# Bareos Python Plugins Introduction

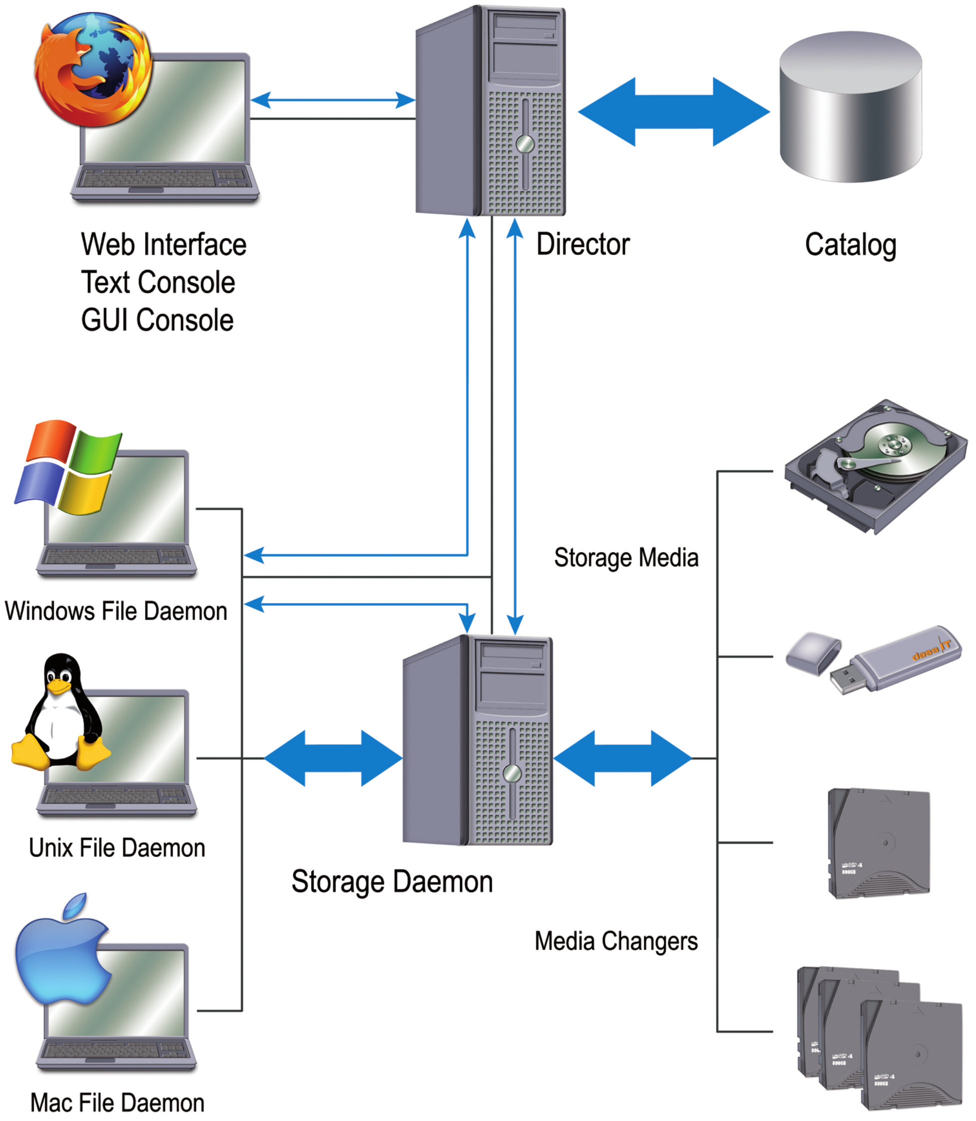
Stephan Dühr

Feb 3, 2017

## Agenda

- Bareos architecture and terminology
- Introduction
- Plugin overview (FD, SD, DIR)
- Detailed View at FileDaemon Plugins
- FD Plugin Examples
- Discussion of Plugin Ideas, Feedback, Questions
- Slides:
  http://download.bareos.org/bareos/people/sduehr/

# Architecture Overview



## Why Python Plugins?

- Extend Bareos functionality
  - Without touching the Bareos C code
  - Can react on numerous events (in contrast to pre- and postscripts)
  - Modify Fileset
  - Special incremental handling possible
  - Connect to other systems (Monitoring, Ticket, Hypervisors, Cloud, Logging, Indexer i.e. elasticsearch)
  - Application specific actions on backup and restore

# New Bareos Python Plugin interface

- Python knowledge wide spread among technical consultants, admins and devops
- Arbitrary Python modules available to handle a large numbers of application / APIs
- Plain Python script for FD / SD / DIR plugins
- For FD additional class based approach, since 15.2 also for SD and DIR
- Need Python version 2.6 or newer
- Uses distribution provided Python packages
- C code already prepared for Python 3.x

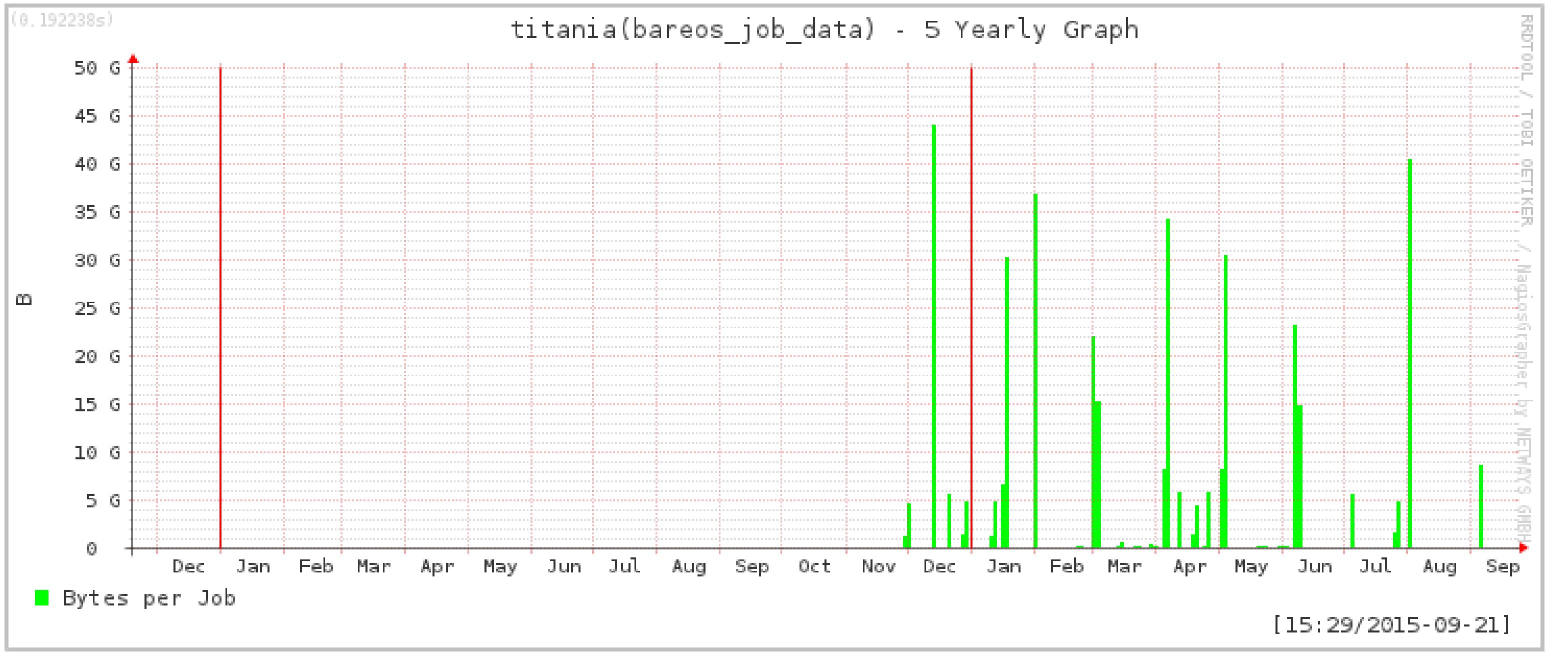
## Bareos Python Plugin interface

- Plugins configured via Bareos configuration
  Pass plugin options to FD plugins
- Bareos core calls functions from the plugins on defined events
- Plugins can influence the backup process and modify Bareos variables
- Plugin usage must be explicitly enabled:

```
Plugin Directory = /usr/lib/bareos/plugins
Plugin Names = python
```

## Director Plugins: NSCA-sender

- Icinga / Nagios NSCA plugin
  - Submits job results and performance data by NSCA right after a job has finished.
  - OK: Bareos job titania-data.2015-09-20_20.05.01_47 on titania-fd with id 19374 level D, 0 errors, 75433922 jobBytes, 24 files terminated with status T



# Director Plugins: NSCA-sender

- Icinga / Nagios NSCA plugin configuration as Job directive:

```
Director {
   Plugin Directory = /usr/lib64/bareos/plugins
   Plugin Names = "python”
   …
}


Job {
   ...
   DIR Plugin Options="python:module_path=/usr/lib64/bareos/plugins:
module_name=bareos-dir-nsca-sender:monitorHost=icingahost:
checkHost=my_bareosFD:checkService=bareos_backup"
…
}
```

- https://github.com/bareos/bareos-contrib/tree/master/dir-plugins/nagios_icinga

## Director Plugins

- Base Class available, that provides basic and derived job information:
  - self.jobName = bareosdir.GetValue(context, brDirVariable['bDirVarJobName'])
  - self.jobLevel = chr(bareosdir.GetValue(context, brDirVariable['bDirVarLevel']))
  - self.jobType = bareosdir.GetValue(context, brDirVariable['bDirVarType'])
  - self.jobId = int(bareosdir.GetValue(context, brDirVariable['bDirVarJobId']))
  - self.jobClient = bareosdir.GetValue(context, brDirVariable['bDirVarClient'])
  - self.jobStatus = bareosdir.GetValue(context, brDirVariable['bDirVarJobStatus'])
  - self.Priority = bareosdir.GetValue(context, brDirVariable['bDirVarPriority'])
  - self.jobPool = bareosdir.GetValue(context, brDirVariable['bDirVarPool'])
  - self.jobStorage = bareosdir.GetValue(context, brDirVariable['bDirVarStorage'])
  - self.jobMediaType = bareosdir.GetValue(context, brDirVariable['bDirVarMediaType'])
- Derived information
  - self.jobTotalTime = self.jobEndTime - self.jobInitTime
  - self.jobRunningTime = self.jobEndTime - self.jobRunTime
  - self.throughput = self.jobBytes / self.jobRunningTime

# FD Plugins

- How to enable Python Plugins in FD?
- Install the package `bareos-filedaemon-python-plugin`
- In `/etc/bareos/bareos-fd.d/client/myself.conf` add or uncomment:

```
FileDaemon {
  ...
  Plugin Directory = /usr/lib64/bareos/plugins
  Plugin Names = python
  ...
}
```

- Restart FD: `systemctl restart bareos-fd`
- Like for SD and Dir Plugins, `Plugin Names` can be omitted. Then all Plugins matching glob `*-fd.so` will be loaded
- With `Plugin Names = python`, FD will only load `python-fd.so`

# FD Plugins

- Multiple plugins possible
- The `Plugin` parameter in Director's FileSet resource determines which python plugin is used with which paramters. Syntax:
  `Plugin = python:module_path=<path-to-python-modules>:module_name=<python-module-to-load>:<custom-param1>=<custom-value1>:...`
- `module_path` and `module_name` are mandatory (used by `python-fd.so`)
- Anything else is arbitrary, the complete string is passed to the hook function `parse_plugin_definition()`
- two Plugin-Types:
  Command-Plugins and Option-Plugins (difference will be explained later)

## How to configure and use a FD Plugin

- `BareosFdPluginLocalFileset.py` is a sample plugin that comes with the `bareos-filedaemon-python-plugin` package
- Create the file `/etc/bareos/bareos-fd.d/client/myself.conf` with content:

```
FileSet {
 Name = "test_PyLocalFileset_Set"
 Include {
   Plugin = "python:module_path=/usr/lib64/bareos/plugins:module_name=bareos-fd-local-fileset:filename=/tmp/filelist"
   Options {
     signature = MD5
     Compression = LZ4
   }
 }
}
```

- Create the file `/etc/bareos/bareos-dir.d/job/test_PyLocalFileset_Job.conf` with content:

```
Job {
  Name = "test_PyLocalFileset_Job"
  JobDefs = "DefaultJob"
  FileSet = "test_PyLocalFileset_Set"
}
```

## How to configure and use a FD Plugin

- Add some filenames to `/tmp/filelist`, eg.
  ```
  find /etc/yum.repos.d -type f > /tmp/filelist
  ```
- Run `bconsole` and enter the following:
  ```
  [root@vgr-f24test1 ~]# bconsole
  Connecting to Director localhost:9101
  1000 OK: bareos-dir Version: 16.2.4 (01 July 2016)
  Enter a period to cancel a command.
  *reload
  reloaded
  *run job=test_PyLocalFileset_Job
  Using Catalog "MyCatalog"
  Run Backup job
  JobName:  test_PyLocalFileset_Job
  Level:    Incremental
  Client:   bareos-fd
  Format:   Native
  FileSet:  test_PyLocalFileset_Set
  Pool:     Incremental (From Job IncPool override)
  Storage:  File (From Job resource)
  When:     2017-02-01 07:17:40
  Priority: 10
  OK to run? (yes/mod/no): yes
  Job queued. JobId=2
  You have messages.
  ```

## How to configure and use a FD Plugin

- Still in bconsole, look at the job messages

```
*messages
02-Feb 19:00 f23bareos1-dir JobId 2: No prior Full backup Job record found.
...
02-Feb 19:00 f23bareos1-sd JobId 2: Ready to append to end of Volume "Full-
0001" size=35659138
02-Feb 19:00 f23bareos1-fd JobId 2: Starting backup of
/etc/yum.repos.d/bareos:master.repo
...
Termination:            Backup OK
```

- List the files that have been backed up

```
*list files jobid=2
Using Catalog "MyCatalog"
 /etc/yum.repos.d/fedora.repo
 /etc/yum.repos.d/fedora-updates.repo
 /etc/yum.repos.d/bareos:master.repo
 /etc/yum.repos.d/fedora-updates-testing.repo
 /etc/yum.repos.d/fedora-local.repo
 /etc/yum.repos.d/fedora-updates-local.repo
```

## How to configure and use a FD Plugin

- How to run a restore (1)

```
[root@f23bareos1 ~]# bconsole
Connecting to Director f23bareos1:9101
1000 OK: f23bareos1-dir Version: 16.1.0 (02 January 2016)
Enter a period to cancel a command.
*restore
Automatically selected Catalog: MyCatalog
Using Catalog "MyCatalog"

First you select one or more JobIds that contain files
to be restored. You will be presented several methods
of specifying the JobIds. Then you will be allowed to
select which files from those JobIds are to be restored.

To select the JobIds, you have the following choices:
     1: List last 20 Jobs run
     ...
     5: Select the most recent backup for a client
     ...
    13: Cancel
Select item:  (1-13): 5
Automatically selected Client: f23bareos1-fd
The defined FileSet resources are:
     1: SelfTest
     2: test_PyLocalFileset_Set
Select FileSet resource (1-2): 2
```

## How to configure and use a FD Plugin

- Run a restore

```
Select FileSet resource (1-2): 2
+-------+-------+----------+----------+---------------------+------------+
| jobid | level | jobfiles | jobbytes | starttime           | volumename |
+-------+-------+----------+----------+---------------------+------------+
|     2 | F     |        6 |    2,479 | 2017-02-01 07:17:52 | Full-0001  |
+-------+-------+----------+----------+---------------------+------------+
You have selected the following JobId: 2
Building directory tree for JobId(s) 2 ...
6 files inserted into the tree.

You are now entering file selection mode where you add (mark) and
remove (unmark) files to be restored. No files are initially added, unless
you used the "all" keyword on the command line.
Enter "done" to leave this mode.

cwd is: /
$ mark *
6 files marked.
$ done
Bootstrap records written to /var/lib/bareos/bareos-dir.restore.1.bsr

The job will require the following
   Volume(s)                 Storage(s)                SD Device(s)
===========================================================================

    Full-0001                  File                      FileStorage

Volumes marked with "*" are online.
6 files selected to be restored.
```

# How to configure and use a FD Plugin

- How to run a restore (2)

```
6 files selected to be restored.
Using Catalog "MyCatalog"
Run Restore job
JobName:          RestoreFiles
Bootstrap:        /var/lib/bareos/bareos-dir.restore.1.bsr
Where:            /tmp/bareos-restores
Replace:          Always
FileSet:          Linux All
Backup Client:    bareos-fd
Restore Client:   bareos-fd
Format:           Native
Storage:          File
When:             2017-02-01 07:37:06
Catalog:          MyCatalog
Priority:         10
Plugin Options:   *None*
OK to run? (yes/mod/no): yes
Job queued. JobId=6
```

## How to configure and use a FD Plugin

- How to run a restore (3)

```
You have messages.
*mes
01-Feb 07:37 bareos-dir JobId 3: Start Restore Job RestoreFiles.2017-02-01_07.37.09_08
01-Feb 07:37 bareos-dir JobId 3: Using Device "FileStorage" to read.
01-Feb 07:37 bareos-sd JobId 3: Ready to read from volume "Full-0001" on device
"FileStorage" (/var/lib/bareos/storage).
01-Feb 07:37 bareos-sd JobId 3: Forward spacing Volume "Full-0001" to file:block
0:14960306.
01-Feb 07:37 bareos-dir JobId 3: Bareos bareos-dir 16.2.4 (01Jul16):
  Build OS:               x86_64-redhat-linux-gnu redhat Fedora release 24 (Twenty
Four)
  JobId:                  3
  Job:                    RestoreFiles.2017-02-01_07.37.09_08
  Restore Client:         bareos-fd
  Start time:             01-Feb-2017 07:37:11
  End time:               01-Feb-2017 07:37:11
  Elapsed time:           0 secs
  Files Expected:         6
  Files Restored:         6
  Bytes Restored:         976
  Rate:                   0.0 KB/s
  FD Errors:              0
  FD termination status:  OK
  SD termination status:  OK
  Termination:            Restore OK
```

# How do FD Plugins work (1)

- When a Job is run, Director passes plugin definition to FD, eg. `module_path=/usr/lib64/bareos/plugins:module_name=bareos-fd`

  FD (`python-fd.so`) does the following:
  - instantiates new Python interpreter
  - extends the Python search path with the given `module_path`
  - imports the module given by `module_name` (for the example above, would be `bareos-fd.py`)
  - makes callback functions available for Python, use `import bareosfd` in Python code

# How do FD Plugins work (2)

- Constants to be used as callback function parameters are defined in `bareos_fd_consts.py`, use eg.
  `from bareos_fd_consts import bJobMessageType, bFileType, bRCs`
  in Python code. All defined constants see:
  http://doc.bareos.org/doxygen/dd/dbb/namespacebareos__fd__consts.html
  or
  `/usr/lib64/bareos/plugins/bareos_fd_consts.py`
- calls `load_bareos_plugin()` in the python plugin code
- calls `parse_plugin_definition(context, plugindef)` in the python code
  - `plugindef` is the complete string as configured in Director (`Plugin = ...`), to be parsed by python code
- different processing loop depending on type of Plugin (Command/Option)

## FD Command-Plugin Configuration

- Command Plugin Configuration in Include section of FileSet Resource in bareos-dir.conf:

```
FileSet {
  Name = "test_PyLocalFileset_Set"
  Include {
     Plugin = 
"python:module_path=/usr/lib64/bareos/plugins:module_n
ame=bareos-fd-local-fileset:filename=/tmp/datafile"
  }
}
```

## FD Option-Plugin Configuration

- Option Plugin Configuration in Options section of Include Section of FileSet Resource in bareos-dir.conf:

```
FileSet {
  Name = "test_PyOptionInteract_Set"
  Include {
    File = /data/project_1
    Options {
       Plugin =
"python:module_path=/usr/lib64/bareos/plugins:module_name=bareos-
fd-file-interact"
    }
  }
}
```

- Note: for Option-Plugin must define files to backup using `File = ...` while for Command-Plugin need not

## Difference FD Command-/Option-Plugins (1)

- Major Difference:
  - Command-Plugin determines what is being backed up, must also handle Diff/Inc itself
  - Option-Plugin gets which files to backup based on whats configured in Director, Diff/Inc handling done by FD

# Difference FD Command-/Option-Plugins (2)

- Command-Plugin processing
  - `start_backup_file(context, savepkt)` must set savepkt properties for each file to back up
  - `plugin_io(context, IOP)` must handle IO Operations
    - Backup: open(r), read, close
  - `end_backup_file(context)`
    - must return `bRCs['bRC_More']` if more files to backup
    - must return `bRCs['bRC_OK']` to finish the looping
  - `handle_backup_file()` is not called

# Difference FD Command-/Option-Plugins (3)

- Option-Plugin processing
  - `handle_backup_file(context, savepkt)` called for each file to be processed, `savepkt` defined by FD
  - `plugin_io()` handling in the same manner as for Command-Plugin
  - `start_backup_file()` and `end_backup_file()` are not called

# FD Plugins – Callback Functions

- Functions provided by `python-fd.so` that can be called from Python code, enabled by
  `import bareosfd`
- Complete list:
  http://regress.bareos.org/doxygen/html/d5/d0e/python-fd_8h_source.html
- Most important callback functions:
  - `bareosfd.JobMessage()`: Error-/Info-/Warning-Messages
    - are passed to Director, appear in messages and logs
  - `bareosfd.DebugMessage()`: Debug-Messages with numeric level
    - only visible when running FD in debug-mode with `-d <level>`
  - `bareosfd.GetValue()`: used to get variables from FD

# FD Plugins – Class Based Approach

- Python FD Plugin can be monolithic
- Better: use classes and inheritance to reuse existing code easier and reduce code redundancy
- To support this approach, the package `bareos-filedaemon-python-plugin` package provides
  - `BareosFdPluginBaseclass.py`
    - Parent Class to inherit from
  - `BareosFdWrapper.py`
    - defines all functions a plugin needs and “wraps” them to  the corresponding methods in the plugin class
  - a Plugin entry-point module glues them together

# Messaging

- `bareosfd.DebugMessage()`: Debug only
  - `bareosfd.DebugMessage(context, level, "message\n");`
    - `context`: used to pass information from core to plugin, don't touch
    - `level`: Debug Level, use >= 100
  - Sample:
    ```
    bareosfd.DebugMessage(context, 100, "handle_backup_file called with " + str(savepkt) + "\n");
    ```
  - To see debug output, run FD in foreground:
    ```
    systemctl stop bareos-fd
    bareos-fd -f -d 100
    ```
    This would output debug messages of level <= 100

# Messaging

- `bareosfd.JobMessage()`: Sent to messaging system
  - `bareosfd.JobMessage(context, bJobMessageType, “Message\n");`
    - Type: Controls job result, `M_INFO, M_ERROR, M_WARNING, M_ABORT`
      http://regress.bareos.org/doxygen/html/dd/dbb/namespacebareos__fd__consts.html
  - Sample:

```
bareosfd.JobMessage(context, bJobMessageType['M_INFO'], "Option Plugin file interact
on" + savepkt.fname + "\n");
```

## Return Codes

- Return Codes control processing, no impact on overall job status.
- Depending on context / function
- Use consts:

```
return bRCs['bRC_OK'];
return bRCs['bRC_Skip']; # skips current file
return bRCs['bRC_Error']; # error but continue
return bRCs['bRC_More']; # in end_backup_file, more files to backup
...
```

# FD Plugin: bareos-fd-local-fileset.py

- Reads a local file on fd with filenames to backup
  - Demonstration / template plugin, functionality can be achieved better by fileset configuration:
    `File = “\\</localfile/on/client”`
- Configuration in fileset resource as command plugin (extends fileset):

```
Plugin = "python:module_path=/usr/lib64/bareos/plugins:module_name=bareos-fd-local-fileset:filename=/tmp/datafile"
```

- Plugin: `/usr/lib64/bareos/plugins/bareos-fd-local-fileset.py`
  Code excerpt:

```
import bareos_fd_consts
import BareosFdWrapper
from BareosFdWrapper import *
import BareosFdPluginLocalFileset
def load_bareos_plugin(context, plugindef):
    BareosFdWrapper.bareos_fd_plugin_object = \
        BareosFdPluginLocalFileset.BareosFdPluginLocalFileset(
            context, plugindef)
    return bareos_fd_consts.bRCs['bRC_OK']
```

- Rest is done in class `BareosFdPluginLocalFileset`

## BareosFdPluginLocalFileset

- Class inherits from `BareosFdPluginBaseclass`
- Method `parse_plugin_definition()`
  Parses the options, filename is mandatory
  Reads filenames from file into array `self.files_to_backup`
- Method `start_backup_file()` asks plugin, if there is anything to backup, sets `savepkt`:

```
file_to_backup = self.files_to_backup.pop();
savepkt.fname = file_to_backup;
savepkt.type = bFileType['FT_REG'];
return bRCs['bRC_OK'];
```

# BareosFdPluginLocalFileset

- Method `end_backup_file()` called to ask plugin if there is more to backup:

```
if self.files_to_backup:
    # there is more to backup, go to start_backup_file again
    return bRCs['bRC_More'];
else
    # no more to backup from this plugin, done
    return bRCs['bRC_OK'];
```

- Basic IO operations covered in BareosFdPluginBaseclass
  - Method `plugin_io()` handles file read / write operations

## BareosFdPluginLocalFileset

- For restore: some more things to do
  - Directories have to be created

```
def create_file (self,context, restorepkt):
    FNAME = restorepkt.ofname;
    dirname = os.path.dirname (FNAME);
    if not os.path.exists(dirname):
        os.makedirs(dirname);
    if restorepkt.type == bFileType['FT_REG']:
        open (FNAME,'wb').close();
        restorepkt.create_status = bCFs['CF_EXTRACT'];
    return bRCs['bRC_OK'];
```

  - Similar in method `plugin_io()` for writing
- Overload this method in your class, if you need different handling

## Understanding `plugin_io()`

- Is called with different I/O operation types:
  - `IO_OPEN`, `IO_CLOSE`
    - Once at begin/end
    - Distinguish backup/restore on `IOP.flags`
    - Open a file for read or write here
    - Or start another tool to read/write in a piped way
  - `IO_READ`, `IO_WRITE`
    - Repeated until end of data
    - Passes data in chunks of 64K to FD by filling a buffer variable
- Study and compare plugin_io() in:
  - https://github.com/bareos/bareos/blob/master/src/plugins/filed/BareosFdPluginBaseclass.py
  - https://github.com/bareos/bareos-contrib/blob/master/fd-plugins/mysql-python/BareosFdMySQLclass.py

# MySQL Plugin

- FD Plugin for MySQL Backup contributed by Evan Felix (https://github.com/karcaw)
- Available at https://github.com/bareos/bareos-contrib/tree/master/fd-plugins/mysql-python
- Package available at http://download.bareos.org/bareos/contrib/
- runs `mysql -B -N -e 'show databases'` to get the list of databases to back up or use databases specified by option `db`
- runs `mysqldump %s --events --single-transaction` for each database, using `subprocess.Popen()` (pipe)
- `plugin_io()` reads the pipe, no temporary local diskspace needed for the dump
- Restore to dumpfile

# MySQL Plugin

- Configuration in Fileset-Include resource:
  `Plugin= "python:module_path=/usr/lib64/bareos/plugins:`
  `module_name=bareos-fd-mysql:db=test,mysql"`
- More options with default settings:
  - mysqlhost = localhost
  - Dumpoptions = --events --single-transaction
  - drop_and_recreate = true
    Adds --add-drop-database –databases to mysqldump options
  - mysqluser = <bareos-fd user (root)>
  - mysqlpassword =
  - dumpbinary = mysqldump
- Possible enhancements:
  - add restore-option to directly pipe data into mysql instead of creating a dump file
- There is another new plugin now based on Percona XtraBackup to make full and incremental backups of MySQL/MariaDB:
  https://github.com/bareos/bareos-contrib/tree/master/fd-plugins/bareos_percona

## Getting started for developing

- Setup a VM for development and testing
  - Installation Instruction see

    http://doc.bareos.org/master/html/bareos-manual-main-reference.html#InstallingBareos
- Or download a preconfigured openSUSE based Bareos appliance: https://susestudio.com/search?q=bareos
- Or use the Vagrantfile from https://gist.github.com/sduehr if you like Vagrant and KVM
- Manual steps (Fedora 24) see next slide

## Bareos Basic Install and Setup

- Setup Repo
```
cd /etc/yum.repos.d
curl -O http://download.bareos.org/bareos/release/latest/Fedora_24/bareos.repo
```
- Install Bareos
```
dnf install bareos bareos-database-postgresql
```
- Install and setup PostgreSQL
```
dnf install postgresql-server
postgresql-setup --initdb
systemctl enable postgresql
systemctl start postgreql
```
- Create Bareos catalog DB
```
su – postgres
cd /usr/lib/bareos/scripts
./create_bareos_database
./make_bareos_tables
./grant_bareos_privileges
exit
```
- Enable and start Bareos Services
```
systemctl enable bareos-fd; systemctl start bareos-fd; systemctl status bareos-fd
systemctl enable bareos-sd; systemctl start bareos-sd; systemctl status bareos-sd
systemctl enable bareos-dir; systemctl start bareos-dir; systemctl status bareos-dir
```

# Bareos Basic Install and Setup

- Check if it works, run a backup:

```
[root@vgr-f24test1 ~]# bconsole
Connecting to Director localhost:9101
1000 OK: bareos-dir Version: 16.2.4 (01 July 2016)
Enter a period to cancel a command.
*run job=backup-bareos-fd
Using Catalog "MyCatalog"
Run Backup job
JobName:  backup-bareos-fd
Level:    Incremental
Client:   bareos-fd
Format:   Native
FileSet:  SelfTest
Pool:     Incremental (From Job IncPool override)
Storage:  File (From Job resource)
When:     2017-02-01 06:48:51
Priority: 10
OK to run? (yes/mod/no): yes
Job queued. JobId=1
You have messages.
*messages
01-Feb 06:49 bareos-dir JobId 1: No prior Full backup Job record found.
01-Feb 06:49 bareos-dir JobId 1: No prior or suitable Full backup found in catalog. Doing FULL
backup.
...
Termination:            Backup OK
```

# More Plugin Ideas

- More ideas – application specific plugins
  - oVirt/RHEV:
    - Has a Backup-Restore API, we will start soon working on using http://www.ovirt.org/develop/release-management/features/storage/backup-restore-api-integration/
  - Snapshot based KVM (some ideas next slide)
  - IMAP / Cyrus: restore to specific mailbox directory
  - Open Xchange (backup / restore of single objects)
  - Kolab
  - other SQL or NoSQL Databases
  - Docker?
    - Pets vs. Cattle: Is there anything to do for backup?
  - Other applications?

## More Plugin Ideas

- Ideas regarding KVM Backup
  - KVM/qemu has nothing like VMware CBT
  - Proposals like http://wiki.qemu.org/Features/Livebackup have never been completed/accepted
  - a CBT-like approach using external QCOW2 snapshots/overlays could be derived from https://kashyapc.fedorapeople.org/virt/lc-2012/snapshots-handout.html
  - Guest-Agent quiescing actions should be looked at
  - Performance impact of overlay chaining?
  - But now there's http://wiki.qemu.org/Features/IncrementalBackup

# Contact and links

- Website: http://www.bareos.org
- Documentation: http://doc.bareos.org
- Package Repos: http://download.bareos.org/bareos/
- All Bareos GitHub Repositories: https://github.com/bareos
- GIT Bareos contrib for plugins: https://github.com/bareos/bareos-contrib
- Mailinglists: see https://www.bareos.org/en/open-source.html
- Bugtracker: https://bugs.bareos.org
  - Please read https://www.bareos.org/en/HOWTO/articles/how-to-create-a-bugreport.html
- Thesis Proposals: https://www.bareos.org/en/thesis-proposals.html
- Open Source Backup Conference: http://osbconf.org/ next time end of September 2016
  - also has an archive with slides and videos from previous years
- Subscriptions, Support, References, Partner: http://www.bareos.com