Azure Virtual Machine Plugin

Bacula Systems Documentation

Contents

Contents

1 Features Summary

- Snapshot-based online backup of any guest VM.
- Full, Incremental and Differential block level image backup
- Ability to restore complete virtual machine image.
- The Azure-VM plugin is compatible with Copy/Migration jobs. Please read the blb:MigrationChapter for more information.

2 Guest VM Backup Strategies

2.1 Installing Bacula Client on Each Guest

This strategy works by installing a Bacula Enterprise File Daemon on every virtual machine as if they were normal physical clients. In order to optimize the I/O usage on the Azure-VM hypervisor, the user will use Bacula's Schedules, Priorities, and Maximum Concurrent Jobs to spread backup jobs over the backup window. Since all VMs could use the same storage on the Microsoft Azure hypervisor, running all backup jobs at the same time could create a bottleneck on the disk/network subsystem since Bacula will walk through all filesystems to open/read/close/stat files.

Installing a Bacula Enterprise File Daemon on each virtual machine permits to manage virtual servers like physical servers and also to use all Bacula Enterprise's features such as:

- Quick restores of individual files.
- Checksum of individual files for Virus and Spyware detection.
- Verify Jobs.
- File/Directory exclusion (such as swap or temporary files).
- File level compression.
- Accurate backups.

2.2 Image Backup With Azure-Vm Plugin

With the image backup level strategy, the Bacula Enterprise Azure-VM Plugin will save the Client disks at the raw level, in the Azure context.

Bacula's Azure-VM plugin will read and save content of virtual machines disks using snapshots.

During backups, Azure plugin will save the integrity of disks images and also guest VM configurations to allow guest VM restores with their original parameters.

3 Backup and Restore Operations

3.1 Backup

The backup of a single guest VM takes the following steps:

- Export guest VM metadata configuration for future restore.
- Backup OS disk.
- Backup all data disks
- Wait for all procedures to finish
- Add new snapshot to a tracker and check for older snapshot to delete

3.2 Disk backup procedure

The same process is used to backup OS or data disk, it follows the following steps:

- Create a snapshot of the disk
- Issue an SAS token on the snapshot just created.
- Move snapshot data to a blob
- Revoke SAS token
- Stream disk data or incremental changes to Bacula

3.3 Bacula files

The backup will create the following backup files for each guest VM:

- A single empty file to match a guest VM name to its UUID /@azure-vm/<vmUuid>_<vmName>.name
- A single configuration metadata file: /@azure-vm/<vmUuid>/<epoch>_conf
- A unique raw data file for the guest VM OS disk /@azure-vm/<vmUuid>/<epoch>_osdisk
- A raw data file for each data disk: /@azure-vm/<vmUuid>/<epoch>_<index>

At restore time the user can identify the guest VM using the UUID to mark the corresponding files:

```
+-----------------------------------------------------------------------+
| filename                                                              |
+-----------------------------------------------------------------------+
| /@azure-vm/690d74f9-9ce4-45fb-9b23-149afebe18f7_VmName.name           |
```

(continues on next page)

(continued from previous page)

```
| /@azure-vm/690d74f9-9ce4-45fb-9b23-149afebe18f7/1661160371_conf   |
| /@azure-vm/690d74f9-9ce4-45fb-9b23-149afebe18f7/1661161330_0      |
| /@azure-vm/690d74f9-9ce4-45fb-9b23-149afebe18f7/1661167810_osdisk |
+-------------------------------------------------------------------+
```

3.4 Restore

The Azure plugin restores data as a new guest VM.

The restore process goes through the following steps.

- Receive configuration metadata file.
- Restore OS disk.
- Restore data disks.
- Create guest VM from OS disk and information from configuration metadata file.
- Attach data disk to newly created guest VM
- Delete locally generated files

Note: If debug level (debug parameter) is at its maximum level (9) locally generated file are not deleted

---

3.5 Disk restore process

This section will describe the restore process for any type of disk.

- Restore disk locally
- Apply incremental changes if any
- Create empty disk on Azure side.
- Issue SAS on the newly created disk.
- Upload locally restored data to disk.
- Revoke SAS

Note: Since disk are restored locally before being upload to azure there is a free space requirement of at least the size of the full VM

3.6 Incremental backup fallback process

During incremental backup if the plugin fails for any reason to compute incremental changes for a disk the disk will be backed up as a full image instead. A message will be displayed and the resulting backup will end with a `Backup OK with warning` status. At restore time the user must carefully mark the files for restore by ignoring older full/incremental images of said disk.

3.7 Snapshot safekeeping and data usage

Microsoft Azure uses a disk by disk snapshot policy. Bacula will keep track of snapshot relation between one another and delete snapshots that are no longer relevant. However, multiple snapshots of the same disk can be expected on the storage at any given time.

On Microsoft Azure incremental snapshots appear to have the same size as the full disk. However, the data contained inside the snapshot will be the differential data. Per Azure documentation incremental snapshots are billed for the used size only.

4 Plugin Installation

The Microsoft Azure plugin, can work on any machine with a Bacula File Daemon.

Since all backup/restore interactions are network based, any Bacula Enterprise File Daemon with access to the necessary Microsoft Azure endpoints can be used to run the plugin.

4.1 Dependencies

The Microsoft Azure plugin needs two additional software to work

- Microsoft Azure CLI
- AzCopy

After the installation the final step is to log into `Azure CLI` by running `az login` or `az login --use-device-code` if the plugin runs on a server with no GUI.

4.2 Configuration of the Bacula File Daemon

The `Plugin Directory` directive of the `File Daemon` resource in */opt/bacula/etc/bacula-fd.conf* should point to the location where the `azure-vm-fd.so` plugin is installed. The default directory is: */opt/bacula/plugins*

```
FileDaemon {
  Name = bacula-fd
  Plugin Directory = /opt/bacula/plugins
  ...
}
```


4.3 Installation of the Plugin

For more information about plugin installation see Linux: Install File Daemon (Client)

5 Plugin Configuration

The plugin is configured using `Plugin Parameters` defined in the "Include" section of a FileSet resource (Bacula Director configuration).

5.1 Generic Plugin Parameters

The following Microsoft Azure plugin parameters impact any type of Job (Backup, Restore, Query).

abort_on_error[= <0 or 1>] Specifies whether or not the plugin should abort its execution if a fatal error happens during backup or restore. This parameter is optional. The default value is `0`.

tenant_id=<string> Will create, with subscription_id, an Azure profile to gain access to Azure sdk functionalities. This parameter is mandatory.

subscription_id=<string> Will create, with tenant_id, an Azure profile to gain access to Azure sdk functionalities. This parameter is mandatory.

application_id=<string> Another parameter to gain access to Azure sdk functionalities. This parameter is mandatory.

application_secret=<string> Another parameter used to gain access to Azure sdk functionalities. This parameter is mandatory

connection_key=<string> Is used to create a `BlobServiceClientBuilder` to interact with blobs. This is a connection string from an Azure storage account. This parameter is mandatory.

debug=[0,9] Specifies the level of debug with `0` being no debug and `9` being the highest level of debug. Warnings and errors are always sent to the joblog and if any debug level is set those messages are sent to the debug file as well. For the Microsoft Azure plugin `1` displays debug level message, `2` displays trace level message. Any value higher than `2` displays additional information about external libraries that handle those values on their own. When the debug level is at its maximum, locally generated disk file at restore will not be deleted. This parameter is optional. The default value is `0`.

group=<string> Specifies which from which group the parser should retrieve the parameters. This parameter is optional. If this parameter is not set and a config file exsist at `/opt/bacula/etc/azure-vm.conf` the first available group will be chosen by default.

See the `Configuration File` section for more informations and examples about configuration files for the Azure plugin.

Note: `tenant_id`, `application_id` and `application_secret` parameters can be manually generated by running the following command `az ad sp create-for-rbac --role Owner --scopes /subscriptions/<subscription-id>`. If successful the command will output four different values.

- `tenant` = `tenant_id`
- `appId` = `application_id`
- `password` = `application_secret`
- `displayName` = Does not correspond to any of the plugin parameter

Note: Parameters from the fileset have precedence over parameters provided by a configuration file.

5.2 Backup Plugin Parameters

include=<Java Regexp> Specifies a list of guest VM names to backup.

exclude=<Java Regexp> Specifies a list of guest VM names to not backup.

vm=<guest VM name> Specifies the name of a single guest VM to backup.

The use of regular expressions in the parameters `include=` and `exclude=` must be a Java compatible regular expression.

In order to be backed up the guest VM must match the `include=...` predicate and not match the `exclude=...`. A guest VM that matches the `vm=...` will be backed up regardless of the include/exclude specifications.

By default all guest VMs match the include predicate and not the exclude. Therefore, if none of the parameters `vm=...`, `include=...` and `exclude=...` are provided, all available guest VMs hosted on the Microsoft Azure hypervisor will be backed up.

On the other hand, if the parameter `vm=...` is specified, all guest VMs will no longer match `include=...` predicate. This means that if only `vm=...` parameter is specified, no other guest VM will be backed up.

See FileSet Examples section for examples of include/exclude/vm setups.

5.3 Restore Plugin Parameters

new_hostname=<String> Specified when a guest VM should be restored with a specific name.

A restore job can only restore one guest VM at a time. To select the relevant VM the user should interact with Bacula bconsole and `mark` all files in the guest VM folder identified by its UUID.

```
cwd is: /
$ ls
$ @azure-vm/
$ cd @azure-vm
cwd is: /@azure-vm/
$ ls
010a9727-ec67-4f3c-ac8b-129310764aa1/
010a9727-ec67-4f3c-ac8b-129310764aa1_testvm.name
$ mark 010a9727-ec67-4f3c-ac8b-129310764aa1*
4 files marked.
$ done
```


6 Configuration File

For this plugin it is possible to pass arguments from a configuration file. Configuration file follows the `.ini` format. Each combination of parameters has to be identified by a bracketed group name then each parameters are set via a `key = value` format.

The azure plugin will look for a configuration file located at `/opt/bacula/etc/azure-vm.conf`. If such file is present the plugin will then look for the relevant group to get parameters from. If the `group` parameter is not set then the first available group will be chosen.

Parameters provided by the Fileset have precedence over the parameters provided by the configuration file.

6.1 Configuration File Example

Here is an example of a configuration file at `/opt/bacula/etc/azure-vm.conf` for two different azure configurations.

```
[main]
tenant_id = 11111111-aaaa-bbbb-cccc-000000000000
subscription_id = 12345678-abcd-efgh-ijkl-01234567890ab
connection_key = DefaultEndpointsProtocol=[...]
application_id = 12345678-2222-3333-4444-555555555555
application_secret = qqqq~----------------------qwertzuiopasd


[second]
tenant_id = aaaaaaaa-1111-2222-3333-zzzzzzzzzzzz
subscription_id = abcdefgh-1234-5678-9012-abcdefghijkl
connection_key = DefaultEndpointsProtocol=[...]
application_id = 87654321-1111-2222-3333-44444444444
application_secret = 9qqqq~----------------------qw123456789d
```

For the following fileset example since no `group` parameter is provided the plugin will look for the fist available group. In this case it will be `main`.

```
Fileset {
Name = "FS_AZURE_VM"
  Include {
    Options {
      signature = MD5
      compression = LZO
    }
    Plugin = "azure-vm: vm=vm"
  }
}
```

For the following fileset example the parameter group `second` will be selected.

```
Fileset {
Name = "FS_AZURE_VM"
  Include {
    Options {
      signature = MD5
```

(continues on next page)

(continued from previous page)

```
      compression = LZO
    }
    Plugin = "azure-vm: vm=vm group=second"
  }
}
```

7 FileSet Examples

In the example below, all guest VMs will be backed up:

```
FileSet {
     Name= Azure-all-params-all-vms
     Include {
       Plugin="azure-vm: tenant_id=12345678-abcd-1234-efgh-0123456789ab subscription_
↪id=abcdefgh-1234-abcd-1234-abcdefghijkl connection_key=DefaultEndpointsProtocol=https;
↪AccountName=example;AccountKey=aBcdEfGhijKlMNOpqrstuvwxyZ012345;EndpointSuffix=core.
↪windows.net application_id=87654321-1111-2222-3333-44444444444 application_
↪secret=9qqqq~--------------------qw123456789d"
     }
}
```

In the example below, all guest VMs will be backed up but this time the `tenant_id`, `subscription_id`, `application_id`, `secret_id` and `connection_key` are set inside a configuration file located at `/opt/bacula/etc/bacula-vm.conf` into a group named `main`

Note: In all the future examples, it will be assumed for less verbose FileSets, that `tenant_id`, `subscription_id`, `application_id`, `secret_id` and `connection_key` are set by the same configuration file and that the group `main` is the first available group.

```
FileSet {
      Name= Azure-no-params
      Include {
        Plugin="azure-vm: group=main"
      }
}
```

In the example below, a single guest VM with a name of "VM1" will be backed up.

```
FileSet {
    Name= Azure-One-Vm
    Include {
        Plugin="azure-vm: vm=VM1"
    }
}
```

In the example below, all guest VMs which have `prod` in their name will be backed up.

```
FileSet {
    Name= Azure-prod
```

(continues on next page)

(continued from previous page)

```
    Include {
        Plugin="azure-vm: include=(.*)prod(.*)"
    }
}
```

In the example below, all guest VMs which have `prod` in their name but do not start with `test` will be backed up.

```
FileSet {
    Name= Azure_no_test
    Include {
        Plugin="azure-vm: include=(.*)prod(.*) exclude=^test(.*)"
    }
}
```

In the example below, all `prod` VMs will be backed up. All `test` VMs will be ignored except for a VM named `exception.test`

```
FileSet {
    Name= Azure_prod_no_test_except
    Include {
        Plugin="azure-vm: include=(.*)prod(.*) exclude=(.*)test(.*) vm=exception.test"
    }
}
```

8 Specific bconsole Query Commands

The Bacula Enterprise Azure-vm plugin supports also the query parameter.

The plugin answer the query in the form of tuple `key=value`.

The plugin supports two operators if none of them is passed the plugin will answer with `Query not recognized, possible value={CONNECTION, VM}`

8.1 CONNECTION

This query check if the system is logged to a Microsoft Azure account

When the query parameter is `CONNECTION`, the returned value will be either `OK` or `NOK`, indicating whether the connection was successful or not.

```
.query plugin="azure-vm:" client=client-fd parameter=CONNECTION
info=Query
CONNECTION=OK
```


8.2 VM

This query sends a request to Azure that lists all available guest VMs. This query displays one `key=value` per virtual machine where `key` is `VM` and `value` is the name of the virtual machine.

The example below shows the execution of a correctly formatted `VM` query that finds three different guest VMs.

```
.query plugin="azure-vm:" client=client-fd parameter=VM
info=Query
VM=Virtual1
VM=Virtual2
VM=Virtual3
```

8.3 GROUP

This query looks for a configuration file located at `/opt/bacula/etc/azure-vm.conf` and prints all available parameter groups in a `GROUP=groupName`

```
.query plugin="azure-vm:" client=client-fd parameter=GROUP
info=Query
GROUP=main
GROUP=second
```

8.4 VERSION

Print information relative to azure and `azcopy` CLI

```
.query plugin="azure-vm:" client=client-fd parameter=VERSION
info=Query
azure-cli=2.40.0
azure-cli-core=2.40.0
azure-cli-telemetry=1.0.8
azcopy=10.16.0
```

9 Troubleshooting

Nothing here (yet)

10 Limitations

- Restore job needs to have the size of the full VM in free space
- On restore, the `where=` option is ignored
- The plugin may restore only one guest VM per restore job
- Virtual full jobs are not supported

