快照的API和实现原理

1. API

（一）url

|  |
| --- |
| /api/v1/namespaces/{namespace}/snapshots/{name} |

（二）json 文件

1. create时json实例

|  |
| --- |
| {  "kind": "Snapshot",  "apiVersion": "v1",  "metadata": {  "namespace": "b02173d486ac418886124e02be52314d",  "name": "testsnap001-12364-ec02f6e6-jb6lx"  },  "spec": {  "volumes": [  "testsnap001-12364-ec02f6e6-muagk"  ],  "restartPolicy": "OnFailure",  "podName": "testsnap001-12364-ec02f6e6-jb6lx"  }  } |

各字段含义：

1. namespace: 需要打快照的pod的namespace
2. name: snapshot的name，随便取，只要同一namespace下的snapshot的namne不要有重复的即可，否则创建会报错
3. spec.volume: 填上需要打快照的pod的spec.dataHoldVolume即可。这里snapshot的spec.volume是个数组，是出于后面可扩展性的考虑(后期可能对多个volume打快照)。当前所有的pod只挂了一个volume(即spec.dataHoldVolume)，用于数据保持，所以这里只需要填上pod的spec.dataHoldVolume即可

4）spec. restartPolicy: 填OnFailure，表示如果失败了一直重试

5）spec. podName: 填需要打快照的pod的Name

1. get时json实例

|  |
| --- |
| {  "kind": "Snapshot",  "apiVersion": "v1",  "metadata": {  "name": "testsnap001-12364-ec02f6e6-jb6lx",  "namespace": "b02173d486ac418886124e02be52314d",  "selfLink": "/api/v1/namespaces/b02173d486ac418886124e02be52314d/snapshots/testsnap001-12364-ec02f6e6-jb6lx",  "uid": "26d092b6-263d-11e6-9a6e-fa163e70fcd5",  "resourceVersion": "7050052",  "creationTimestamp": "2016-05-30T08:04:31Z"  },  "spec": {  "volumes": [  "testsnap001-12364-ec02f6e6-muagk"  ],  "restartPolicy": "OnFailure",  "podName": "testsnap001-12364-ec02f6e6-jb6lx"  },  "status": {  "volumeSnapshotStatuses": [  {  "volumeName": "testsnap001-12364-ec02f6e6-muagk",  "volumeID": "e50bd43e-c035-451b-9bb5-786ac252ab21",  "snapshotID": "dd802569-79b6-4972-878c-35f9fb381de5",  "phase": "Succeeded"  }  ]  } |

各字段含义：

1. metadata和spec字段跟创建时一样。只有status字段会变化，用于上层周期性轮询或者watch的时候检查快照是否成功
2. status. volumeSnapshotStatuses是个数组，也是出于后面扩展性的考虑。由于我们只在spec.volume里填了一个volume。所以这里也只会有一个volume的状态
3. status. volumeSnapshotStatuses. volumeName
4. status. volumeSnapshotStatuses. volumeID: 需要打快照的pod的dataHoldVolume的PV所对应的NBS的volumeID. 举例说明：如果snapshot.spec. dataHoldVolume 如果填的是”test-pv”，那么status. volumeSnapshotStatuses. volumeID就等于test-pv.spec. nceNbsStore. volumeID
5. status. volumeSnapshotStatuses.phase: 快照的状态，可以是Running, Succeeded,Failed. 分别表示：正在打快照，打快照成功，打快照失败。
6. status. volumeSnapshotStatuses. snapshotID：如果打快照成功，快照对应的NBS的快照ID
7. status.mesage: 打快照失败的时候的一些信息

二． 基本实现原理

1. 异步接口：

上层需要不断轮询或者watch来获取快照是否成功

1. 基本实现原理：
2. 具有数据保持功能的Pod的所有数据都在云硬盘上（即pod.spc.dataHoldVolume）。所以打快照是针对这个盘打快照
3. 快照功能是调用云主机的NBS接口来做的。NBS的接口是异步的。因此k8s的调用NBS的接口后会不断去check快照的状态。如果NBS那边返回快照成功，则k8s的snapshot的状态更新为成功；如果NBS那边返回快照失败，则k8s的snapshot的状态更新为失败；否则一直去checkout。上层需要有超时逻辑。
4. 当k8s的controller组件watch到有一个关于snapshot的事件(增加，更新，删除)的时候，会先判断这个snapshot的状态：

* 如果snapshot.status.volumeSnapshotStatuses[0].phase 已经为Succeeded或者Failed了，说明快照已经成功了或者失败了，就不做处理。
* 如果snapshot.status.volumeSnapshotStatuses[0].phase为running，说明之前已经调用过NBS的创建快照的接口了，NBS方面正在打快照。这时候就直接调用 NBS的接口去检查快照的状态，如果NBS那边返回快照成功或者失败，就更新状态。否则不做处理。
* 如果snapshot.status.volumeSnapshotStatuses为空，说明是个新创建的snapshot，这时会调用NBS的接口去创建快照，并更新snapshot.status.volumeSnapshotStatuses[0].phase为running；

snapshot.status.volumeSnapshotStatuses[0].volume为dataHoldVolume.spec. nceNbsStore. volumeID；

snapshot.status.volumeSnapshotStatuses[0].snapshotID为NBS返回的snapshotID

二．使用案例

1. snapshot.json 文件内容

|  |
| --- |
| {  "kind": "Snapshot",  "apiVersion": "v1",  "metadata": {  "namespace": "b02173d486ac418886124e02be52314d",  "name": "testsnap001-12364-ec02f6e6-jb6lx"  },  "spec": {  "volumes": [  "testsnap001-12364-ec02f6e6-muagk"  ],  "restartPolicy": "OnFailure",  "podName": "testsnap001-12364-ec02f6e6-jb6lx"  }  } |

1. 创建snapshot

|  |
| --- |
| # kubeclt create -f snapshot.json |

1. Get snapshot的状态
2. 成功时：

|  |
| --- |
| {  "kind": "Snapshot",  "apiVersion": "v1",  "metadata": {  "name": "testsnap001-12364-ec02f6e6-jb6lx",  "namespace": "b02173d486ac418886124e02be52314d",  "selfLink": "/api/v1/namespaces/b02173d486ac418886124e02be52314d/snapshots/testsnap001-12364-ec02f6e6-jb6lx",  "uid": "26d092b6-263d-11e6-9a6e-fa163e70fcd5",  "resourceVersion": "7050052",  "creationTimestamp": "2016-05-30T08:04:31Z"  },  "spec": {  "volumes": [  "testsnap001-12364-ec02f6e6-muagk"  ],  "restartPolicy": "OnFailure",  "podName": "testsnap001-12364-ec02f6e6-jb6lx"  },  "status": {  "volumeSnapshotStatuses": [  {  "volumeName": "testsnap001-12364-ec02f6e6-muagk",  "volumeID": "e50bd43e-c035-451b-9bb5-786ac252ab21",  "snapshotID": "dd802569-79b6-4972-878c-35f9fb381de5",  "phase": "Succeeded"  }  ]  }  } |

1. Pod或者PV不存在的导致失败，此时由于是json文件不合法法，k8s不需要调用NBS接口就可以直接返回失败。所以status里的volumeName，volumeID，snapshotID都为空。Message里有具体信息，表明是Pod不存在还是PV不存在。

|  |
| --- |
| {  "kind": "Snapshot",  "apiVersion": "v1",  "metadata": {  "name": "testsnap001-12364-ec02f6e6-jb6lx",  "namespace": "b02173d486ac418886124e02be52314d",  "selfLink": "/api/v1/namespaces/b02173d486ac418886124e02be52314d/snapshots/testsnap001-12364-ec02f6e6-jb6lx",  "uid": "26d092b6-263d-11e6-9a6e-fa163e70fcd5",  "resourceVersion": "7050052",  "creationTimestamp": "2016-05-30T08:04:31Z"  },  "spec": {  "volumes": [  "testsnap001-12364-ec02f6e6-muagk"  ],  "restartPolicy": "OnFailure",  "podName": "testsnap001-12364-ec02f6e6-jb6lx"  },  "status": {  "volumeSnapshotStatuses": [  {  "volumeName": "testsnap001-12364-ec02f6e6-muagk",  "volumeID": "e50bd43e-c035-451b-9bb5-786ac252ab21",  "snapshotID": "dd802569-79b6-4972-878c-35f9fb381de5",  "phase": "Succeeded"  }  ],  "message": "can not get corresponding data hold pv testrc-obg6k for snapshot"  }  } |

1. Json文件合法，NBS返回打快照失败，此时所以status里的volumeName，volumeID，snapshotID都有值。但是status里的phase为Failed。这种情况由于NBS只返回快照失败，并没有失败的具体原因，所以Message字段为空。

|  |
| --- |
| {  "kind": "Snapshot",  "apiVersion": "v1",  "metadata": {  "name": "testsnap001-12364-ec02f6e6-jb6lx",  "namespace": "b02173d486ac418886124e02be52314d",  "selfLink": "/api/v1/namespaces/b02173d486ac418886124e02be52314d/snapshots/testsnap001-12364-ec02f6e6-jb6lx",  "uid": "26d092b6-263d-11e6-9a6e-fa163e70fcd5",  "resourceVersion": "7050052",  "creationTimestamp": "2016-05-30T08:04:31Z"  },  "spec": {  "volumes": [  "testsnap001-12364-ec02f6e6-muagk"  ],  "restartPolicy": "OnFailure",  "podName": "testsnap001-12364-ec02f6e6-jb6lx"  },  "status": {  "volumeSnapshotStatuses": [  {  "volumeName": "testsnap001-12364-ec02f6e6-muagk",  "volumeID": "e50bd43e-c035-451b-9bb5-786ac252ab21",  "snapshotID": "dd802569-79b6-4972-878c-35f9fb381de5",  "phase": "Failed"  }  ]  }  } |

1. Delete snapshot的状态
2. 异步接口

原理：由于delete k8s的snapshot的时候，要顺带将对应的NBS的snapshot删掉，但是考虑到调用NBS接口的时候可能发生的各种异常情况（网络抖动等），删除操作做成异步，以便k8s可以在后台不断重试

1. 状态变化过程

* 正常流程：接口返回202，快照状态同步变为Terminating，snapshot. DeletionTimestamp 设置为当前时间
* 异常流程：同步返回其他错误码，snapshot. DeletionTimestamp为空

1. 删除是否成功的确认方法

需要定时轮询或者watch，查询快照状态

* 如果watch到一个delete事件，或者get的时候get不到(404错误码)，则表明删除失败
* 如果卷快照状态变为DeleteFailed，则表示删除卷快照失败。此时snapshot. DeletionTimestamp会被置为空，同时status里的message字段里有NBS那边返回的删除失败的原因。
* 示例

|  |
| --- |
| {  "kind": "Snapshot",  "apiVersion": "v1",  "metadata": {  "name": "testsnap001-12364-ec02f6e6-jb6lx",  "namespace": "b02173d486ac418886124e02be52314d",  "selfLink": "/api/v1/namespaces/b02173d486ac418886124e02be52314d/snapshots/testsnap001-12364-ec02f6e6-jb6lx",  "uid": "26d092b6-263d-11e6-9a6e-fa163e70fcd5",  "resourceVersion": "7050052",  "creationTimestamp": "2016-05-30T08:04:31Z"  },  "spec": {  "volumes": [  "testsnap001-12364-ec02f6e6-muagk"  ],  "restartPolicy": "OnFailure",  "podName": "testsnap001-12364-ec02f6e6-jb6lx"  },  "status": {  "volumeSnapshotStatuses": [  {  "volumeName": "testsnap001-12364-ec02f6e6-muagk",  "volumeID": "e50bd43e-c035-451b-9bb5-786ac252ab21",  "snapshotID": "dd802569-79b6-4972-878c-35f9fb381de5",  "phase": "DeleteFailed"  }  ],  "message": "snapshot was in use, can not delete it"  }  } |