Hands-on voorbeelden

We hopen met onderstaande hands-on voorbeelden wat inzicht te geven in hoe het gebruik van snapshots effect hebben op beschikbaar 'quota' van de afgenomen netwerkback-updienst.

Hiervoor gaan we op de Linux command line aan de tiep met ZFS!

In grote lijnen is dit hieronder wat onze back-up ook doet. We beginnen met het maken van een eigen filesysteem voor de back-up, stellen daar een quota op in en om het verwarrende aspect "compressie" even buiten beschouwing te laten schakelen we compressie UIT, in tegenstelling tot wat we normaal doen (lz4 compressie):

```
# zfs create backup/testpool
# zfs set quota=100MB backup/testpool
# zfs set compression=off backup/testpool
```

We kijken elke stap even naar de getallen om een beter beeld te krijgen van wat de relevante waardes betekenen:

```
# zfs get -r used,available,referenced,usedbydataset,usedbysnapshots backup/testpool
NAME                PROPERTY          VALUE      SOURCE
backup/testpool     used              19K        -
backup/testpool     available         100M       -
backup/testpool     referenced        19K        -
backup/testpool     usedbydataset     19K        -
backup/testpool     usedbysnapshots   0          -
```

Zoals we zien is er direct bij het aanmaken van het filesysteem al 19K in gebruik. Dit is "metadata" binnen het filesysteem en we hebben 100MB beschikbare ruimte, zoals het ingestelde quota.

Nu maken we binnen de back-up een 50MB grote file, stel dit is de 1e back-up die we maken:

```
# dd if=/dev/urandom of=50MB.bin bs=1048576 count=50
50+0 records in
50+0 records out
52428800 bytes (52 MB) copied, 6.23532 s, 8.4 MB/s
```

We kijken weer naar de getallen:

```
# zfs get -r used,available,referenced,usedbydataset,usedbysnapshots backup/testpool
NAME                PROPERTY          VALUE      SOURCE
backup/testpool     used              50.1M      -
backup/testpool     available         49.9M      -
backup/testpool     referenced        50.1M      -
backup/testpool     usedbydataset     50.1M      -
backup/testpool     usedbysnapshots   0          -
```

We zien nu duidelijk dat de used en referenced van de pool zelf naar 50MB zijn gesprongen. Er zijn geen snapshots, dus usedbysnapshots is ook 0.

Nu gaan we een nieuwe "back-up maken", en net als met onze netwerkback-updienst maken we dan eerst een snapshot voor we de nieuwe data naar het filesysteem schrijven:

```
# zfs snapshot backup/testpool@snapA
```

En de getallen:

```
# zfs get -r used,available,referenced,usedbydataset,usedbysnapshots backup/testpool
NAME                  PROPERTY          VALUE         SOURCE
backup/testpool       used              50.1M         -
backup/testpool       available         49.9M         -
backup/testpool       referenced        50.1M         -
backup/testpool       usedbydataset     50.1M         -
backup/testpool       usedbysnapshots   0             -
backup/testpool@snapA used              0             -
backup/testpool@snapA referenced        50.1M         -
```

Hier zien we dat er een snapshot is gemaakt. Het snapshot gebruikt zelf 0 bytes omdat er nog niets is gewijzigd tussen het filesysteem en het snapshot. Het snapshot "refereert" aan 50.1M data, dezelfde data die ook in het filesysteem zelf zit. We kunnen nu dus binnen dit snapshot naar dezelfde data die ook in het filesysteem zit, zonder dat dit snapshot tweemaal zoveel ruimte inneemt.

Nu maken we een nieuwe back-up. We stellen dat er 25MB aan data is veranderd op de te back-uppen machine, dus we wijzigen de eerste 25MB van de 50MB grote file die we in het begin als 'back-up' hebben gemaakt:

```
# dd conv=notrunc if=/dev/urandom of=50MB.bin bs=1048576 count=25
25+0 records in
25+0 records out
26214400 bytes (26 MB) copied, 4.09956 s, 6.4 MB/s
```

En we kijken meteen naar de getallen:

```
# zfs get -r used,available,referenced,usedbydataset,usedbysnapshots backup/testpool
NAME                  PROPERTY          VALUE         SOURCE
backup/testpool       used              75.1M         -
backup/testpool       available         24.9M         -
backup/testpool       referenced        50.1M         -
backup/testpool       usedbydataset     50.1M         -
backup/testpool       usedbysnapshots   25.0M         -
backup/testpool@snapA used              25.0M         -
backup/testpool@snapA referenced        50.1M         -
```

Er vallen hier een aantal dingen op:

* 'used' (van het fileststeem) is dus een totaal van de data in de laatst gemaakte back-up PLUS de GEWIJZIGDE data in de snapshot(s) en staat nu op 75MB. De 50MB van de eerst gemaakte back-up PLUS de 25MB gewijzigde data van de tweede back-up.

* 'referenced' van het snapshot blijft '50.1MB', want het filesysteem op het moment dat dat snapshot gemaakt werd, bevatte 50.1MB aan data. Maar 'used' van dit snapshot is ineens naar 25MB gesprongen. De 25MB gewijzigde data is middels het 'Copy On Write'-principe in het snapshot verschoven.

* De 'available' space van het filesysteem is plots nog maar 25MB, want zowel de wijzigingen in de snapshots als de "live data" gaan van deze beschikbare ruimte af.

We maken nog een back-up, en beginnen dus weer met een snapshot:

```
# zfs snapshot backup/testpool@snapB
```

En meteen kijken we weer naar de nummertjes:

```
# zfs get -r used,available,referenced,usedbydataset,usedbysnapshots backup/testpool
NAME                  PROPERTY          VALUE         SOURCE
backup/testpool       used              75.1M         -
backup/testpool       available         24.9M         -
backup/testpool       referenced        50.1M         -
backup/testpool       usedbydataset     50.1M         -
backup/testpool       usedbysnapshots   25.0M         -
backup/testpool@snapA used              25.0M         -
backup/testpool@snapA referenced        50.1M         -
backup/testpool@snapB used              0             -
backup/testpool@snapB referenced        50.1M         -
```

Zoals verwacht, used van dit nieuwe snapshot blijft 0 en de used van het eerste snapshot blijft 25MB. Nu maken we de feitelijke back-up en wijzigen wederom 25MB aan data:

```
# dd conv=notrunc if=/dev/urandom of=50MB.bin bs=1048576 count=25
25+0 records in
25+0 records out
26214400 bytes (26 MB) copied, 3.34764 s, 7.8 MB/s
```

Meteen duiken we weer in de cijfetjes:

```
# zfs get -r used,available,referenced,usedbydataset,usedbysnapshots backup/testpool
NAME                  PROPERTY          VALUE         SOURCE
backup/testpool       used              100M          -
backup/testpool       available         7K            -
backup/testpool       referenced        50.1M         -
backup/testpool       usedbydataset     50.1M         -
backup/testpool       usedbysnapshots   49.9M         -
backup/testpool@snapA used              25.0M         -
backup/testpool@snapA referenced        50.1M         -
backup/testpool@snapB used              24.9M         -
backup/testpool@snapB referenced        50.1M         -
```

Het eerste snapshot blijft ongewijzigd op 25MB used en 50MB referenced staan. Het tweede snapshot groeit, geheel volgens verwachting, met 25MB, de data die we voor deze tweede back-up hebben gewijzigd.

Voor de derde back-up, je voelt 'm al aankomen, beginnen we met een snapshot!

```
# zfs snapshot backup/testpool@snapC
```

En de getalletjes, al zou je deze nu moeten kunnen raden, voor de volledigheid:

```
# zfs get -r used,available,referenced,usedbydataset,usedbysnapshots backup/testpool
NAME                  PROPERTY         VALUE        SOURCE
backup/testpool       used             100M         -
backup/testpool       available        0            -
backup/testpool       referenced       50.1M        -
backup/testpool       usedbydataset    50.1M        -
backup/testpool       usedbysnapshots  50.1M        -
backup/testpool@snapA used             25.0M        -
backup/testpool@snapA referenced       50.1M        -
backup/testpool@snapB used             25.0M        -
backup/testpool@snapB referenced       50.1M        -
backup/testpool@snapC used             0            -
backup/testpool@snapC referenced       50.1M        -
```

Echter, nu is er op de denkbeeldige te back-uppen machine wat gebeurd. De 50MB file is vervangen door een geheel nieuw 10MB exemplaar. De hele 50MB aan data zal dus 'veranderen' waar eerder de eerste 25MB veranderde en de laatste 25MB gelijk bleef.

```
# rm 50MB.bin
rm: cannot remove ‘50MB.bin’: Disk quota exceeded
```

Dat is grappig. We kunnen geen bestanden verwijderen omdat de disk vol zit? Maar dit is natuurlijk logisch!

De snapshot(s) 'refereren' nog aan de data in deze file! Bij het verwijderen van de gehele file uit het filesysteem, zal die data moeten verplaatsen naar de snapshot(s) zelf. Het Copy On Write principe. Wijzigt er data? Dan gaat het pas (extra) ruimte in gebruik nemen.

Een kijkje in de getalletjes hierboven bevestigt dat er 0 bytes 'available' zijn. Zowel 'usedbydataset' als 'usedbysnapshots' is 50.1MB, totaal 100MB, geen ruimte meer om de 25MB aan veranderde data op te slaan.

We zouden nu een snapshot kunnen weggooien om de 'used'-hoeveelheid aan ruimte vrij te krijgen, maar dan kunnen we ook nooit meer naar die staat van het filesysteem terug. We kiezen er dus voor het quota te verhogen.

```
# zfs set quota=1024MB backup/testpool
# zfs get -r used,available,referenced,usedbydataset,usedbysnapshots backup/testpool
NAME                  PROPERTY         VALUE        SOURCE
backup/testpool       used             100M         -
backup/testpool       available        924M         -
backup/testpool       referenced       19K          -
```

```
backup/testpool          usedbydataset       19K             -
backup/testpool          usedbysnapshots     100M            -
backup/testpool@snapA    used                25.0M           -
backup/testpool@snapA    referenced          50.1M           -
backup/testpool@snapB    used                25.0M           -
backup/testpool@snapB    referenced          50.1M           -
backup/testpool@snapC    used                25.0M           -
backup/testpool@snapC    referenced          50.1M           -
```

Nu kunnen we de file weghalen, en een nieuwe file van 10MB maken:

```
# rm 50MB.bin
# dd if=/dev/urandom of=10MB.bin bs=1048576 count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB) copied, 0.802496 s, 13.1 MB/s
```

We duiken wederom de getalletjes in:

```
# zfs get -r used,available,referenced,usedbydataset,usedbysnapshots backup/testpool
NAME                     PROPERTY            VALUE           SOURCE
backup/testpool          used                110M            -
backup/testpool          available           914M            -
backup/testpool          referenced          10.0M           -
backup/testpool          usedbydataset       10.0M           -
backup/testpool          usedbysnapshots     100M            -
backup/testpool@snapA    used                25.0M           -
backup/testpool@snapA    referenced          50.1M           -
backup/testpool@snapB    used                25.0M           -
backup/testpool@snapB    referenced          50.1M           -
backup/testpool@snapC    used                25.0M           -
backup/testpool@snapC    referenced          50.1M           -
```

Eerder schreven we over de optie om snapshots weg te halen en 'used' ruimte vrij te krijgen, maar het verbruik van een snapshot verschuift. Gegevens die tussen snapA, snapB en snapC gelijk zijn, maar niet meer in het filesysteem zelf staan, verschuiven als een van deze snapshots opgeruimd wordt.

We verwijderen 'snapA', en krijgen (used) 25M aan ruimte terug.
914M available + 25M is 939M available:

```
# zfs destroy backup/testpool@snapA

# zfs get -r used,available,referenced,usedbydataset,usedbysnapshots backup/testpool
NAME                     PROPERTY            VALUE           SOURCE
backup/testpool          used                85.1M           -
backup/testpool          available           939M            -
backup/testpool          referenced          10.0M           -
backup/testpool          usedbydataset       10.0M           -
backup/testpool          usedbysnapshots     75.1M           -
backup/testpool@snapB    used                25.0M           -
backup/testpool@snapB    referenced          50.1M           -
backup/testpool@snapC    used                25.0M           -
backup/testpool@snapC    referenced          50.1M           -
```

Nu verwijderen we ook snapB:

```
# zfs destroy backup/testpool@snapB

# zfs get -r used,available,referenced,usedbydataset,usedbysnapshots backup/testpool
NAME                  PROPERTY          VALUE      SOURCE
backup/testpool       used              60.1M      -
backup/testpool       available         964M       -
backup/testpool       referenced        10.0M      -
backup/testpool       usedbydataset     10.0M      -
backup/testpool       usedbysnapshots   50.0M      -
backup/testpool@snapC  used             50.0M      -
backup/testpool@snapC  referenced       50.1M      -
```

Merk hier nu op dat snapC ineens 50M 'used', terwijl dat met snapB er bij nog maar 25M was. snapB was het laatste snapshot waar de 25MB data die ongewijzigd is gebleven ten opzichte van snapA in zat opgeslagen. SnapC moet die data nu zelf huisvesten.

Oh! Bijna vergeten. Snapshots zijn "read only". Net als een foto. Het is dus niet mogelijk om die grote file uit snapshots te verwijderen en zo alsnog de ruimte vrij te krijgen.

Dit is waarom de snapshots die verder terug de tijd in gaan, uiteindelijk ook meer ruimte gebruiken dan de snapshots van de laatst gemaakte back-up. Het oudste snapshot zal de meeste gewijzigde bytes ten opzichte van het filesysteem moeten huisvesten.

Na het verschijnen van dit technische blogje zal de back-uptool van ons worden aangepast, waarmee er meer informatie aangaande het gebruik van uw back-up en snapshots wordt gemaild. Tevens slaan we nog meer gedetailleerde informatie op in de "back-up stamp"-file waar je als techneut nog meer inzicht kunt krijgen in uw back-upverbruik:

```
  test.example.tld
    Backup started: Wed Jul 19 10:32:04 2017
    Backup ended: Wed Jul 19 10:33:06 2017
    Backup duration: 0h 1m 2s
    Diskspace change: 30.0 MB
    Diskspace used: 3.3/20.0 GB
          By last backup: 1.6 GB
          By snapshots: 1.8 GB
  Excludes:
    /sys
    [ .. ]
  Available snapshots:
    2017-06-19-020126 (308M v.s. last backup)
    2017-06-26-032602 (232M v.s. previous snapshot)
    [ .. ]
    2017-07-19-015924 (83.8M v.s. previous snapshot)
```

En technische informatie in de "backup stamp"-file:

```
  # cat 000_BACKUP-DONE_2017-07-19_10-33-06
  Backup run was successful.
  Log:
```

Number of files: 114,729 (reg: 89,888, dir: 15,492, link: 9,317, special: 32)
Number of created files: 5 (reg: 5)
Number of deleted files: 1 (reg: 1)
Number of regular files transferred: 170
Total file size: 3,278,250,709 bytes
Total transferred file size: 353,719,680 bytes
Literal data: 27,764,999 bytes
Matched data: 325,954,681 bytes
File list size: 683,873
File list generation time: 0.001 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 615,753
Total bytes received: 30,771,087

sent 615,753 bytes received 30,771,087 bytes 689,820.66 bytes/sec
total size is 3,278,250,709 speedup is 104.45

rsync 17264 for text.example.tld exit value 0 (0)

ZFS stats:

text.example.tld used 3.32G
text.example.tld available 66.7G
text.example.tld referenced 1.56G
text.example.tld compressratio 2.23x
text.example.tld quota 70G
text.example.tld reservation 20G
text.example.tld usedbysnapshots 1.76G
text.example.tld usedbydataset 1.56G
text.example.tld refcompressratio 2.09x
text.example.tld logicalused 7.36G
text.example.tld logicalreferenced 3.23G

text.example.tld@2017-06-19-020126 used 308M
text.example.tld@2017-06-19-020126 referenced 1.54G
text.example.tld@2017-06-19-020126 compressratio 2.11x
text.example.tld@2017-06-19-020126 refcompressratio 2.11x
text.example.tld@2017-06-19-020126 logicalreferenced 3.23G

text.example.tld@2017-06-26-032602 used 232M
text.example.tld@2017-06-26-032602 referenced 1.51G
text.example.tld@2017-06-26-032602 compressratio 2.07x
text.example.tld@2017-06-26-032602 refcompressratio 2.07x
text.example.tld@2017-06-26-032602 logicalreferenced 3.10G

[..]

text.example.tld@2017-07-19-103205 used 29.1M
text.example.tld@2017-07-19-103205 referenced 1.56G
text.example.tld@2017-07-19-103205 compressratio 2.09x
text.example.tld@2017-07-19-103205 refcompressratio 2.09x
text.example.tld@2017-07-19-103205 logicalreferenced 3.24G

Oplettende lezers merken op dat we 'reservation' en 'quota' omgedraaid gebruiken en dat het 'quota' voor elke back-up zelfs hoger staat dan de daadwerkelijk afgenomen (reserved) ruimte op het back-upplatform.

Nog wat meer techniek: Reserved space is echt vastgelegd voor deze machine, alles wat quota

daarboven gebruikt is 'gedeelde, niet toegewezen' ruimte. We hopen er zo voor te zorgen dat back-ups blijven werken als de back-up net aan de grens van het "quota" komt.

Tevens is in het begin van dit technische verhaal gesproken over het uitschakelen van compressie op het filesysteem. In de praktijk maken we wel gebruik van compressie (lz4) en de winst die je hiermee behaalt kan je terugzien in de 'logicalused' en 'compressratio'-waardes.

We hopen dat met dit blog de werking van onze netwerkback-up enigszins verduidelijkt is en dat het helderder is dat het simpelweg even opruimen van een groot bestand op een server niet meteen voor 'verlichting' zorgt in het ruimtegebruik van de back-ups. De verwijderde file zal immers ruimte blijven innemen tot alle snapshots die deze file 'refereren' zijn opgeruimd.