oVirt

Incremental Backup in oVirt

Nir Soffer
Principal Software Engineer
nsoffer@redhat.com

Daniel Erez
Senior Software Engineer
derez@redhat.com

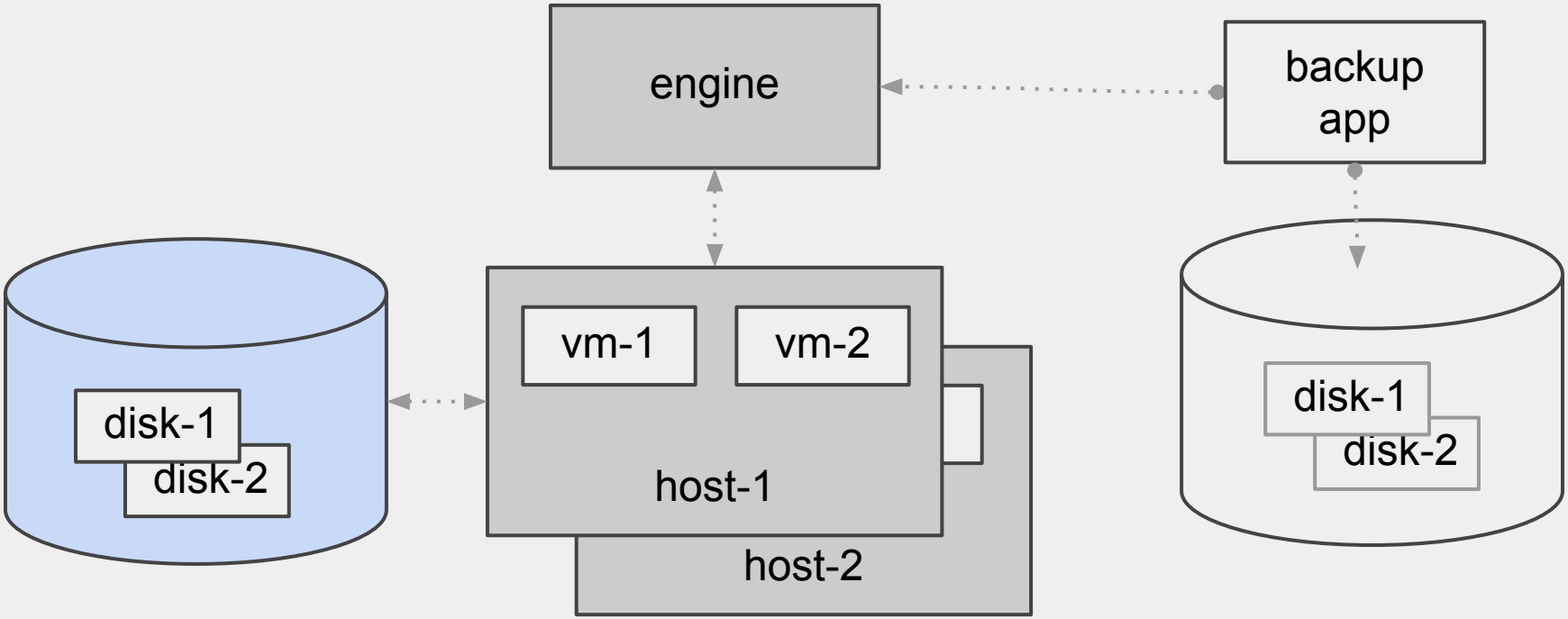
Eyal Shenitzky
Senior Software Engineer
eshenitz@redhat.com

01/2020

Agenda for our time journey

1. oVirt backup APIs
2. Incremental backup API
3. Under the hood
4. Using engine backup API
5. Using imageio backup API
6. Nuts and bolts
7. Backup data path
8. Troubleshooting

oVirt from 10,000 meters above



Old backup APIs

Backup appliance

Backup application runs in a VM

Backup appliance - backup flow

1. Take a snapshot
2. Attach disk snapshot to backup VM
3. Inside the backup VM: copy the entire disk
4. Detach the disk snapshot from the backup VM
5. Delete the snapshot

Image transfer API

Download and upload snapshots via imageio REST API.

uploading and downloading images is done by ovirt-imageio package, available since ovirt 4.0.

Download snapshot - backup flow

1. Take a snapshot
2. Download the snapshot (qcow2)
3. Delete an older snapshot

Upload disk - restore flow

1. Prepare a disk for restore (backup app)
2. Upload a disk or a chain of snapshots (raw, qcow2)
3. Create a VM from the disk

Incremental backup API

Incremental backup API

Changed block tracking

Will be in tech preview in oVirt 4.4 - requires libvirt 6.0.z and qemu 4.2.

Full backup flow

1. Start a backup
2. Download disk/s (raw)
3. Stop a backup

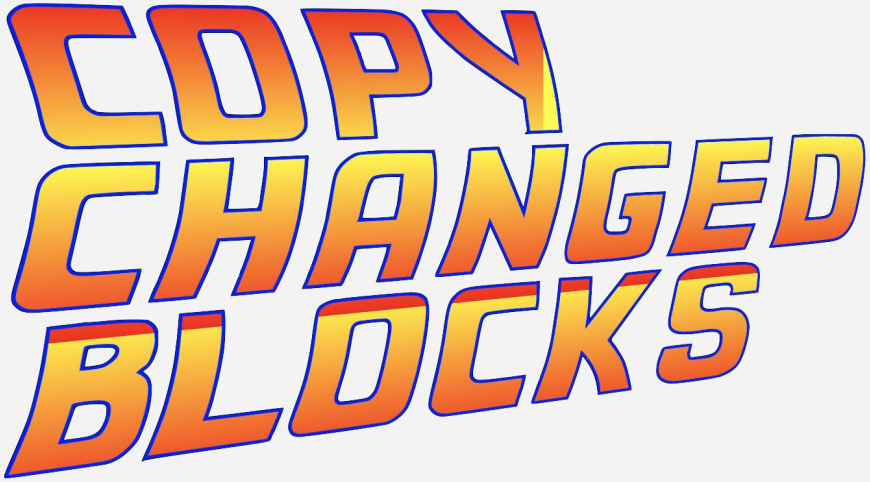
Incremental backup flow

1. Start an incremental backup
2. Download the changed blocks (raw)
3. Stop the backup

Restore from incremental backup flow

1. Prepare a disk for restore (backup app)
2. Upload the disk (raw)
3. Attach the disk to a new VM

Incremental backup - advantages



Speed up incremental backup by copying only blocks that changed since the last backup.

Incremental backup - advantages

Speed up full backup by copying only the data extents and skipping zero extents.

Incremental backup - advantages

No need to create and delete a snapshot.

Incremental backup - advantages

Access raw guest data in backup and restore regardless of the underlying disk format and snapshots.

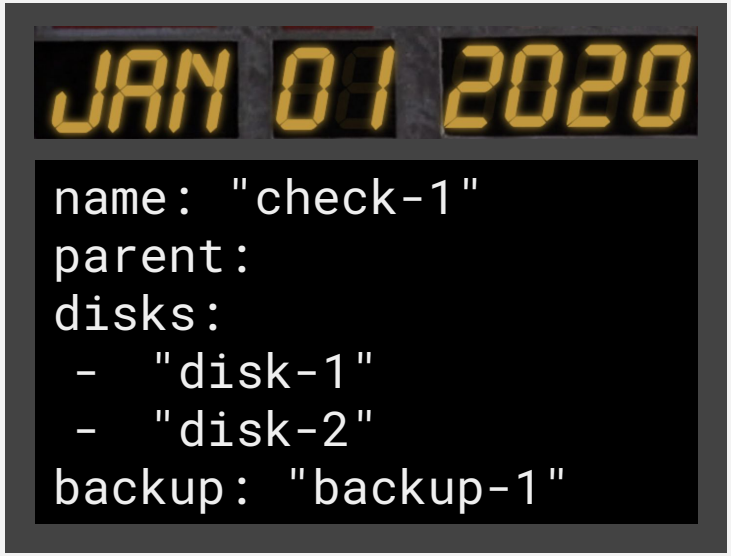
Incremental backup - advantages

imageio client library can upload/download raw/qcow2 images (including the backing files).

Under the hood

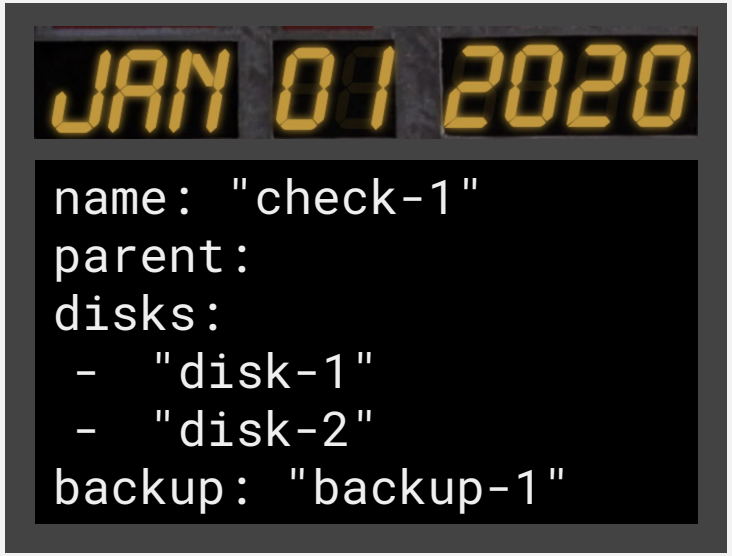
Checkpoints

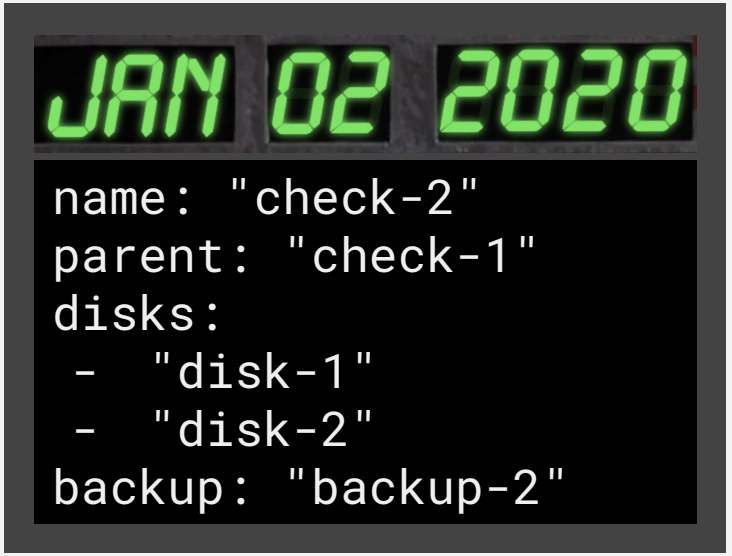
Checkpoints



For every backup, the system creates a new checkpoint recording the disks being backed up.

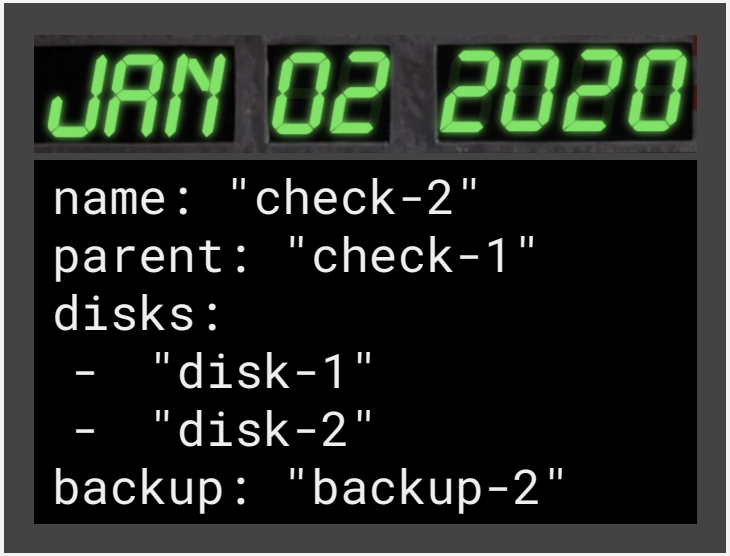
Checkpoints

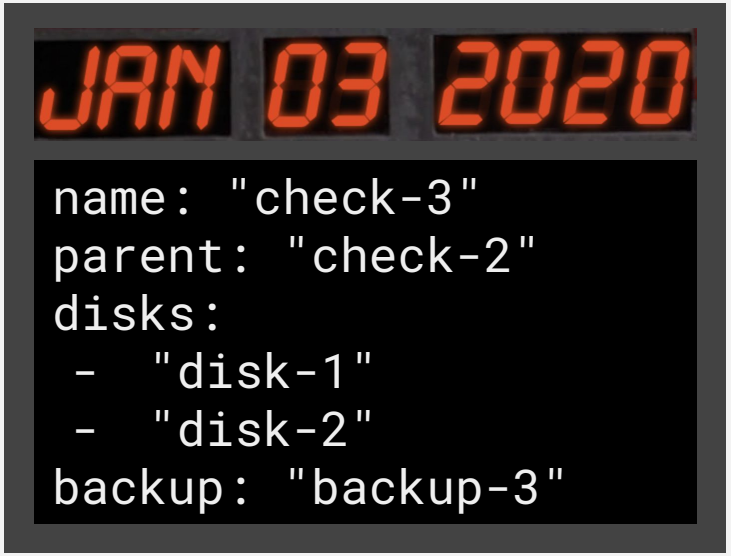




Every checkpoint keep the parent checkpoint id.

Checkpoints

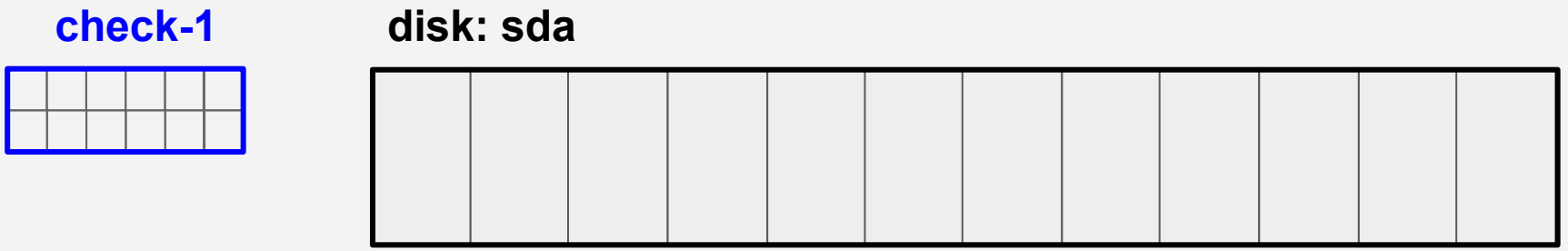




Engine persists the checkpoints in the database. Libvirt keeps the checkpoints on the host when the VM is running.

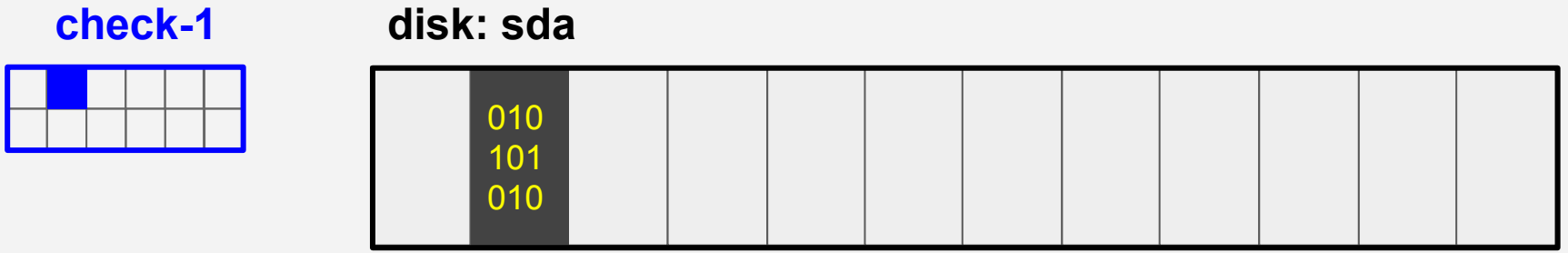
Dirty bitmaps

Dirty bitmaps



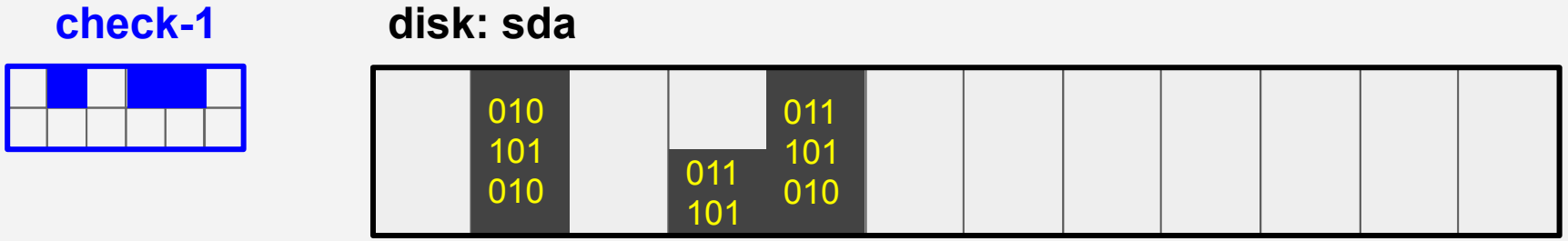
For every checkpoint, libvirt creates a new dirty bitmap, recording changes after the checkpoint was created. Disk sda has no changes yet.

Dirty bitmaps



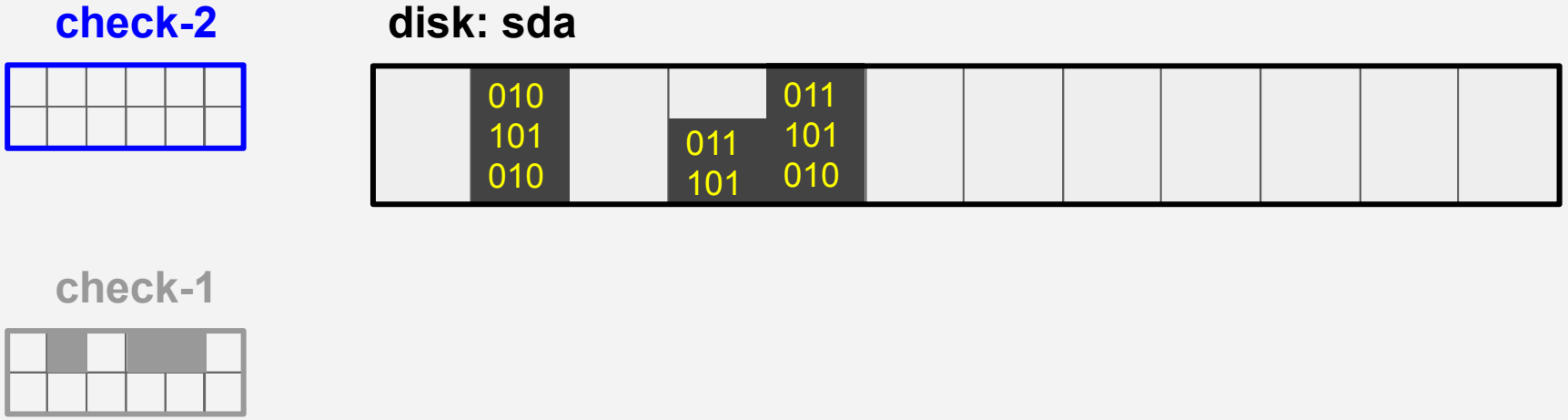
When the guest writes to the disk, qemu record the changed blocks in the bitmap check-1.

Dirty bitmaps



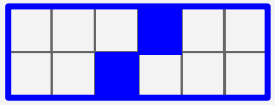
By default every bit represents one cluster, if the guest writes less than a cluster the entire cluster is considered as dirty.

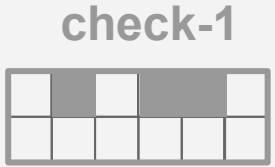
Dirty bitmaps

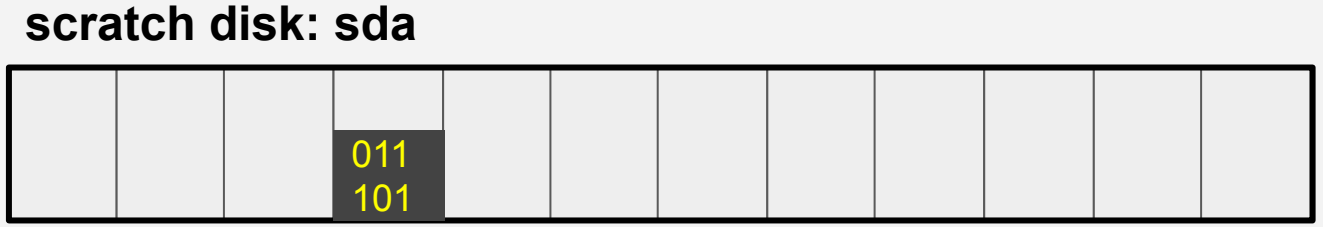


When libvirt creates a new checkpoint, it deactivates the current bitmap (check-1) and create new bitmap (check-2) for recording new changes.

Scratch disk







During backup, if guest writes data to cluster referenced by check-1, older data is copied to a scratch disk. Scratch disk is deleted after backup.

Dirty extents

check-1

```
[
  {"start": 0, "length": 65536, "dirty": false},
  {"start": 65356, "length": 65536, "dirty": true},
  {"start": 131072, "length": 65536, "dirty": false},
  {"start": 196608, "length": 131072, "dirty": true},
  {"start": 327680, "length": 458752, "dirty": false}
]
```

When getting dirty extents, every extent represent consecutive bits in the bitmap. During incremental backup we copy only the dirty extents.

Using oVirt backup API

oVirt backup API

Incremental backup

How to start full backup?

Starting full backup

```
# Start VM backup for the given disks of a VM
# using the backup service.

backup = backups_service.add(
    Backup(
        disks=[
            Disk(id=disk.id),
            ...
```

Backup status

```
# Wait until backup is ready.

while backup.phase != BackupPhase.READY:
    time.sleep(1)
    backup = backup_service.get()
```

Backup status

```
# to_checkpoint_id will be used as
# from_checkpoint_id in the next incremental
# backup.

from_checkpoint_id = backup.to_checkpoint_id
```

Starting transfer for backup

```
# Add new image transfer for backup.

transfer = transfers_service.add(
    ImageTransfer(
        disk=Disk(id=disk.id),
        direction=ImageTransferDirection.DOWNLOAD,
        backup=Backup(id=backup.id),
        format=DiskFormat.RAW,
```

Transfer status

```
# Wait until the transfer is ready.

while transfer.phase != \
      ImageTransferPhase.INITIALIZING:
   time.sleep(1)
   transfer = transfer_service.get()
```

Transfer URL

```
# Use the transfer URL to download the backup.

transfer.transfer_url
```

Download backup

Download the backup

(more on this later)

Finalizing transfer

```
# After downloading all data, finalize
# the transfer.
transfer.finalize()

# After transfering all disks, finalize
# the backup.
backup.finalize()
```

Incremental backup

How to start incremental backup?

Finding disks for incremental backup

```
# Fetch VM's disks and select disks enabled for
# incremental backup.

disks = [disk for disk in get_vm_disks()
         if disk.backup == DiskBackup.INCREMENTAL]
```

Starting incremental backup

```
# For starting an incremental backup, specify
# from_checkpoint_id. The backup will include
# data from all checkpoints since this checkpoint.

backup = backups_service.add(
    Backup(
        disks=disks,
        from_checkpoint_id=from_checkpoint_id,
```

Using imageio backup API

Imageio backup REST API

OPTIONS - request

```
OPTIONS /images/xxx HTTP/1.1
```

- First thing to do when connecting to imageio.
- Get features supported for specified resource.

OPTIONS - response

```
{
    "features": ["extents", "zero", "flush"],
    "unix_socket": "\0/org/ovirt/imageio",
}
```

- Can use the extents, zero and flush features
- Can switch to HTTP over Unix socket

Getting zero extents

How many GiB are in 100 GiB image?

Hint: we don't care about the zeroes.

Getting zero extents - request

```
GET /images/xxx/extents?context=zero HTTP/1.1
```

- Extent is a contiguous range of bytes on the storage.
- Fetch extents information from qemu NBD server for the entire image and returns a list of zero-extents.

Getting zero extents - response

```
[
  {"start": 0, "length": 1073676288, "zero": true},
  {"start": 1073676288, "length": 134217728, "zero": false},
  ...
]
```

- "zero": true - means this extent will read as zero, typically unallocated areas in the image. We can skip these extents during download.
- "zero": false - means this areas may contain data, typically allocated area. We need to download only these extents.

Getting dirty extents

Which blocks changed since the last backup?

We can tell by getting the dirty extents.

Getting dirty extents - request

```
GET /images/xxx/extents?context=dirty HTTP/1.1
```

- Fetch extents information from qemu NBD server for the entire image and returns list of dirty extents.
- Available only during incremental backup. If dirty extents are not available, will fail with HTTP error "404 Not Found".

Getting dirty extents - response

```
[
  {"start": 0, "length": 64536, "dirty": true},
  {"start": 64536, "length": 1073676288, "dirty": false},
  ...
]
```

- "dirty": true - means some bytes within this extent have been changed since the last backup.
- "dirty: false - means no byte has been changed since the last backup.

Getting data

How do we get data extents?

Use HTTP Range request.

Downloading extents

```
GET /images/xxx-yyy HTTP/1.1
Range: bytes=0-134217727
```

- Get data for extent starting at 0, with length 134217728.
- You need to send one request per extent.

Uploading data extents

How can we speed up restore?

Upload only the data extents.

Uploading data extents

```
PUT /images/xxx-yyy?flush=n HTTP/1.1
Content-Range: bytes 0-134217727/*
Content-Length: 134217728
```

- Uploads 134217728 message body bytes at offset 0 in the image.
- Specify flush=n to avoid flush (fsync) on every request.

Zeroing an extent on storage

What about the zero extents?

Use the "zero" feature.

Zeroing an extent on storage

```
PATCH /images/xxx-yyy HTTP/1.1

{
    "op": "zero",
    "offset": 4096,
    "size": 8192,
    "flush": false
}
```

- Zeros an extent of size 8192 bytes starting at offset 4096, without sending the actual zeroes over the wire.

Flushing

Is my data on storage?

Maybe, you must flush before you finalize the transfer.

Flushing uploaded data

```
PATCH /images/xxx-yyy HTTP/1.1

{
   "op": "flush"
}
```

- Flushes the data written to the image to the underlying storage.
- The call returns only when the device reports that the flush was done.

Imageio backup REST API

Too complicated?

We have a better solution!

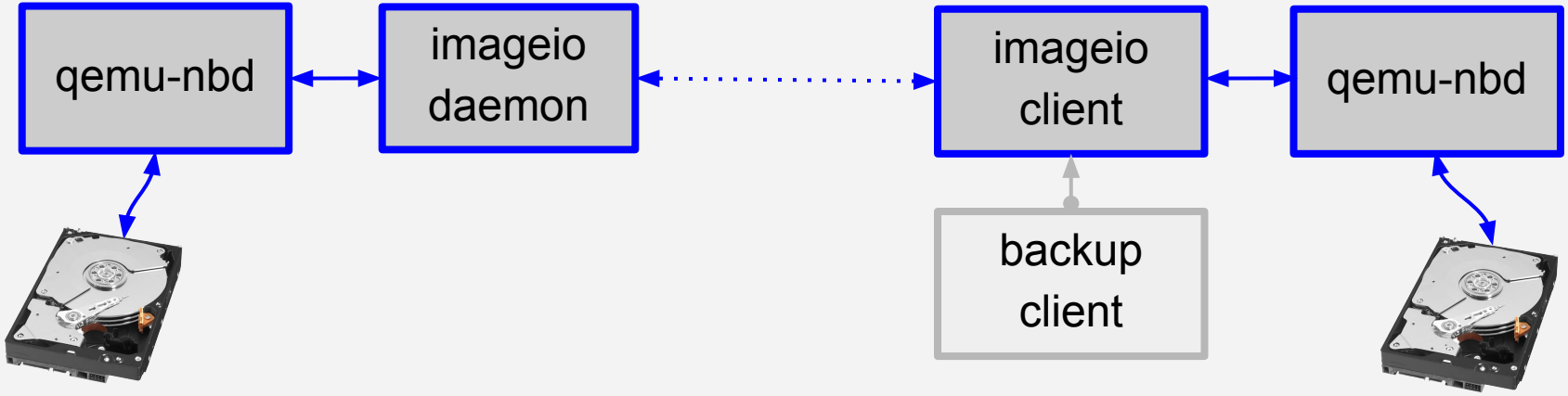
imageio client library

imageio client library

Write a time machine - the easy way

- Reference implementation for using imageio REST API.
- Build your own backup solution.
- For testing the backup APIs.

imageio client library - pipeline



imageio client library can download or upload data during backup and restore using qemu-nbd.

imageio client library - installation

```
# Package installed by default on oVirt hypervisors
# Can be installed anywhere if needed.

$ dnf install ovirt-imageio-client
```

imageio client library - full backup

```
from ovirt_imageio import client

# Downloads data extents, skipping zero extents.
# Using unix socket (if possible). Storing the
# data in qcow2 or raw formats.

client.download(
    "https://host:54322/images/xxx",
    "full-backup-2020-01-12.qcow2")
```

imageio client library - incremental backup

```
from ovirt_imageio import client

# Downloads dirty extents, using unix socket if
# possible, storing the data in qcow2 format.

client.download(
    "https://host:54322/images/xxx",
    "incr-backup-2020-01-13.qcow2",
    incremental=True)
```

imageio client library - prepare for restore

```
# Rebase every incremental backup image on the
# previous backup image, down to the last full
# backup image.

$ qemu-img rebase -u incr-backup-2020-01-14.qcow2 \
    -b incr-backup-2020-01-13.qcow2 -F qcow2

$ qemu-img rebase -u incr-backup-2020-01-13.qcow2 \
    -b full-backup-2020-01-12.qcow2 -F qcow2
```

imageio client library - restore

```
from ovirt_imageio import client

# Upload data and zero extents from specified image
# including data from entire qcow2 chain.

client.upload(
    "incr-backup-2020-01-14.qcow2"
    "https://host:54322/images/xxx")
```

Demo time - https://youtu.be/E2VWUVcycj4

Incremental backup nuts and bolts

Incremental backup nuts and bolts

Starting full backup

```
$ python3 ./backup_vm.py start \
    --engine-url https://engine3 \
    --username admin@internal \
    --password-file password \
    --cafile engine3.pem \
    614d...07
```

Starting full backup - engine log

```
2020-05-24 09:52:17,534+03 INFO
[org.ovirt.engine.core.bll.StartVmBackupCommand] (default task-1)
[825...ff] Creating VmBackup entity for '614d...07'

2020-05-24 09:52:17,548+03 INFO
[org.ovirt.engine.core.bll.StartVmBackupCommand] (default task-1)
[825b...ff] Creating VmCheckpoint entity for VM '614d...07'
```

Starting full backup - engine log

```
2020-05-24 09:52:18,455+03 INFO
[org.ovirt.engine.core.vdsbroker.vdsbroker.
StartVmBackupVDSCommand]
(EE-ManagedScheduledExecutorService-engineScheduledThreadPool-
Thread-88) [825b...ff] START, StartVmBackupVDSCommand(HostName =
10.35.1.33, VmBackupVDSParameters:{hostId='1390...03',
backupId='e1e3...01'}), log id: 3d2d5849
```

Starting full backup - database state

```
select * from vm_backups;
```

backup_id	from_checkpoint_id	to_checkpoint_id	vm_id	phase	_create_date
e1e3...01	[NULL]	**[NULL]**	614d...07	**Initializing**	2020-05-24 09:52:17

Starting full backup - database state

```
select * from vm_backup_disks_map;
```

backup_id	disk_id	backup_url
e1e3...01	9b0a...19	[NULL]
e1e3...01	6d96...ab	[NULL]

Starting full backup - database state

```
select * from vm_checkpoints;
```

checkpoint_id	parent_id	vm_id	_create_date	checkpoint_xml
23af...3b	[NULL]	614d...07	2020-05-24 09:52:17	[NULL]

Starting full backup - database state

```
select * from vm_checkpoints_disks_map;
```

checkpoint_id	disk_id
23af...3b	9b0a...19
23af...3b	6d96...ab

Starting full backup - VDSM log

```
2020-05-24 09:52:18,480+0300 INFO  (jsonrpc/6) [api.virt] START
start_backup(config={'backup_id': 'e1e3...01', 'disks':
[{'checkpoint': True, 'imageID': '9b0a...19', 'volumeID':
'64e2...ba', 'domainID': '2a52...cc'}, {'checkpoint': True
, 'imageID': '6d96...ab', 'volumeID': 'eae4...02', 'domainID':
'2a52...cc'}], 'from_checkpoint_id': None, 'to_checkpoint_id':
'23af...3b'}) from=::ffff:10.35.206.71,33532, flow_id=825b...ff,
vmId=614d...07 (api:48)
```

Starting full backup - VDSM log - config object

```
Config={
    'backup_id': 'e1e3...01',
    'disks': [
        {
            'checkpoint': True,
            'imageID': '...','volumeID': '...', 'domainID': '...'
        },
        {...}
    ],
    'from_checkpoint_id': None,
    'to_checkpoint_id': '23af...3b'
})
```

Starting full backup - VDSM log

```
2020-05-24 09:52:18,877+0300 INFO  (jsonrpc/6) [api.virt] FINISH
start_backup return={'result': {'disks': {'9b0a...19':
'nbd:unix:/var/run/vdsm/backup/e1e3...01:exportname=vda',
'6d96...ab':
'nbd:unix:/var/run/vdsm/backup/e1e3...ad01:exportname=sda'},
'checkpoint': '<domaincheckpoint> … </domaincheckpoint>,
'status': {'code': 0, 'message': 'Done'}}
from=::ffff:10.35.206.71,33532, flow_id=825b...ff, vmId=614d...07
(api:54)
```

Starting full backup - VDSM log - result object

```
result={
    'disks':
        {
            '9b0a...19': 'nbd:unix:/.../e1e3...01:exportname=vda',
            '6d96...ab': 'nbd:unix:/.../e1e3...ad01:exportname=sda'
        },
    'checkpoint': '<domaincheckpoint> … </domaincheckpoint>,,
    'status':
        {
            'code': 0,
            'message': 'Done'
        }}
```

Starting full backup - host scratch disks

```
$ tree /var/lib/vdsm/storage/transient_disks/

/var/lib/vdsm/storage/transient_disks/
└── 614de3b6-48ea-4ba7-914d-0b7397779d07
    ├── e1e3248f-a91c-48d1-be29-cbb9f8e0ad01.sda
    └── e1e3248f-a91c-48d1-be29-cbb9f8e0ad01.vda
```

Starting full backup - host unix socket

```
$ tree /var/run/vdsm/backup/

/var/run/vdsm/backup/
└── e1e3248f-a91c-48d1-be29-cbb9f8e0ad01
```

Starting full backup - engine log

```
2020-05-24 09:53:18,455+05 INFO
[org.ovirt.engine.core.vdsbroker.vdsbroker.StartVmBackupVDSCommand] (EE-ManagedScheduledExecutorService-engineScheduledThreadPool-Thread-100) [02dace02-83b8-4a18-bb39-879b3bdadb8e] FINISH,
StartVmBackupVDSCommand, return:
VmBackupInfo: {status='Status [code=0, message=Done]'}
{9b0a...19=nbd:unix:/var/run/vdsm/backup/e1e3...01:exportname=vda
,...} <domaincheckpoint>...</domaincheckpoint>
```

Backup status - READY - engine log

```
2020-05-24 09:52:18,964+03 INFO
[org.ovirt.engine.core.bll.StartVmBackupCommand]
(EE-ManagedScheduledExecutorService-engineScheduledThreadPool-
Thread-88) [825b...ff] Ready to start image transfers using
backup URLs
```

Backup status - READY - database state

```
select * from vm_backups;
```

backup_id	from_checkpoint_id	to_checkpoint_id	vm_id	phase	_create_date
e1e3...01	[NULL]	**23af...3b**	614d...07	**Ready**	2020-05-24 09:52:17

Backup status - READY - database state

```
select * from vm_backup_disks_map;
```

backup_id	disk_id	backup_url
e1e3...01	9b0a...19	nbd:unix:/var/run/vdsm/backup/e1e3...01:exportname=vda
e1e3...01	6d96...ab	nbd:unix:/var/run/vdsm/backup/e1e3...01:exportname=sda

Backup status - READY - database state

```
select * from vm_checkpoints;
```

checkpoint_id	parent_id	vm_id	_create_date	checkpoint_xml
23af...3b	[NULL]	614d...07	2020-05-24 09:52:17	<domaincheckpoint>...</domaincheckpoint>

Finalizing backup

```
$ python3 ./backup_vm.py stop \
    --engine-url https://engine3 \
    --username admin@internal \
    --password-file password \
    --cafile engine3.pem \
    614d...07
    e1e3...01
```

Finalizing backup - engine log

```
2020-05-24 10:34:03,121+03 INFO
[org.ovirt.engine.core.bll.StopVmBackupCommand] (default task-1)
[4051...51] Stopping VmBackup 'e1e3...01'

2020-05-24 10:34:03,124+03 INFO
[org.ovirt.engine.core.vdsbroker.vdsbroker.StopVmBackupVDSCommand
] (default task-1) [4051...51] START,
StopVmBackupVDSCommand(HostName = 10.35.1.33,
VmBackupVDSParameters:{hostId='1390...03',
backupId='e1e3...01'}), log id: 71aadb2c
```

Finalizing backup - engine log

```
2020-05-24 10:34:06,930+03 INFO
[org.ovirt.engine.core.dal.dbbroker.auditloghandling.
AuditLogDirector]
(EE-ManagedScheduledExecutorService-engineScheduledThreadPool-
Thread-49) [825b...ff] EVENT_ID: VM_BACKUP_SUCCEEDED(10,793), VM
vm2 backup has been completed successfully
(User:admin@internal-authz).
```

Finalizing backup - VDSM log

```
2020-05-24 10:34:03,141+0300 INFO  (jsonrpc/5) [api.virt] START
stop_backup(backup_id='e1e3...01')
from=::ffff:10.35.206.71,33532, flow_id=4051...51, vmId=614d...07
(api:48)

2020-05-24 10:34:03,146+0300 INFO  (jsonrpc/5) [api.virt] FINISH
stop_backup return={'status': {'code': 0, 'message': 'Done'}}
from=::ffff:10.35.206.71,33532, flow_id=4051...51, vmId=614d...07
(api:54)
```

Finalizing backup - host scratch disks

```
$ tree /var/lib/vdsm/storage/transient_disks/

/var/lib/vdsm/storage/transient_disks/

0 directories, 0 files
```

Finalizing backup - host unix socket

```
$ tree /var/run/vdsm/backup/

/var/run/vdsm/backup/

0 directories, 0 files
```

Finalizing full backup - database state

```
select * from vm_backups;
```

backup_id	from_checkpoint_id	to_checkpoint_id	vm_id	phase	_create_date

```
select * from vm_backup_disks_map;
```

backup_id	disk_id	backup_url

Finalizing full backup - database state

```
select * from vm_checkpoints;
```

checkpoint_id	parent_id	vm_id	_create_date	checkpoint_xml
23af...3b	[NULL]	614d...07	2020-05-24 09:52:17	<domaincheckpoint>...</domaincheckpoint>

Finalizing full backup - database state

```
select * from vm_checkpoint_disks_map;
```

checkpoint_id	disk_id
23af...3b	9b0a...19
23af...3b	6d96...ab

Starting incremental backup

```
$ python3 ./backup_vm.py start \
    --engine-url https://engine3 \
    --username admin@internal \
    --password-file password \
    --cafile engine3.pem \
    --from-checkpoint-uuid 23af...3b
    614d...07
```

Starting incremental backup - engine log

```
2020-05-24 13:19:20,108+03 INFO
[org.ovirt.engine.core.bll.StartVmBackupCommand] (default task-1)
[21ef...43] Creating VmBackup entity for VM '614d...07'

2020-05-24 13:19:20,115+03 INFO
[org.ovirt.engine.core.bll.StartVmBackupCommand] (default task-1)
[21ef...43] Redefine previous VM checkpoints for VM '614d...07'
```

Starting incremental backup - engine log

```
2020-05-24 13:19:20,119+03 INFO
[org.ovirt.engine.core.vdsbroker.vdsbroker.
ListVmCheckpointsVDSCommand] (default task-1) [21ef...43] START,
ListVmCheckpointsVDSCommand(HostName = 10.35.1.33,
VdsAndVmIDVDSParametersBase:{hostId='1390...03',
vmId='614d...07'}), log id: a1b7038

2020-05-24 13:19:20,171+03 INFO
[org.ovirt.engine.core.bll.StartVmBackupCommand] (default task-1)
[21ef...43] Checkpoints chain is already defined for VM
'614d...07'
```

Starting incremental backup - engine log

```
2020-05-24 13:19:20,171+03 INFO
[org.ovirt.engine.core.bll.StartVmBackupCommand] (default task-1)
[21ef...43] Successfully redefined previous VM checkpoints for VM
'614d...07

2020-05-24 13:19:20,171+03 INFO
[org.ovirt.engine.core.bll.StartVmBackupCommand] (default task-1)
[21ef...43] Creating VmCheckpoint entity for VM '614d...07'
```

Starting incremental backup - database state

```
select * from vm_backups;
```

backup_id	from_checkpoint_id	to_checkpoint_id	vm_id	phase	_create_date
fab4...cb	23af...3b	[NULL]	614d...07	Initializing	2020-05-24 13:19:20

Starting incremental backup - database state

```
select * from vm_backup_disks_map;
```

backup_id	disk_id	backup_url
fab4...cb	9b0a...19	[NULL]
fab4...cb	6d96...ab	[NULL]

Starting incremental backup - database state

```
select * from vm_checkpoints;
```

checkpoint_id	parent_id	vm_id	_create_date	checkpoint_xml
23af...3b	[NULL]	614d...07	2020-05-24 09:52:17	<domaincheckpoint>...</domaincheckpoint>
4a60...c2	**23af...3b**	614d...07	2020-05-24 13:19:20	**[NULL]**

Starting incremental backup - database state

```
select * from vm_checkpoint_disks_map;
```

checkpoint_id	disk_id
23af...3b	9b0a...19
23af...3b	6d96...ab
4a60...c2	9b0a...19
4a60...c2	9b0a...19

Starting incremental backup - VDSM log

```
2020-05-24 13:19:20,964+0300 INFO  (jsonrpc/6) [api.virt] START
start_backup(config={'backup_id': 'fab4...cb', 'disks':
[{'checkpoint': True, 'imageID': '9b0a...19', 'volumeID':
'64e2...ba', 'domainID': '2a52...cc'}, {'checkpoint': True
, 'imageID': '6d96...ab', 'volumeID': 'ea...a102', 'domainID':
'2a52...cc'}], 'from_checkpoint_id': '23af...3b',
'to_checkpoint_id': '4a60...c2'}) from=::ffff:10.35.206.71,33532,
flow_id=21ef...43, vmId=614d...07 (api:48)
```

Starting full backup - VDSM log - config object

```
Config={
    'backup_id': 'fab4...cb',
    'disks': [
        {
            'checkpoint': True,
            'imageID': '...','volumeID': '...', 'domainID': '...'
        },
        {...}
    ],
    'from_checkpoint_id': 23af...3b,
    'to_checkpoint_id': '4a60...c2'
})
```

Incremental backup status - READY - engine log

```
2020-05-24 13:19:21,507+03 INFO
[org.ovirt.engine.core.bll.StartVmBackupCommand]
(EE-ManagedScheduledExecutorService-engineScheduledThreadPool-
Thread-64) [21ef...43] Ready to start image transfers using
backup URLs
```

Incremental backup status - READY - database state

```
Select * from vm_backups;
```

backup_id	from_checkpoint_id	to_checkpoint_id	vm_id	phase	_create_date
fab4...cb	23af...3b	**4a60...c2**	614d...07	**Ready**	2020-05-24 13:19:20

Incremental backup status - READY - database state

```
select * from vm_backup_disks_map;
```

backup_id	disk_id	backup_url
fab4...cb	9b0a...19	nbd:unix:/var/run/vdsm/backup/fab4...cb:exportname=vda
fab4...cb	6d96...ab	nbd:unix:/var/run/vdsm/backup/fab4...cb:exportname=sda

Incremental backup status - READY - database state

```
select * from vm_checkpoints;
```

checkpoint_id	parent_id	vm_id	_create_date	checkpoint_xml
23af...3b	[NULL]	614d...07	2020-05-24 09:52:17	<domaincheckpoint>...</domaincheckpoint>
4a60...c2	23af...3b	614d...07	2020-05-24 13:19:20	**<domaincheckpoint>...</domaincheckpoint>**

Incremental backup data path

Running full backup example

```
$ python3 ./backup_vm.py full \
    --engine-url https://engine3 \
    --username admin@internal \
    --password-file password \
    --cafile engine3.pem \
    --backup-dir ./backup \
    2d554b69-3c03-4f06-84b8-f4a7dd3756d3
```

Backup progress 1

```
[   0.0 ] Starting full backup for VM 2d554b69-...
[   1.5 ] Waiting until backup 280d4790-... is ready
[   2.5 ] Created checkpoint 7fe8fe0d-...  (use as
--from-checkpoint-uuid in next incremental backup)
[   2.6 ] Creating image transfer for disk 9364f6e4-...
[   3.8 ] Waiting until transfer 3d317a84-... is ready
```

Backup progress 2

```
Formatting './backup/9364f6e4-....202006022238.full.qcow2',
fmt=qcow2 size=6442450944 cluster_size=65536
lazy_refcounts=off refcount_bits=16
[ 100.00% ] 6.00 GiB, 15.52 seconds, 395.75 MiB/s
[  19.3 ] Finalizing transfer 3d317a84-...
[  19.5 ] Full backup completed successfully
```

Downloaded image

```
$ qemu-img info ./backup/9364f6e4-....full.qcow2
image: ./backup/9364f6e4-....202006022238.full.qcow2
file format: qcow2
virtual size: 6 GiB (6442450944 bytes)
disk size: 1.56 GiB
...
```

Diving into vdsm log

How to track backup logs?

Use transfer and ticket ids:

```
$ grep 3d317a84-9c18 /var/log/vdsm/vdsm.log
```

Adding ticket

```
2020-06-02 22:38:03,626+0300 INFO  (jsonrpc/5) [vdsm.api]
START add_image_ticket(ticket={'dirty': False, 'ops':
['read'], 'size': 6442450944, 'sparse': True, 'transfer_id':
'3d317a84-9c18-44a5-8c49-9ac4e6a85989', ...)
from=::ffff:192.168.122.12,43944,
flow_id=ccc24730-7ab3-4e7f-a594-41fc7e01a7cb,
task_id=ed7d35ee-edb4-4856-b4ed-cddcab73ce89 (api:48)
```

Monitoring transfer progress

```
2020-06-02 22:38:03,949+0300 INFO  (jsonrpc/3) [vdsm.api]
FINISH get_image_ticket return={'result': {'active': True,
'expires': 4295573, 'idle_time': 0, ... 'transfer_id':
'3d317a84-9c18-44a5-8c49-9ac4e6a85989', 'transferred':
23265280}} from=::ffff:192.168.122.12,43944,
flow_id=ccc24730-7ab3-4e7f-a594-41fc7e01a7cb,
task_id=d550c40e-4bb4-46af-bf60-cbad13c15969 (api:54)
```

Removing ticket

```
2020-06-02 22:38:28,257+0300 INFO  (jsonrpc/0) [vdsm.api]
START remove_image_ticket(
uuid='33d1c166-3b34-4b21-bbcd-7d10cb2e7a4c')
from=::ffff:192.168.122.12,43944,
flow_id=ccc24730-7ab3-4e7f-a594-41fc7e01a7cb,
task_id=cb1aed2c-7028-4e63-8386-0c20e81d3b6b (api:48)
```

Diving into imageio log

How to track backup logs?

Use transfer and ticket ids

```
$ grep 3d317a84-... /var/log/ovirt-imageio/daemon.log
```

Adding ticket

```
2020-06-02 22:38:03,628 INFO    (Thread-1) [tickets] [local]
ADD ticket={'dirty': False, 'ops': ['read'], 'size':
6442450944, 'sparse': True, 'transfer_id':
'3d317a84-9c18-44a5-8c49-9ac4e6a85989', 'uuid':
'33d1c166-3b34-4b21-bbcd-7d10cb2e7a4c', 'timeout': 300,
'url': 'nbd:unix:/var/run/vdsm/backup/...:exportname=sda'}
```

Tickets details - dirty

```
'dirty': False
```

```
This ticket does not support reporting dirty extents
```

Tickets details - url

```
'url':
'nbd:unix:/var/run/vdsm/backup/280d4
790...cbc6cafc21db:exportname=sda'
```

This ticket uses nbd backend to provide data for disk "sda"

Tickets details - uuid

```
'uuid': '33d1c166...7d10cb2e7a4c'
```

Use the ticket uuid to follow ticket lifetime.

Client connecting to imageio

```
2020-06-02 22:38:03,687 INFO    (Thread-2) [http] OPEN
client=192.168.122.1
```

No much info, need to lookup the next logs

Diving into imageio logs

How to track connection logs?

Use the thread name

```
$ grep (Thread-2) /var/log/ovirt-imageio/daemon.log
```

OPTIONS request

```
2020-06-02 22:38:03,688 INFO    (Thread-2) [images]
[192.168.122.1] OPTIONS ticket=33d1c166-...-7d10cb2e7a4c
```

First thing good client will do is send OPTIONS request to learn about server capabilities.

EXTENTS request

```
2020-06-02 22:38:03,688 INFO    (Thread-2) [extents]
[192.168.122.1] EXTENTS ticket=33d1c166-...-7d10cb2e7a4c
context=zero
```

Get list of data and zero extents for this image.

Open backend

```
2020-06-02 22:38:03,690 INFO    (Thread-2) [backends.nbd]
Open backend
address='/var/run/vdsm/backup/280d4790-...-cbc6cafc21db'
export_name='sda'
```

For getting extents we need to connect to QEMU NBD server.

Client reads data...

(No logs)

Logging all requests is too noisy and slow

READ requests logged only in DEBUG level

Client finished

```
2020-06-02 22:38:19,208 INFO    (Thread-2) [http] CLOSE
client=192.168.122.1 [connection 1 ops, 15.520613 s]
[dispatch 108 ops, 11.482541 s] [extents 1 ops, 0.002150 s]
[operation 106 ops, 11.458472 s] [read 276 ops, 2.411138 s,
1.56 GiB, 724.17 MiB/s] [write 276 ops, 8.987284 s, 1.56 GiB,
178.15 MiB/s]
```

Transfer stats

```
connection 1 ops, 15.520613 s
dispatch 108 ops, 11.482541 s
extents 1 ops, 0.002150 s
read 276 ops, 2.411138 s, 1.56 GiB, 724.17 MiB/s
write 276 ops, 8.987284 s, 1.56 GiB, 178.15 MiB/s
```

Full backup is fast

Disk size: 6 GiB

Transferred: 1.56 GiB

Backup of raw preallocated disk is much slower

Closing backend

```
2020-06-02 22:38:19,208 INFO    (Thread-2) [backends.nbd]
Close backend
address='/var/run/vdsm/backup/280d4790-a6f7-40c9-8665-cbc6caf
c21db'
```

Backend is owned by the client connection,
closed when connection is closed.

Remove ticket

```
2020-06-02 22:38:28,258 INFO    (Thread-7) [tickets] [local]
REMOVE ticket=33d1c166-3b34-4b21-bbcd-7d10cb2e7a4c
```

Vdsm remove the ticket when the backup is done.

Running incremental backup example

```
$ python3 ./backup_vm.py incremental \
    --engine-url https://engine3 \
    --username admin@internal \
    --password-file password \
    --cafile engine3.pem \
    --backup-dir ./backup \
    --from-checkpoint-uuid 7fe8fe0d-...-e68539c0de0b \
    2d554b69-...-f4a7dd3756d3
```

Incremental backup progress 1

```
[   0.0 ] Starting incremental backup for VM 2d554b69-...
[   1.2 ] Waiting until backup e9da81c3-... is ready
[   2.2 ] Created checkpoint '8a784000-...' (use as
--from-checkpoint-uuid in next incremental backup)
[   2.3 ] Creating image transfer for disk 9364f6e4-...
[   3.4 ] Waiting until transfer f7ed2b83-... is ready
```

Incremental backup progress 2

```
Formatting './backup/9364f6e4-....incremental.qcow2',
fmt=qcow2 size=6442450944 cluster_size=65536
lazy_refcounts=off refcount_bits=16
[ 100.00% ] 6.00 GiB, 0.34 seconds, 17.86 GiB/s
[   3.8 ] Finalizing transfer f7ed2b83-...
[   3.9 ] Incremental backup completed successfully
```

Downloaded image

```
$ qemu-img info ./backup/9364f6e4-...incremental.qcow2
image: ./backup/9364f6e4-...202006022242.incremental.qcow2
file format: qcow2
virtual size: 6 GiB (6442450944 bytes)
disk size: 2.5 MiB
...
```

Diving into vdsm log

How to track incremental backup logs?

Use transfer id

```
$ grep f7ed2b83-... /var/log/vdsm/vdsm.log
```

Adding ticket

```
2020-06-02 22:42:03,763+0300 INFO  (jsonrpc/0) [vdsm.api]
START add_image_ticket(ticket={'dirty': True, 'ops':
['read'], 'size': 6442450944, 'sparse': True, 'transfer_id':
'f7ed2b83-7ae2-4c0b-b353-8b8aec08cd89', 'uuid':
'fbb56f9b-8bc6-482d-8a4f-749ea3ffb997', ...
```

Monitoring progress

(No logs in this examples)

This backup was too fast, no progress was monitored.

Removing ticket

```
2020-06-02 22:42:04,589+0300 INFO  (jsonrpc/7) [vdsm.api]
START remove_image_ticket
(uuid='fbb56f9b-8bc6-482d-8a4f-749ea3ffb997')
from=::ffff:192.168.122.12,43944, flow_id=28
```

Diving into imageio logs

How to track incremental backup log?

Use transfer and ticket ids

```
$ grep f7ed2b83-... /var/log/ovirt-imageio/daemon.log
```

How to track incremental backup in imageio log?

```
2020-06-02 22:42:03,765 INFO    (Thread-8) [tickets] [local]
ADD ticket={'dirty': True, 'ops': ['read'], 'size':
6442450944, 'sparse': True, 'transfer_id':
'f7ed2b83-7ae2-4c0b-b353-8b8aec08cd89', 'uuid':
'fbb56f9b-8bc6-482d-8a4f-749ea3ffb997', 'timeout': 300,
'url': 'nbd:unix:/var/run/vdsm/backup/...:exportname=sda'}
```

Tickets details - dirty

```
'dirty': True
```

This ticket supports reporting dirty extents;
areas on storage modified since last backup.

OPTIONS request

```
2020-06-02 22:42:03,811 INFO    (Thread-9) [images]
[192.168.122.1] OPTIONS
ticket=fbb56f9b-8bc6-482d-8a4f-749ea3ffb997
```

This a good client

EXTENTS zero request

```
2020-06-02 22:42:03,813 INFO    (Thread-9) [extents]
[192.168.122.1] EXTENTS
ticket=fbb56f9b-8bc6-482d-8a4f-749ea3ffb997 context=zero
```

Client request image extents to get image size.

EXTENTS dirty request

```
2020-06-02 22:42:03,877 INFO    (Thread-9) [extents]
[192.168.122.1] EXTENTS
ticket=fbb56f9b-8bc6-482d-8a4f-749ea3ffb997 context=dirty
```

Client requests image dirty extents so it can download only the areas modified since last backup.

Client reads dirty extents...

(No logs)

Logging all requests is too noisy and slow

READ requests logged only in DEBUG level

Client finished

```
2020-06-02 22:42:04,143 INFO    (Thread-9) [http] CLOSE
client=192.168.122.1 [connection 1 ops, 0.331756 s] [dispatch
34 ops, 0.050843 s] [extents 2 ops, 0.002295 s] [operation 31
ops, 0.040774 s] [read 31 ops, 0.030551 s, 2.06 MiB, 67.51
MiB/s] [write 31 ops, 0.007008 s, 2.06 MiB, 294.32 MiB/s]
```

Transfer stats

```
connection 1 ops, 0.331756 s
dispatch 34 ops, 0.050843 s
extents 2 ops, 0.002295 s
read 31 ops, 0.030551 s, 2.06 MiB, 67.51 MiB/s
write 31 ops, 0.007008 s, 2.06 MiB, 294.32 MiB/s
```

Incremental backup is fast

Disk size: 6 GiB

Transferred: 2.06 MiB

Depends on the amount of data changed since the last backup.

Remove ticket

```
2020-06-02 22:42:04,589 INFO    (Thread-10) [tickets] [local]
REMOVE ticket=fbb56f9b-8bc6-482d-8a4f-749ea3ffb997
```

Quiz: who sent this request?

Incremental backup troubleshooting

Starting backup validation failures

- VM is not running - wait until VM is up
- Disk is not enabled for incremental backup - enable disk for backup
- Requested from_checkpoint_id not available - perform full backup
- New disk added to VM or backup - perform full backup
- Disks locked - retry later
- Backup already running for this VM
- isIncrementalBackupSupported engine config not set

Backup operation failed

- Redefining checkpoints failed - perform full backup
- Transfer failed - retry transfer or stop backup
- Finalize backup failed - retry
- Internal errors (engine, VDSM, libvirt, qemu) - need investigation by support

Libvirt/QEMU known issues

- Live migrate VM with backups
  - https://bugzilla.redhat.com/1799011 (RHEL 8.2) not tested yet
- Full backup required after storage migration
  - https://bugzilla.redhat.com/1779893 (RHEL 8.2.1)
- Full backup needed after cold snapshot
  - https://bugzilla.redhat.com/1804593 (RHEL 8.2.1)
- Full backup needed after adding disks
  - https://bugzilla.redhat.com/1829829 (RHEL 8.3)

RHV known issues

- Full backup needed after cold merge
- Cannot backup non-running VM
- Incremental backup not supported for raw images
- Backup not supported for direct LUN or Managed Block Storage
- Documentation missing or needs updates
- Backup events need example code

More info

- ovirt-imageio random I/O API docs
- ovirt-image documentation
- ovirt-imageio client
- Full backup examples
- ovirt-engine-sdk examples
- oVirt Incremental backup feature page
- qemu incremental backup feature page
- Domain state capture using Libvirt
- ovirt.org site

@ovirt https://ovirt.org users@ovirt.org

@nirsof
www.linkedin.com/in/nirsof
nirs

@danielerez
www.linkedin.com/in/danielerez
danielerez

@shenitzky
www.linkedin.com/in/eshenitzky
shenitzky