文档名称：ZDNS-Zcloud-kubernetes 本地存储
文档分类：ZDNS-Zcloud-kubernetes 文档
服务等级：
文档编号：
作　　者: 余春云
联系电话：18811483030
电子邮件：yuchunyun@zdns.cn

kubernetes 本地存储据

文档变更历史记录：

变更日期	变更人	变更内容摘要	组长确认
2019-03-11	余春云	初次建立文档	

目录

必要性

- 特殊使用场景需求，如需要个临时存储空间，运行 cAdvisor 需要能访问到 node 节点 /sys/fs/cgroup 的数据，做本机单节点的 k8s 环境功能测试等等。
- 容器集群只是做小规模部署，满足开发测试、集成测试需求。
- 作为分布式存储服务的一种补充手段，比如我在一台 node 主机上插了块 SSD，准备给某个容器吃小灶。
- 目前主流的两个容器集群存储解决方案是 ceph 和 glusterfs，二者都是典型的网络分布式存储，所有的数据读、写都是对磁盘 IO 和网络 IO 的考验，所以部署存储集群时至少要使用万兆的光纤网卡和光纤交换机。如果你都没有这些硬货的话，强上分布式存储方案的结果就是收获一个以”慢动作”见长的容器集群啦。
- 分布式存储集群服务的规划、部署和长期的监控、扩容与运行维护是专业性很强的工作，需要有专职的技术人员做长期的技术建设投入。

类型

- emptyDir
- hostPath
- local volume

emptyDir

emptyDir 类型的 Volume 在 Pod 分配到 Node 上时被创建，Kubernetes 会在 Node 上自动分配一个目录，因此无需指定宿主机 Node 上对应的目录文件。 这个目录的初始内容为空，当 Pod 从 Node 上移除时，emptyDir 中的数据会被永久删除。
注：容器的 crashing 事件并不会导致 emptyDir 中的数据被删除。

适用场景

根据官方给出的最佳使用实践的建议，emptyDir 可以在以下几种场景下使用：

- 临时空间，例如基于磁盘的合并排序
- 设置检查点以从崩溃事件中恢复未执行完毕的长计算
- 保存内容管理器容器从 Web 服务器容器提供数据时所获取的文件

默认情况下，emptyDir 可以使用任何类型的由 node 节点提供的后端存储。如果你有特殊的场景，需要使用 tmpfs 作为 emptyDir 的可用存储资源也是可以的，只需要在创建 emptyDir 卷时增加一个 emptyDir.medium 字段的定义，并赋值为”Memory”即可。

注：在使用 tmpfs 文件系统作为 emptyDir 的存储后端时，如果遇到 node 节点重启，则 emptyDir 中的数据也会全部丢失。同时，你编写的任何文件也都将计入 Container 的内存使用限制。

示例 1

```
apiVersion: v1
kind: Pod
metadata:
  name: emptydir-pod
spec:
  containers:
  - image: centos
    name: client
    command: [ "sleep", "3600" ]
    volumeMounts:
    - mountPath: /data
      name: data
  volumes:
  - name: data
    emptyDir: {}
```

示例 2

```
apiVersion: v1
kind: Pod
metadata:
  name: emptydir-memory-pod
spec:
  containers:
  - image: centos
    name: client
    command: [ "sleep", "3600" ]
    volumeMounts:
    - mountPath: /data
      name: data
  volumes:
  - name: data
    emptyDir:
      medium: Memory
```

hostPath

hostPath 类型则是映射 node 文件系统中的文件或者目录到 pod 里。在使用 hostPath 类型的存储卷时，也可以设置 type 字段，支持的类型有文件、目录、File、Socket、CharDevice 和 BlockDevice。

下面是来自官网对 hostPath 的使用场景和注意事项的介绍。

适用场景

- 当运行的容器需要访问 Docker 内部结构时，如使用 hostPath 映射/var/lib/docker 到容器
- 当在容器中运行 cAdvisor 时，可以使用 hostPath 映射/dev/cgroups 到容器中

注意事项

- 配置相同的 pod（如通过 podTemplate 创建），可能在不同的 Node 上表现不同，因为不同节点上映射的文件内容不同
- 当 Kubernetes 增加了资源敏感的调度程序，hostPath 使用的资源不会被计算在内
- 宿主机下创建的目录只有 root 有写权限。你需要让你的程序运行在 privileged container 上，或者修改宿主机上的文件权限。

示例

```
apiVersion: v1
kind: Pod
metadata:
  name: hostpath-pod
spec:
  containers:
  - image: centos
    name: client
    command: [ "sleep", "3600" ]
    volumeMounts:
    - mountPath: /data
      name: data
  volumes:
  - name: data
    hostPath:
      path: /tmp/test
      type: DirectoryOrCreate
```

emptyDir 和 hostPath 在功能上的异同分析

- 二者都是 node 节点的本地存储卷方式
- emptyDir 可以选择把数据存到 tmpfs 类型的本地文件系统中去，hostPath 并不支持这一点

- hostPath 除了支持挂载目录外，还支持 File、Socket、CharDevice 和 BlockDevice，既支持把已有的文件和目录挂载到容器中，也提供了“如果文件或目录不存在，就创建一个”的功能
- emptyDir 是临时存储空间，完全不提供持久化支持
- hostPath 的卷数据是持久化在 node 节点的文件系统中的，即便 pod 已经被删除了，volume 卷中的数据还会留存在 node 节点上

local volume

这是一个很新的存储类型，Local volume 允许用户通过标准 PVC 接口以简单且可移植的方式访问 node 节点的本地存储。 PV 的定义中需要包含描述节点亲和性的信息，k8s 系统则使用该信息将容器调度到正确的 node 节点。

解决 hostPath 的痛点

在 local volume 出现之前，statefulsets 也可以利用本地 SSD，方法是配置 hostPath，并通过 nodeSelector 或者 nodeAffinity 绑定到具体 node 上。

- 由于集群内每个节点的差异化，要使用 hostPath Volume，我们需要通过 NodeSelector 等方式进行精确调度，这种事情多了，你就会不耐烦了。
- 注意 DirectoryOrCreate 和 FileOrCreate 两种类型的 hostPath，当 Node 上没有对应的 File/Directory 时，你需要保证 kubelet 有在 Node 上 Create File/Directory 的权限。
- 另外，如果 Node 上的文件或目录是由 root 创建的，挂载到容器内之后，你通常还要保证容器内进程有权限对该文件或者目录进行写入，比如你需要以 root 用户启动进程并运行于 privileged 容器，或者你需要事先修改好 Node 上的文件权限配置。
- Scheduler 并不会考虑 hostPath volume 的大小，hostPath 也不能申明需要的 storage size，这样调度时存储的考虑，就需要人为检查并保证。
- StatefulSet 无法使用 hostPath volume，已经写好的使用共享存储的 Helm Chart 不能兼容 hostPath volume，需要修改的地方还不少，这也挺难受的。

适用场景

- 数据缓存，应用可以就近访问数据，快速处理。
- 分布式存储系统，如分布式数据库 Cassandra ，分布式文件系统 ceph/gluster

配置要求

- 使用 local-volume 插件时，要求使用到了存储设备名或路径都相对固定，不会随着

系统重启或增加、减少磁盘而发生变化。

- provisioner 配置程序仅支持发现和管理挂载点（对于 Filesystem 模式存储卷）或符号链接（对于块设备模式存储卷）。对于基于本地目录的存储卷，必须将它们通过 bind-mounted 的方式绑定到发现目录中。
- 官方推荐在使用 local volumes 时，创建一个 StorageClass 并把 volumeBindingMode 字段设置为“WaitForFirstConsumer”。虽然 local volumes 还不支持动态的 provisioning 管理功能，但我们仍然可以创建一个 StorageClass 并使用延迟卷绑定的功能，将 volume binding 延迟至 pod scheduling 阶段执行。这样可以确保 PersistentVolumeClaim 绑定策略将 Pod 可能具有的任何其他 node 节点约束也进行评估，例如节点资源要求、节点选择器、Pod 亲和性和 Pod 反亲和性。
- volumeMode 可以是 FileSystem（Default）和 Block，并且需要 enable BlockVolume Alpha feature gate。为了使用本地存储需要启动 FeatureGate:PersistentLocalVolumes 支持本地存储，1.9 是 alpha 版本，1.10 是 beta 版，默认开启，v1.9 版本需要 api-server, controller-manager, scheduler, and all kubelets 开启 feature-gates 的功能：--feature-gates="PersistentLocalVolumes=true,VolumeScheduling=true,MountPropagation=true,BlockVolume=true"

实现方式

本地持久化卷允许用户通过标准 PVC 接口以简单便携的方式访问本地存储。PV 中包含系统用于将 Pod 安排到正确节点的节点亲和性信息。
一旦配置了本地卷，外部静态配置器（provisioner）可用于帮助简化本地存储管理。请注意，本地存储配置器与大多数配置器不同，并且尚不支持动态配置。相反，它要求管理员预先配置每个节点上的本地卷，并且这些卷应该是：

1. Filesystem volumeMode（默认）PV—— 将它们挂载到发现目录下。
2. Block volumeMode PV——在发现目录下为节点上的块设备创建一个符号链接。 配置器将通过为每个卷创建和清除 PersistentVolumes 来管理发现目录下的卷。

遇到的坑

问题一：pod 处于 creating 状态，pv 处于 Available，pvc 处于 pending 状态，describe 查看 pod 日志提示
MountVolume.NewMounter initialization failed for volume "example-local-pv" : path "/data/local/ycy" does not exist
解决方案：
https://github.com/rancher/rke/issues/500
由于我们的 kubelet 运行在容器中，它默认无法看见主机的目录，需要将主机目录映射到容器中
通过 zke 部署的集群修改 cluster.yml 文件后再执行一遍 zke up --config cluster.yml 即可

services: kubelet: extra_binds: - /mnt:/mnt:rshared

问题二：pod 能正常挂载 block 块设备，但无法挂载，mount 报错
mount: /dev/xvde is write-protected, mounting read-only
mount: cannot mount /dev/xvde read-only
解决方案：
添加 pod.metadata.annotations
annotations:
apparmor should be unconfied for mounting the device inside container.
container.apparmor.security.beta.kubernetes.io/fc-container: unconfined

问题二：pod 能正常挂载 block 块设备，但无法挂载，mount 报错
mount: permission denied (are you root?)
解决方案：
添加 pod.spec.containers.securityContext
securityContext:
capabilities:
CAP_SYS_ADMIN is required for mount() syscall.
add: ["SYS_ADMIN"]

数据安全风险

local volume 仍受 node 节点可用性方面的限制，因此并不适用于所有应用程序。 如果 node 节点变得不健康，则 local volume 也将变得不可访问，使用这个 local volume 的 Pod 也将无法运行。 使用 local voluems 的应用程序必须能够容忍这种降低的可用性以及潜在的数据丢失，是否会真得导致这个后果将取决于 node 节点底层磁盘存储与数据保护的具体实现了。

局限性

- 资源利用率降低。一旦本地存储使用完，即使 CPU、Memory 剩余再多，该节点也无法提供服务
- 需要做好本地存储规划，譬如每个节点 Volume 的数量、容量等，就像原来使用存储时需要把 LUN 规划好一样，在一个大规模运行的环境，存在落地难度

最佳实践

- 对于 IO 隔离，建议每个卷使用整个磁盘
- 对于容量隔离，建议使用单个分区
- 避免重新创建具有相同节点名称的节点，而仍然存在指定了该节点亲和性的旧 PV。

否则，系统可能认为新节点包含旧的 PV。

- 对于带有文件系统的卷，建议在 fstab 条目和该挂载点的目录名称中使用它们的 UUID（例如 ls -l/dev/disk/by-uuid 的输出）。这种做法可确保即使设备路径发生变化（例如，如果 /dev/sda1 在添加新磁盘时变为/dev/sdb1），也不会错误地挂在本地卷。此外，这种做法将确保如果创建具有相同名称的另一个节点，则该节点上的任何卷都是唯一的，而不会误认为是具有相同名称的另一个节点上的卷。
- 对于没有文件系统的 raw block 卷，使用唯一的 ID 作为符号链接名称。根据您的环境，/dev/disk/by-id/中的卷 ID 可能包含唯一的硬件序列号。否则，应该生成一个唯一的 ID。符号链接名称的唯一性将确保如果创建具有相同名称的另一个节点，则该节点上的任何卷都是唯一的，而不会误认为是具有相同名称的另一个节点上的卷。

hostPath 与 local volume 在功能上的异同分析

- 二者都基于 node 节点本地存储资源实现了容器内数据的持久化功能，都为某些特殊场景下提供了更为适用的存储解决方案
- 前者时间很久了，所以功能稳定，而后者因为年轻，所以功能的可靠性与稳定性还需要经历时间和案例的历练，尤其是对 Block 设备的支持还只是 alpha 版本
- 二者都为 k8s 存储管理提供了 PV、PVC 和 StorageClass 的方法实现
- local volume 实现的 StorageClass 不具备完整功能，目前只支持卷的延迟绑定
- hostPath 是单节点的本地存储卷方案，不提供任何基于 node 节点亲和性的 pod 调度管理支持
- local volume 适用于小规模的、多节点的 k8s 开发或测试环境，尤其是在不具备一套安全、可靠且性能有保证的存储集群服务时