APPM

Installation Guide

Table of Contents

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. Setup access to Debian or Oracle Linux repository containing APPM release:

```
# Debian:
wget -O - https://appm.abakus.si/tools/setup-debian-repo.sh | bash
apt update && apt full-upgrade
apt install aba-appm-repository
```

```
# Oracle Linux:
wget -O - https://appm.abakus.si/tools/setup-oel-repo.sh | bash
dnf update
dnf install aba-appm-repository
```

2. 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
```

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

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

Logs

When things don't work as expected, you can check log(s) at:

- `/srv/appm/vol/wf-log/server.log`
- `/srv/appm/vol/pg-log/*`

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

This is optional because APPM uses self-signed certificate by default.

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, rerun installer using following command:

```
systemctl stop aba-appm
cd /srv/appm/utl/
./install.sh
```

This will regenerate 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'
```

Quick helper to generate CSR:

```
openssl req -new -newkey rsa:4096 -nodes -keyout private.key -out public.csr
```

Upgrade

First, upgrade APPM software:

```
# Debian:
apt update && apt upgrade
```

```
# Oracle Linux
dnf upgrade
```

And then go to graphical interface, Repository → Databases and for each database, click on "Upgrade Schema" to update/recreate Oracle objects under APPM2 schema. See Create / Upgrade

Schema chapter for more info.

You should also restart APPM collector after upgrading the database objects.

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. either:
    1. Select Oracle database(s) on which you wish to either create or upgrade `APPM2` and `APPM_COLLECTOR` schemas and click `Upgrade Schema` button (it creates the schemas if they don't yet exist). You will need to enter `SYSDBA` credentials in order to allow APPM to create them.
    2. **or**, if you don't want to provide `SYSDBA` credentials, click `appm.sql` link to download the script and execute it yourself as `sqlplus / as sysdba @appm.sql`. Then do the same with `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.

Finally, restart following services in order to force APPM collector to reconnect:

```
systemctl restart aba-appm
systemctl restart appm_collector # optional and only available in EE edition
```

External Collector

This is an optional feature, available only in Enterprise Edition. There is "internal" collector embedded within application server in Free Edition.

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. Install:

```
# Debian
apt install aba-appm-collector
```

```
# Oracle Linux
dnf install aba-appm-collector
```

2. 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
```

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
```