

Releasing(OUTDATED)

This page describes the release process of Apache Flink.

Verifying a Release Candidate

PLEASE NOTE: It is **NOT** necessary to run all checks to cast a vote for a release candidate. However, you should clearly state which checks you did. The release manager needs to ensure that each following check was done.

Legal: (required checks for a valid ASF compilant release)

- Check if checksums and GPG files match the corresponding release files
- Verify that the source archives do not contains any binaries
- Check if the source release is building properly with Maven (including license header check (default) and checkstyle). Also the tests should be executed (mvn clean verify)
 - check build for custom hadoop version
 - check build for Scala 2.11
- Verify that the LICENSE and NOTICE file is correct **for the binary and source release**.
 - All dependencies must be checked for their license and the license must be ASL 2.0 compatible (<http://www.apache.org/legal/resolved.html#category-x>)
 - The LICENSE and NOTICE files in the root directory refer to dependencies in the source release, i.e., files in the git repository (such as fonts, css, JavaScript, images)
 - The LICENSE and NOTICE files in `flink-dist/src/main/flink-bin` refer to the binary distribution and mention all of Flink's Maven dependencies as well
- Check that all POM files point to the same version (mostly relevant to examine quickstart artifact files)
- Read the README.md file

Functional: (checks for delivering a release with good quality)

- Run the `start-local.sh`, `start-cluster.sh` scripts and verify that the processes come up
 - Examine the *.out files (should be empty) and the log files (should contain no exceptions)
 - Test for Linux, OS X, Windows (for Windows as far as possible, not all scripts exist)
 - Shutdown and verify there are no exceptions in the log output (after shutdown)
 - Check all start+submission scripts for paths with and without spaces (`./bin/*` scripts are quite fragile for paths with spaces)
- Verify that the examples are running from both `./bin/flink` and from the web-based job submission tool
 - Should be run on
 - local mode (`start-local.sh`)
 - cluster mode (`start-cluster.sh`)
 - multi-node cluster (can simulate locally by starting two taskmanagers)
 - The `flink-conf.yml` should define more than one task slot
 - Results of job are produced and correct
 - Check also that the examples are running with the build-in data and external sources.
 - Examine the log output - no error messages should be encountered
 - Web interface shows progress and finished job in history
- Test on a cluster with HDFS.
 - Check that a good amount of input splits is read locally (JobManager log reveals local assignments)
- Test against a Kafka installation
- Test the `./bin/flink` command line client
 - Test `"info"` option, paste the JSON into the plan visualizer HTML file, check that plan is rendered
 - Test the parallelism flag (`-p`) to override the configured default parallelism
- Verify the plan visualizer with different browsers/operating systems
- Verify that the quickstarts for scala and java are working with the staging repository for both IntelliJ and Eclipse.

- in particular the dependencies of the quickstart project need to be set correctly and the QS project needs to build from the staging repository (replace the snapshot repo URL with the staging repo URL)
 - The dependency tree of the QuickStart project must not contain any dependencies we shade away upstream (guava, netty, ...)
 - Test that quickstart archetypes are working on all platforms
- Run examples on a YARN cluster
- Run all examples from the IDE (Eclipse & IntelliJ)
- Run an example with the RemoteEnvironment against a cluster started from the shell script
- Pay special attention to new features
- Test recovery and exactly-once guarantees with master and worker failures [@todo @uce Will update this with scripts](#)
 - YARN (see <https://github.com/apache/flink/pull/1213> for details)
 - 2.3.0 <= version < 2.4.0
 - Set yarn.application-attempts for Flink
 - Set yarn.resourcemanager.am.max-attempts for YARN (upper bound on number of failures)
 - Note: it's expected for these Hadoop versions that all containers are killed when the application master fails
 - 2.4.0 <= version < 2.6.0
 - Important: in this version the task manager containers should stay alive when the application master is killed
 - 2.6.0 <= version
 - Check that the application is only killed by YARN after the system has seen the maximum number of application attempts during one interval
 - Standalone
 - Start multiple JobManager and TaskManager instances
 - Kill random instances (make sure that enough task slots and standby job managers are available)
- Test building a SBT project depending on Flink and an optional dependency (connector, gelly, flink-ml).
- Test the Scala/SBT giter8 template `g8 tillrohrmann/flink-project`
- Test the Scala/SBT vanilla project in <https://github.com/tillrohrmann/flink-project>
- Test the Scala/SBT quickstart script under <https://flink.apache.org/q/sbt-quickstart.sh>

Documentation

- Check that all links work, the front page is up to date
- Check that new features are documented and updates to existing features are written down.
- Ensure that the migration guide from the last release to the new release is available and up to date.

Creating a release candidate

- Read and understand: <http://www.apache.org/dev/release-publishing.html>
- Read <http://www.apache.org/dev/release-signing.html> and create yourself a PGP key
- run the `./tools/create_release_files.sh` script from the Flink repo, with the following parameters: (the call below as used to create RC1 of Flink 0.8.1)

```
sonatype_user=YOURAPACHEID sonatype_pw=YOURAPACHEIDPASSWORD NEW_VERSION=0.8.1 RELEASE_CANDIDATE="rc1"
RELEASE_BRANCH=release-0.8 OLD_VERSION=0.8-SNAPSHOT USER_NAME=YOURAPACHEID
GPG_PASSPHRASE=YOURGPGPASSPHRASE GPG_KEY=YOURGPGKEY
GIT_AUTHOR="`git config --get user.name` <`git config --get user.email`>" ./create_release_files.sh
```

Note: Make sure to run the `./tools/create_release_files.sh` script (in particular the "mvn deploy" call) with Java 8 to release the java8 module to mvn central as well.

- Don't forget to update the documentation configuration to the new release version
- Usually the `create_release_files.sh` script needs to be adopted a bit depending on the used linux distribution
- Open repository.apache.org, login in with your Apache user, and close the staging repository you created (DO NOT PRESS "RELEASE!")
- Check that the release files are located in your people.apache.org home directory
- go to the `./tools/flink` directory and push the release commit to `release-x.y.z-rcn` branch. (DO NOT CHANGE THE COMMIT OR COMMIT THE LOCAL CHANGES, the release commit has already been created.)
- Send the VOTE mail to the [dev@flink.a.o](mailto:dev@flink.apache.org) list, containing the release commit hash, the link to your people.apache.org and to the staging repository.

Releasing a Release Candidate

- Create a git tag for the release
- Upload binaries to svn repo at: <https://dist.apache.org/repos/dist/release/flink> (sync needs 24 hours)
- Release staging repository <https://repository.apache.org/> to Maven Central.
 - After this step, add the release to the Apache Report Helper at <https://reporter.apache.org/addrrelease.html?flink> (if you are not a PMC member, ask one for help)
- Update Flink website with updated download URLs and Maven artifacts (by updating the `_config.yml`)
 - Don't forget to update the quickstarts versions

- Update list of contributors wiki page with new contributors in the release
- Update Homebrew: <https://gist.github.com/EronWright/b62bd3b192a15be4c200a2542f7c9376>
- Publish release announcement blog post
- Email to dev, news, user AT flink.a.o; also announce AT apache.
- Update the reference Flink version in the japicmp maven plugin for the API stability checks to the just released major version.