0dBREFLUVZ5aDJDcVRKaTZITDdubkYKNGhJeGgzSFBGVmUrS3NyQkpHYmd
TRStLdmthU1hORGNQT1dmeDRUUlgzUE1heTk1OFlPVXJJTHRteS9DKzdJUApkeXhEYm1QR1U4SW5sREhPMVhHZjNDT0ZobXRIUzg2Nkt
sNXBPbGc5SGJRZkgvWUdpcWREa2psbS9KRFdBZ2NuCnY1cDVJYXRKNXRsK3FmYythVTczNWhFQ2dZRUE5V1hCSC9za2dkOGNXaEJXcEF
CaEhDbklIUWdvMzRCT0ZJK3cKcUVXNFQvQ25rQUZnS3hRa1FSNFBERlJVeEx6dDRXbUxTaGF5MXZTYm5MZUhZaXhtMm9WMkt0QzZlNGI
1S2xlVQplb2thMWRleXpPcmgvRG9rc05mSitBTmFNMnExaDBHZ0gwaEEwdTk4UGNMclYwQ0xSbXNBUDd1RFNQVjhlcG40CnN1cnFic1V
DZ1lBVTk3aEJISElpWWYwVFBVYkgxenVUTGN4ZXZQTTl4MkFBQkliRnRBWm9sV1RjdklMOHNzZGkKQ003UDN5UGM5S3dBS3lVa2ZYT2R
QbmI1OHJmbUltUkFsVU9JWGhjWElPQ2FWSk0xc0pzRVRRODJDMlFTNlFacAp0azQ0cTliR21aT1VuZGgwTmtIbmJXSnJlL1Vhdnp4eDc
1T1lBYnhzejQyK2J6ODN5My9VZnc9PQotLS0tLUVORCBSU0EgUFJJVkFURSBLRVktLS0tLQo=

```
kind: ConfigMap
metadata:
name: stableconfig
```

上面是我再稳定性测试中使用的config map。 里面提供了两种配置文件， 一个是config.json，
里面提供了代码运行需要的所有信息，包括jenkins job的地址和参数以及运行时需要监控的信息。 但是在我的程序中，
我需要通过k8s的client-go来实时的监控被测k8s系统的各种事件，这样我才能在出现异常的时候实时的报警并且通过webhook触发相应的事件通知到主服务。 因此我仍然需要被测k8s系统的kubeconfig来达到这个目的。
所以上面我又定义了第二个配置，也就是名为autouicluster的kubeconfig。 然后通过kubectl create -f命令，我们即可创建出此config map，那么下一步就是在POD中使用这个配置。

```
apiVersion: batch/v1
kind: Job
metadata:
name: stable-test
spec:
template:
spec:
containers:
- name: stable
image: registry.4paradigm.com/stable_test
imagePullPolicy: Always
volumeMounts:
- name: stableconfig
mountPath: "/home/work/configs"
readOnly: false
restartPolicy: Never
imagePullSecrets:
- name: docker4paradigm
volumes:
- name: "stableconfig"
configMap:
name: stableconfig

backoffLimit: 4
parallelism: 1
completions: 100
```

上面试稳定性测试Job的定义,Job是k8s中撬动离线业务的资源类型，关于JOB的具体使用方式我们先抛开不理，我之后的文章会有这方面的介绍。 这里主要关注volumes部分， 可以看到跟上面使用hostPath和emptyDir一样。
我们在编写volume字段的时候，声明此volume是configMap类型，并且是一个名字叫stableconfig的configMap。
然后在容器中，仍然通过volumeMounts来挂载使用。 注意这里有一个readOnly的选项，表明容器是否有对此配置文件有读写权限。

PS：configMap在设计之初就为热更新做好了准备。 也就是说它希望用户在改变配置的时候，可以无需重启服务。
所以他被设计成每隔5分钟(我记得好像是~~~)就会把最新的配置刷新到容器中。 所以如果我们在需要时使用kubectl edit这样的命令修改了config map中的内容，那么它会在不久之后就将最新的配置更新到了容器中。
所以如果我们服务本身也希望实现热更新的功能。 只需要专门启动一个goroutine定时同步缓存中的配置即可。

Secret

secret与config map的使用方式几乎一模一样， 只不过它的目的是为容器挂载加密数据， 比如用户名和密码。
这里简单介绍一下我们更常用的场景--拉取私有镜像仓库中的镜像。
在k8s中，如果我们需要启动pod是需要从pod被调度到的节点上下载镜像的。 但是如果对方镜像开启了认证功能。
那么就需要我们有个机制通过验证，否则在运行POD时就会收到imagepullbackoff的异常。 使用docker的时候我们都知道只需要运行docker login命令并填写相应的用户名密码就可以了。 但是在k8s中，我们需要提供相应的secret。 首先我们需要创建这个secret， 如下：

```
kubectl create secret docker-registry docker4paradigm --docker-server=https://registry.4paradigm.com
--docker-username=docker-registry --docker-password=1qaz9ol. --docker-email=sungaofei.4paradigm.com
```

这样我们在系统中就创建好了这个叫docker4paradigm的secret了。 所以在上面的例子中我们会看到POD中有这样一段字段定义：

```
imagePullSecrets:
- name: docker4paradigm
```

这里就表示在运行容器的时候，我们使用这个secret来与远程镜像仓库通信。

尾声

大过年的就先写这些吧。 关于POD的使用方式和相应的字段其实还有很多，但我们先不继续深究下去，
下一期开始我们会开始介绍k8s是如何撬动分布式离线业务和在线服务的。