APPM

User Manual

Table of Contents

Minimum System Requirements 1
Central Repository 2
- Installation 2
- Upgrade 3
- Start & Stop 3

Oracle Database Schema 4
- Register database 4
- Create / Upgrade Schema 4

Collector 5
- Install 5
- Configuration Options 6

Minimum System Requirements

- Operating System: Linux or Windows
- RAM: 4 GB
- HDD: 30 GB
- CPU: 2 core

Supported Oracle database versions are (including) 11g and beyond, any edition.

Central Repository

Installation

Following steps must be executed as user `root` in order to install APPM Central Repository:

1. Install `docker` and `docker-compose`. Example:

```
apt install docker.io docker-compose unzip
```

2. Create folder `/srv/appm/`. Consider setting up dedicated mountpoint for `/srv/appm/vol/pg-dbf` because this is where samples are physically stored (it can take considerable amount of disk space).

```
mkdir /srv/appm/
```

3. Download `appm-repository-<VERSION>.zip` from http://release.abakus.si/appm/latest/ and extract it to `/srv/appm/` like:

```
wget http://release.abakus.si/appm/latest/appm-repository-<VERSION>.zip
unzip appm-repository-<VERSION>.zip -d /srv/appm
```

4. Run installer which loads docker containers, creates initial postgres database and installs system-d service. Note that this step might take a few minutes.

```
/srv/appm/utl/install.sh
```

5. Enable log rotating and backup scripts (using `crontab -e` and then entering the following):

```
SHELL=/bin/bash
15 * * * *  /srv/appm/utl/logrotate.sh > /srv/appm/vol/hk-log/logrotate-$(date
'+%Y-%m-%d').log 2>&1
30 1 * * *  /srv/appm/utl/backup.sh    > /srv/appm/vol/hk-log/backup-$(date
'+%Y-%m-%d').log 2>&1
```

6. Open browser and access APPM at following url. Default username/password are `admin` / `change_me`.

```
https://<HOSTNAME>/appm/
```

Backup

Backup script `/srv/appm/utl/backup.sh` assumes that `/srv/appm/vol/pg-dbf` is a dedicated mountpoint **based on LVM LV.** It creates backups to `/backup/pg-backup`. This script can be used as-is or adopted to your specific needs (you may backup PostgreSQL database in any way you want).

Change SSL Certificate

Certificate and private key files (both in PEM format) are required in order to produce pkcs12 keystore, compatible with

default setting of Wildfly. Example:

```
cd /srv/appm/utl/
./create-keystore.sh /path/to/cert.pem /path/to/private-key.pem
```

After creating keystore using this procedure, follow the Upgrade procedure (because ./install.sh regenerates docker-compose.yml file, which points to newly created keystore). Alternatively, edit the docker-compose.yml yourself and restart wildfly container. Relevant section of docker-compose.yml:

```
wildfly:
  ...
  volumes:
     ...
     - '/etc/appm/appm-
keystore.p12:/opt/abakus/wildfly/as/standalone/configuration/application.keystore:ro
'
```

Upgrade

1. Stop `aba-appm` service:

   ```
   systemctl stop aba-appm
   ```

2. Remove current docker containers and images by running:

   ```
   docker system prune -a
   ```

3. Delete everything except `/srv/appm/vol/pg-dbf` (samples are physically stored here).
4. Follow the same procedure as for installation.

Start & Stop

All services are running as a set of docker containers and can be started/stopped using systemctl:

```
systemctl stop aba-appm
systemctl start aba-appm
```

Oracle Database Schema

Register database

Firstly, register new monitored Oracle Database(s) with APPM by navigating to `Repository` → `Databases` → `Create Database`. Following fields are required:

- `Collector Name` - name of database and also name of Postgres schema containing its samples. Use lowercase when possible.
- `Connection String` - TNS Connection string in format `hostname:port/service_name`.
- `APPM Schema` - Name of Oracle Database schema that will be created in one of the following steps.
- `History Days` - Samples older than this number of days are automatically deleted from repository.

Create / Upgrade Schema

1. Navigate to `Repository` → `Databases`
2. Select Oracle database(s) on which you wish to either create or upgrade `APPM2` schema (you can name it differently) and click `Upgrade Schema` button (it creates the schema if it does not yet exist).
    1. provide `SYSDBA` credentials, or,
    2. click `appm.sql` link to download the script and execute it yourself as `sqlplus / as sysdba @appm.sql`
3. Optionally select `Create collector user` in order to create `APPM_COLLECTOR` schema on selected Oracle database(s). (This schema is not required if collector will be using BEQ (Bequeath) connections and logging on directly as SYSDBA).
    1. User is automatically created/upgraded if `Create collector user` is specified, or,
    2. Click `collector.sql` link to download the script and execute it yourself as `sqlplus / as sysdba @collector.sql`

Note that every SQL sent to Oracle database during the installation of those schemas is logged in `/srv/appm/vol/wf-log/server.log`.

Installation will create/update `APPM2` schema which includes views to `V$` performance views. It does not include any data, because samples are stored in separate repository database.

Global objects that are created as a part of APPM installation are named like `APPM_<NAME>`, where `APPM_` is a constant prefix.

Created schema could take up to 250MB of disk space.

Collector

Install

Collector can be installed on any host, not necessarily on the same host as the database. However, when it is installed on the database host it can utilize `bequeath` connections and dedicated `sysdba` connections which makes sample collection a bit faster. Such `bequeath` connections require **Oracle Instant Client** to be installed (it is not required otherwise).

1. Obtain Adopt Open JDK, version 11 (HotSpot) from https://adoptopenjdk.net/?variant=openjdk11&jvmVariant=hotspot and install it to `/opt/abakus/java/jdk11`.

```
wget https://github.com/AdoptOpenJDK/openjdk11-binaries/releases/download/jdk-11.0.6%2B10/OpenJDK11U-jdk_x64_linux_hotspot_11.0.6_10.tar.gz
mkdir -p /opt/abakus/java/
tar zxvf OpenJDK11U-jdk_x64_linux_hotspot_11.0.6_10.tar.gz -C /opt/abakus/java/
ln -s /opt/abakus/java/jdk-11.0.6+10 /opt/abakus/java/jdk11
```

2. Obtain APPM Agent from http://release.abakus.si/appm/latest/appm-collector-<VERSION>.zip

```
wget http://release.abakus.si/appm/3.0.37/appm-collector-3.0.37.zip
mkdir -p /opt/abakus/appm_collector
unzip appm-collector-3.0.37.zip -d /opt/abakus/appm_collector
```

3. Configure APPM Agent by creating file `/etc/appm/collector.ini`. Fully functional example of this file is as follows. You only really need to change:
    1. `url` setting to point to where APPM Central Inventory is listening at https.
    2. last four lines for each registered database (`[dbname]` must be exactly the same as what is displayed under `Repository` → `Databases` → `Collector Name` column)
    3. please refer to the next chapter for list of all possible settings in this file.

```
[default]
java_home = /opt/abakus/java/jdk11
instant_client = /opt/abakus/instantclient_19_3
url = https://myhost/appm/collect
stashLocation = /opt/abakus/appm_collector/stash/
alertLocation = /opt/abakus/appm_collector/log/
alertHistory = 14
upload.sleepMillis = 5000
;
[dbname]
username = appm_collector
password = appm_password
database = host:1521/service
```

4. Decide under which user to run the collector. It should not be running as `root`. Only if you wish to connect `/ as sysdba` then choose a user which is a member of configured "OS DBA" group. If unsure, use `abakus` user like we do by default.

```
chown -R abakus:abakus /opt/abakus/appm_collector
```

5. Create systemd service and modify it to run as previously selected user.

```
cp /opt/abakus/appm_collector/appm_collector.service
/etc/systemd/system/appm_collector.service
vi /etc/systemd/system/appm_collector.service # here change the User and Group
parameters as discussed in previous step
systemctl enable appm_collector.service
systemctl start appm_collector.service
```

Configuration Options

Configuration in `/etc/appm/collector.ini` is composed of two sections. The `[default]` Section and per-database section.

`[default]` Section

- `java_home = /opt/abakus/java/jdk11`
  startup scripts will launch this agent using java from this java home
- `instant_client = /opt/abakus/instantclient_18_3`
  startup scripts will set environment to use OCI from this IC. This is only needed if you intend to use BEQ connections (/ as sysdba)
- `url = https://localhost/appm/collect`
  collected samples are pushed web services on this url
- `stashLocation = /opt/abakus/appm_collector/stash/`
  collected samples are stored on local disk in this folder until they are successfully pushed to central repository
- `alertLocation = /opt/abakus/appm_collector/log/`
  file named "collector-alert.yyyy-mm-dd.log" will be available in this folder. It contains error messages if such events occur.
- `alertHistory = 14`
  number of days to keep "collector-aler.yyyy-mm-dd.log" files. Files older than this number of days are deleted.
- `upload.sleepMillis = 5000`
  go through all available files every sleepMillis to upload each file found. Files which have been successfully uploaded are deleted.

Per-database Sections

Note that section name (`example` and `sample` in the following example) must match `Collector Name` as displayed in APPM application. You can go to `Repository` → `Databases` to see `Collector Name` for each of the registered databases.

```
; connect using SQL*Net
[example]
username = APPM_COLLECTOR
password = demo
database = atlas.abakus.si:1521/dev.abakus.si
```

```
; connect using BEQ
[sample]
username = /
password = /
database = dev:/oracle/db_se/18.4.0/dbhome_1
pdb = DEVP
schema = APPM2
userrole = sysdba
```

Collector Settings

Following is the default collector configuration which you probably should not change (unless if you are certain of what you're doing). In most cases config file should not contain any of those options (since following defaults apply). That being said, if you want to override any of those you can do it in either Default Sections (for all databases) or per-database in specific Database Section.

Following options can be overridden for each collector:

- `queueSize`: results of select are stored in array. If query returns more than `queueSize` rows then only queueSize rows are kept in memory, flushed, and then next `queueSize` rows are read into memory and so on..
- `sleepMillis`: time between each iteration (collection SQL is executed every sleepMillis)
- `alertMillis`: if single iteration take more than thresholdMillis to complete it is logged to collector-alert.log (in alertLocation)
- `rotateIterations`: start to write to new file every n iteration. One iteration happens every `sleepMillis`. If sleepMillis = 1000 (1 sec), and rotateIterations=10, then you get new stash file every 10 seconds.
- `reconnect`: number of iterations after which database session is closed and reconnected.
    1. `0` disables reconnect (the same session all the time)
    2. `1` create new session, collect samples, disconnect (on each collection)
    3. `>1` reconnect when this number of collections is done
- `enabled`: if `0` then this collector won't run. All collectors are enabled (`1`) by default.
- `blackSetFlushInterval` and `blackSetFlushSize`: those are only valid for `plan` and `sql` collector. Those collectors keep a list of last few already colllected `sql_id` and `plan_hash_value` values. `Size` parameter determines how many entries can such list have at most and `Interval` specifies retention period (in hours) for this list.

List of Default Settings

```
ash.queueSize = 1000
ash.sleepMillis = 1000 # 1 second
ash.alertMillis = 250
ash.rotateIterations = 10
ash.reconnect = 0"
ash.enabled = 1
```

```
longops.queueSize = 500
longops.sleepMillis = 3000 # 3 seconds
longops.alertMillis = 200
longops.rotateIterations = 3
longops.reconnect = 0
longops.enabled = 1
```

```
bsh.queueSize = 500
bsh.sleepMillis = 10000            # 10 seconds
bsh.alertMillis = 500
bsh.rotateIterations = 3
bsh.reconnect = 0
bsh.enabled = 1
```

```
transaction.queueSize = 500
transaction.sleepMillis = 15000    # 15 seconds
transaction.alertMillis = 200
transaction.rotateIterations = 1
transaction.reconnect = 0
transaction.enabled = 1
```

```
process.queueSize = 3000
process.sleepMillis = 30000        # 30 seconds
process.alertMillis = 200
process.rotateIterations = 1
process.reconnect = 0
process.enabled = 1
```

```
sql.queueSize = 1000
sql.sleepMillis = 63000            # 63 seconds
sql.alertMillis = 1000
sql.rotateIterations = 3
sql.reconnect = 1
sql.blackSetFlushInterval = 8      # 8 hours
sql.blackSetFlushSize = 1000
sql.enabled = 1
```

```
plan.queueSize = 1000
plan.sleepMillis = 67000           # 67 seconds
plan.alertMillis = 1000
plan.rotateIterations = 3
plan.reconnect = 1
plan.blackSetFlushInterval = 8     # 8 hours
plan.blackSetFlushSize = 1000
plan.enabled = 1
```

```
parameter.queueSize = 500
parameter.sleepMillis = 28800000   # 8 hours
parameter.alertMillis = 250
parameter.rotateIterations = 1
parameter.reconnect = 1
parameter.enabled = 1
```

```
version.queueSize = 5
version.sleepMillis = 86400000     # 1 day
version.alertMillis = 250
version.rotateIterations = 1
version.reconnect = 1
version.enabled = 1
```