How to release

The last line is to remove the *previous* version, since only the most recent version on a particular branch should be in the dist directory (older versions are archived automatically, see http://archive.apache.org/dist/bigtop/ and http://www.apache.org/dev/mirrors.html).

This page describes how to make a release of Apache Bigtop. Most of the content is shamelessly stolen from the the Whirr release process – thanks Whirr team!

0. Before you begin.

Apache Bigtop is following a predictable "train model" of releases popularized by the Ubuntu Linux. Currently releases happen every quarter. A release of Bigtop is a culmination of a quarter of development activities and also a testament to the particular state of the Hadoop ecosystem at the time. IOW, what are the most cutting edge versions of the components that still work together as an integrated system.

All release activities are coordinated by a volunteer release manager (RM) who is chosen during the release planning. If you happen to be a first time RM here's what you need to do in order to be able to carry out the mechnics of the release

1. Generate PGP code signing keys. It's a good idea to generate RSA-based key (see the Binary Package Deployment section below)
    a. Because of long-standing bug in RPM sigs validation, you can not use subkeys for signing. See here for more information.
2. Ensure that your PGP signing keys are available in: https://dist.apache.org/repos/dist/release/bigtop/KEYS More details can be found here. Basically you need:
    a. $ svn co https://dist.apache.org/repos/dist/release/bigtop; cd bigtop
    b. $ (gpg --list-sigs <your name> && gpg --armor --export <your name>) >> KEYS
    c. svn commit -m "Adding <your name>'s code signing key"
3. Copy the new KEYS file to the release folder /www/www.apache.org/dist/bigtop on people.apache.org

If you are not already a member of the Web Of Trust (WOT) it would be a good idea to do so. (contact one of the prior release managers, e.g. Rvs, Cos, etc...). You can read more about key signing here.

Ensure that you have setup your ssh keys on people.apache.org, otherwise you'll have to enter your login password a number of times (best use ssh-agent for this as well). A good overview of this process can be found here (ssh-copy-id and ssh-agent in particular)

Workaround: an open issue BIGTOP-1499 requires that the release repo is clean. You can either make a new clone; or remove everything non-related to the repo (including files and directories listed in `.gitignore` file

Configure scm-publish plugin

~~Please make sure that you have correct configuration for scm-plugin SVN Provider condifuration. Create `$user.home/.scm/svn-settings.xml` with the following content~~

```
<svn-settings> <user>your apache user id</user> <password>your LDAP password</password> </svn-settings>
```

~~set the permissions on the directory and the file to 700 as it contains your clean-text password (DANGER!)~~

See http://stackoverflow.com/questions/3618330/what-is-the-format-of-svn-settings-xml-for-use-with-maven-scm-plugin how to change your scm:svn: url and provide a password.

Done with the above? Proceed with the rest of the steps:

1. Scrubbing the JIRA.

3-4 weeks before a committed release date a release manager (RM) for a given release is supposed to start a process of scrubbing the JIRAs. This involves:

1. taking care of all the patch available issues for a given release
2. send an email to the bigtop-dev mailing list reminding the community of the upcoming release and asking them to spend 2-3 days raising the priority of the "must fix" issues for a given release to a "Blocker" status AND making sure that Fix Versions field is set to an upcoming release. Of course, it goes without saying that people have to volunteer to work on this issues – this is not an exercise of assigning work to others. Typically everybody has their own favorite issues that they would like to see fixed in the upcoming release, however do encourage folks to also take a look at the Unscheduled issues since those tend to accumulate lurkers. Also 'git grep' for the 'FIXME/WORKAROUD' prefix in the source code as to identify anything that doesn't need to be worked around anymore
3. after waiting for 2-3 days for community to respond and for the Blockers to settle down – make sure that the resulting list looks reasonable and that there is a general expectation that the release can happen given the state of the source base.
4. create the version tag for the next milestone release: navigate to the https://issues.apache.org/jira/plugins/servlet/project-config/BIGTOP/versions and add the tag there of the form 0.X.Z
5. move all the non-blocker issues assigned to the current release to the next one by navigating to (make sure to replace 0.X.Y with your actual version number): https://issues.apache.org/jira/secure/IssueNavigator!executeAdvanced.jspa?jqlQuery=project+%

3D+BIGTOP+AND+resolution+%3D+Unresolved+AND+fixVersion+%3D+%220.X.Y%22+and+priority+%21%3D+blocker and then select 'bulk update' from the tools menu (WARNING: MAKE SURE TO UNCHECK the 'Send mail for this update' toggle at the very bottom of the edit screen!!!)

2. Create a Release Series Branch

This only needs doing if this is the first release in a series (X.Y.0).

- Update CHANGES.txt in master to replace `Release X.Y.0 (unreleased changes)` with `Release X.Y.0 - YYYY-MM-DD`. Commit:

```
git commit -m "Preparing for release X.Y.Z"
```

- Create a branch for the release series:

```
git checkout -b branch-X.Y.Z origin/master
```

- Add `A.B.C-SNAPSHOT (unreleased changes)` to CHANGES.txt in master.
- Bump the version number in the master branch:

```
./gradlew setversion -Pnextversion="A.B.C-SNAPSHOT"
```

- Commit these changes to the master and push.
- Checkout the release branch

```
git checkout branch-X.Y.Z
```

- Update the release branch's version information: the version number in the release branch ends in `-SNAPSHOT`, but we need to remove this for the release. For example, `0.8.0-SNAPSHOT` needs to be changed to `0.8.0`.

```
./gradlew setversion -Pnextversion="X.Y.Z"
```

- Commit these changes to the release branch and push

```
git commit -m "Changing version to X.Y.Z"
```

- Rename docker images and repo URLs:

```
grep -R --color :trunk- * |grep yaml
grep -R --color "http://repos.bigtop.apache.org/releases/1.3.0" *
```

- Shall you need to commit additional fixes into ongoing release, the commits should go to the release branch and only then be merged into master. Doing this other way around forces `git cherry-pick` which leads to discrepancies in the commit hash-codes and makes the branch look untidy and hard to follow.

3. Generate the Release Notes

JIRA has the ability to generate release notes automatically (this is why it is so important to keep the fix version number accurate).

https://issues.apache.org/jira/secure/ConfigureReleaseNote.jspa?projectId=12311420

Manually check this list for accuracy! I've repeatedly seen closed bugs that were not fixed (i.e., duplicate) marked with a fix version, so that they incorrectly show up in this list. Find those, edit them to remove the fix release (only actually fixed bugs should have a fix release) and re-run the report. A better way to deal with it is to run

```
fixVersion = X.Y.Z AND project = BIGTOP AND resolution = Duplicate ORDER by priority DESC
```

Select the correct version. Ask for both plain text and HTML formats.

Paste the plain text notes to the top of CHANGES.txt. Paste the HTML version into *src/site/xdoc/release-notes.xml*. Check this into master and the branch.

Wrap the title ("Release Notes - Bigtop - Version X.Y.Z") inside an <h3> element.

4. Commit and Tag

Tag, and push the changes and the tag to git:

```
git tag release-x.y.z -m "Bigtop X.Y.Z release."
git push --tags
```

5. Build and run Package and Smoke Tests

5.1. Build bigtop/slaves Docker images

Create a release specific job to build images:

https://ci.bigtop.apache.org/view/Docker/job/Docker-Toolchain-1.4.0/

Make sure all the built images are uploaded to Dockerhub

https://hub.docker.com/r/bigtop/slaves/tags/

Make sure all the built images are downloaded on docker-slave-06 and docker-slave-07

https://ci.bigtop.apache.org/view/Docker/job/Docker-Toolchain-1.4.0-pull/

5.2. Build RPM/DEB packages

Create a release specific job to build packages:

https://ci.bigtop.apache.org/view/Releases/job/Bigtop-1.4.0/

Download all built packages via archive download feature provided by Jenkins on a machine that you want to proceed the signing:

```
# for all OS and arch supported in Bigtop
for i in centos-7 fedora-26 opensuse-42.3 debian-9 ubuntu-16.04; do \
  for j in amd64-slave aarch64-slave ppc64le-slave; do \
    wget https://ci.bigtop.apache.org/view/Releases/job/Bigtop-1.4.0/DISTRO=${i},PLATFORM=amd64-slave
/lastSuccessfulBuild/artifact/*zip*/archive.zip
  done
done
```

5.3. Sign RPM packages and yum repos

Ref: **BIGTOP-2736** - Automate last mile of release process OPEN

Startup a docker images that is RPM based system:

```
cd ~/
# Change the image for AARCH64 or PPC64LE
docker run -ti --rm -v $PWD:/tmp bigtop/puppet:1.4.0-centos-7 bash
```

Prepare the environment for signing:

```
gpg --import YOUR_CODE_SIGNING_SECRET_KEY
echo "%_gpg_name YOUR_CODE_SIGNING_KEY_ID" > ~/.rpmmacros
yum install -y rpm-sign createrepo
```

Signing:

```
cd /tmp
OS=centos-7

# Sign all RPM packages (This step required to input passphrase, so don't copy and paste the entire script here)
rpm --addsign `find ${OS} -name \*rpm`

# Recreate the metadata for repository
createrepo ${OS}

# Armor the metadata
gpg --detach-sign --armor ${OS}/repodata/repomd.xml
```

[OPENSUSE ONLY]

```
gpg --armor --export evansye@apache.org > opensuse-42.3/repodata/repomd.xml.key
for i in `find opensuse-42.3/repodata -name *.xml.gz` opensuse-42.3/repodata/repomd.xml.key ; do gpg --detach-
sign --armor $i ; done
```

5.4. Sign DEB packages and apt repos

Ref: **BIGTOP-2736** - Automate last mile of release process OPEN , https://manpages.debian.org/jessie/dpkg-sig/dpkg-sig.1.en.html

Startup a docker images that is DEB based system:

```
cd ~/
docker run -ti --rm -v $PWD:/tmp bigtop/puppet:1.4.0-debian-9 bash
```

Prepare the environment for signing:

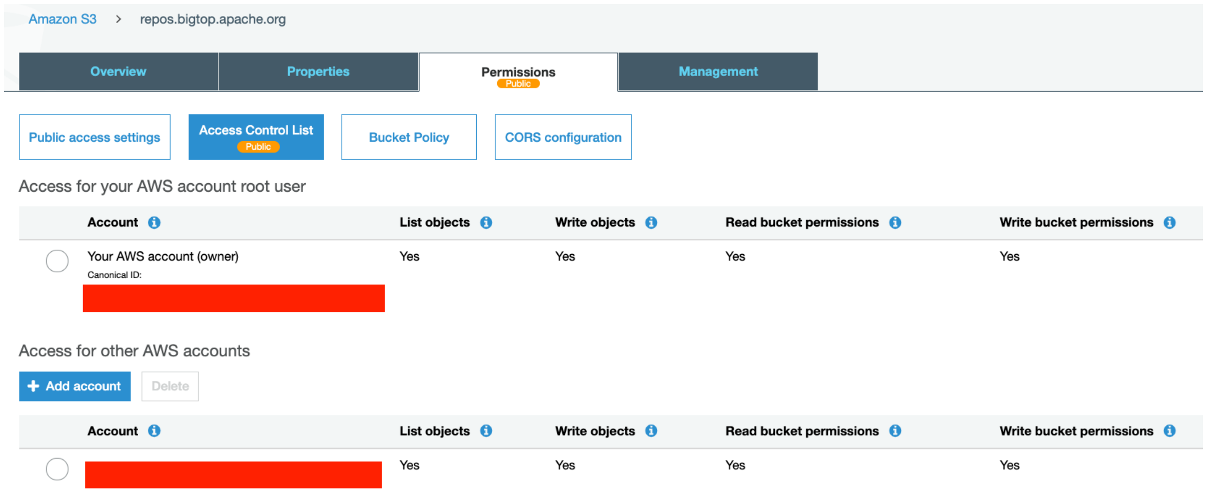
```
apt-get update
apt-get install -y gpg
apt-get install -y libterm-readkey-perl
apt-get install -y dpkg-sig
apt-get install -y reprepro
gpg --import YOUR_CODE_SIGNING_SECRET_KEY
```

Signing:

```
cd /tmp
VERSION=1.4.0
OS=debian-9
ARCH=amd64
SIGN_KEY=B7B4BD70
export GPG_TTY=$(tty)

# Sign DEB packages (This step required to input passphrase, so don't copy and paste the entire script here)
dpkg-sig --cache-passphrase --sign builder  `find ${OS}/ -name \*deb`

# Build signed apt repository
mkdir -p conf
cat > conf/distributions <<__EOT__
Origin: Bigtop
Label: Bigtop
Suite: stable
Codename: bigtop
Version: ${VERSION}
Architectures: ${ARCH} source
Components: contrib
Description: Apache Bigtop
SignWith: ${SIGN_KEY}
__EOT__

cat > conf/options <<__EOT__
verbose
ask-passphrase
__EOT__

rm -rf ${OS}/apt
reprepro --ask-passphrase -Vb . includedeb bigtop `find ${OS}/ -name \*deb`
```

5.5. Upload to S3

The easiest way to upload artifacts to S3 is via your own AWS account. Add your account(email) to bigtop's bucket in the section "Access for other AWS accounts":



Access for your AWS account root user

Account	List objects	Write objects	Read bucket permissions	Write bucket permissions
Your AWS account (owner) Canonical ID: [redacted]	Yes	Yes	Yes	Yes

Access for other AWS accounts

Account	List objects	Write objects	Read bucket permissions	Write bucket permissions
[redacted]	Yes	Yes	Yes	Yes

Once permission granted, you're able to use your account's access key and secret key with aws s3 sync command for upload:

```
aws s3 sync --acl public-read ./ubuntu-16.04/ s3://repos.bigtop.apache.org/releases/1.4.0/ubuntu/16.04/amd64/
```

The directory layouts on S3 bucket looked like the following:

```
repos.bigtop.apache.org/releases/1.2.1/centos/6/x86_64
repos.bigtop.apache.org/releases/1.2.1/centos/7/x86_64
repos.bigtop.apache.org/releases/1.2.1/fedora/25/x86_64
repos.bigtop.apache.org/releases/1.2.1/opensuse/42.1/x86_64
repos.bigtop.apache.org/releases/1.2.1/debian/8/x86_64
repos.bigtop.apache.org/releases/1.2.1/ubuntu/16.04/x86_64
```

For YUM repos, upload files /tmp/centos-7/* into repos.bigtop.apache.org/releases/1.2.1/centos/7/x86_64/. For example:

```
repos.bigtop.apache.org/releases/1.2.1/centos/7/x86_64/alluxio
repos.bigtop.apache.org/releases/1.2.1/centos/7/x86_64/ambari
...
repos.bigtop.apache.org/releases/1.2.1/centos/7/x86_64/repodata
...
```

For APT repos, upload files /tmp/debian-8/* into repos.bigtop.apache.org/releases/1.2.1/debian/8/x86_64/. For example:

```
repos.bigtop.apache.org/releases/1.2.1/debian/8/x86_64/conf
repos.bigtop.apache.org/releases/1.2.1/debian/8/x86_64/db
repos.bigtop.apache.org/releases/1.2.1/debian/8/x86_64/dists
repos.bigtop.apache.org/releases/1.2.1/debian/8/x86_64/pool
```

5.6. Create repo files

`Create one for each of our Distro. Following are examples for YUM and APT:`

```
# YUM
cat > bigtop.repo <<__EOT__
[bigtop]
name=Bigtop
enabled=1
gpgcheck=1
type=NONE
baseurl=http://repos.bigtop.apache.org/releases/1.2.1/centos/7/x86_64
gpgkey=https://dist.apache.org/repos/dist/release/bigtop/KEYS
__EOT__

gpg --detach-sign --armor bigtop.repo

# APT
cat > bigtop.list <<__EOT__
deb http://repos.bigtop.apache.org/releases/1.2.1/debian/8/x86_64        bigtop contrib
__EOT__

gpg --detach-sign --armor bigtop.list
```

Add your signed armored GPG key to `repos` directory to ease the key import for the users

```
gpg --armor --export <your name> >> GPG-KEY-bigtop
gpg --detach-sign --armor GPG-KEY-bigtop
```

Result looks like below:

```
GPG-KEY-bigtop
GPG-KEY-bigtop.asc
centos7/
debian9/
fedora26/
opensuse42.3/
ubuntu16.04/
```

5.7. Commit repo files into https://dist.apache.org/repos/dist/dev/bigtop/repos

6. Build and Deploy Artifacts

First we need to prepare a build environment. Mac OS X is unlikely supported because we'll run through some tests that depends on the OS. Here I'm running on Ubuntu-14.04 (There's an issue for running on 16.04 traced by BIGTOP-2830).

```
cd ~/
docker run -ti -u jenkins -v $PWD:/tmp bigtop/slaves:trunk-ubuntu-14.04 bash -l
# The subsequence commands should be executed inside the docker container
cp -r /tmp/.gnupg ~/
```

Create a maven settings file *~/.m2/settings.xml* with the following content:

```
<settings>
   <servers>
      <server>
         <id>apache.snapshots.https</id>
         <username>APACHE-ID</username>
         <password>APACHE-PASSWORD</password>
      </server>
      <server>
         <id>apache.staging.https</id>
         <username>APACHE-ID</username>
         <password>APACHE-PASSWORD</password>
      </server>
      <server>
         <id>apache.releases.https</id>
         <username>APACHE-ID</username>
         <password>APACHE-PASSWORD</password>
      </server>
   </servers>
   <profiles>
      <profile>
         <id>gpg</id>
         <properties>
            <gpg.executable>gpg</gpg.executable>
            <gpg.passphrase>GPG-PASSWORD</gpg.passphrase>
         </properties>
      </profile>
   </profiles>
   <activeProfiles>
      <activeProfile>gpg</activeProfile>
   </activeProfiles>
</settings>
```

Build the artifacts:

```
mvn site
mvn -Prelease package assembly:single
```

The following command deploys the binary release artifacts for iTest, tests and other helper files, checksums, and signatures (you will need to enter a GPG passphrase) to the Apache Staging repo.

```
mvn deploy -Prelease -f pom.xml
mvn deploy -Prelease -f bigtop-test-framework/pom.xml
mvn deploy -Prelease -f bigtop-tests/test-artifacts/pom.xml
mvn deploy -Prelease -f bigtop-tests/test-execution/pom.xml
```

For the last step to succeed you'd need to export HADOOP_HOME and HADOOP_CONF_DIR (BIGTOP-1464): the values don't matter - it is just a workaround. If this step fails with an *Access denied* error check that you have the required permissions on Nexus.

If the deployment step fails with an *Access denied* error check that you have the required permissions on Nexus.

If you have multiple keys, the build process seems to pick up the first one w/o asking. Make sure you're using the CODE SIGNING KEY by explicitly specifying it. For example:

```
mvn deploy -Prelease -f pom.xml -Dgpg.keyname="Evans Ye (CODE SIGNING KEY) <evansye@apache.org>"
mvn deploy -Prelease -f bigtop-test-framework/pom.xml -Dgpg.keyname="Evans Ye (CODE SIGNING KEY)
<evansye@apache.org>"
mvn deploy -Prelease -f bigtop-tests/test-artifacts/pom.xml -Dgpg.keyname="Evans Ye (CODE SIGNING KEY)
<evansye@apache.org>"
mvn deploy -Prelease -f bigtop-tests/test-execution/pom.xml -Dgpg.keyname="Evans Ye (CODE SIGNING KEY)
<evansye@apache.org>"
```

Login to https://repository.apache.org using your Apache SVN credentials. Click on **Staging** on the left. Then click on **orgapachebigtop** in the list of repositories. In the panel below you should see an open repository that is linked to your username and IP. Select this repository and click **Close**. This will close the repository from future deployments and make it available for others to view.

BIGTOP-1463 is open to track the improvements of the release process.

7. Copy Release Artifacts

The artifacts that end up in the distribution directory are the source distributions (along with their checksums and signatures), so they need to be copied from the Maven repo to a release candidate directory on apache dist, so the vote can begin:

```
VERSION=X.Y.Z
REPOSITORY_ID=xxxx # Should be a number, for example 1013. Find the number in NEXUS staging repository:
orgapachebigtop-1013

svn checkout https://dist.apache.org/repos/dist/dev/bigtop bigtop-dist-dev
cd bigtop-dist-dev

mkdir bigtop-$VERSION-RC1
cd bigtop-$VERSION-RC1
# md5 and sha1 are out-of-date and should not be used, see: http://www.apache.org/dev/release-distribution#sigs-
and-sums
wget --no-check-certificate https://repository.apache.org/content/repositories/orgapachebigtop-$REPOSITORY_ID
/org/apache/bigtop/bigtop/$VERSION/bigtop-$VERSION-project.tar.gz
wget --no-check-certificate https://repository.apache.org/content/repositories/orgapachebigtop-$REPOSITORY_ID
/org/apache/bigtop/bigtop/$VERSION/bigtop-$VERSION-project.tar.gz.asc
# manually generate sha256 and sha512
sha256sum bigtop-$VERSION-project.tar.gz > bigtop-$VERSION-project.tar.gz.sha256
sha512sum bigtop-$VERSION-project.tar.gz > bigtop-$VERSION-project.tar.gz.sha512

cd ..
svn add bigtop-$VERSION-RC1
svn ci -m "Apache Bigtop $VERSION-RC1"
```

8. Sanity Check

TODO

- Check the SHA1 checksums

9. Run the Vote

Run the vote on the dev@bigtop.apache.org

Here is an example email:

```
To: "Bigtop Developers List" <dev@bigtop.apache.org>
Subject: [VOTE] Release Bigtop version 1.1.0

This is the vote for release 1.1.0 of Apache Bigtop.

It fixes the following issues:
        https://issues.apache.org/jira/secure/ReleaseNote.jspa?projectId=12311420&version=12324841

The vote will be going for at least 72 hours and will be closed on Wednesday,
February 3rd, 2016 at noon PDT.  Please download, test and vote with

[ ] +1, accept RC1 as the official 1.1.0 release of Apache Bigtop
[ ] +0, I don't care either way,
[ ] -1, do not accept RC1 as the official 1.1.0 release of Apache Bigtop, because...

Source and binary files:
        https://dist.apache.org/repos/dist/dev/bigtop/1.1.0-rc1

Maven staging repo:
                https://repository.apache.org/content/repositories/orgapachebigtop-[YOUR REPOSITORY ID]

The git tag to be voted upon is release-1.1.0

Bigtop's KEYS file containing PGP keys we use to sign the release:
        https://dist.apache.org/repos/dist/release/bigtop/KEYS
```

The release needs 3 +1 votes from the PMC.

10. Roll Out

Assuming the vote passes, the release can be rolled out as follows:

Move Artifacts into Place

TODO Update the instructions as per new use of apache dist

This step makes the artifacts available on the mirrors.

```
VERSION=X.Y.Z #Example: 1.3.0
CANDIDATE=C #Example: RC2
svn co https://dist.apache.org/repos/dist/release/bigtop
cd bigtop
svn mv https://dist.apache.org/repos/dist/dev/bigtop/bigtop-$VERSION-$CANDIDATE https://dist.apache.org/repos
/dist/release/bigtop/bigtop-$VERSION
rm stable
ln -s bigtop-$VERSION stable
svn commit
```

Log in to https://repository.apache.org, click on **Staging** on the left. Select the repository that you closed earlier, and click **Release**, using a description like "Apache Bigtop X.Y.Z artifacts". This will make the artifacts publicly available.

You'll get an email from "**Apache Reporter Service**" asking to update the release info. If you aren't a member of the PMC, ask someone to log in and enter the release name and date.

Create permanent release tag under `rel/`

```
git checkout release-x.y.z
git tag rel/x,y,z -u <signing key ID>
git push --tags
```

Wait 24 Hours

It takes up to 24 hours for all the mirrors to sync, so don't announce the new release just yet.

Build and Deploy Site

Full version of this procedure is described and maintained in Deploying Bigtop's Apache Website

```
mvn site-deploy
```

This will deploy the generated website to /www/bigtop.apache.org/ on people.apache.org. You can SSH there to check that things look OK.Once the deployment is done you need to publish new website via https://cms.apache.org/bigtop/: login and click⚠ "Publish bigtop site" link. Otherwise the content won't get propagated.

It will take an hour or so for the website to become live, but you can inspect the website by temporarily setting your browser proxy to see the new, unmirrored content (I used FoxyProxy in Firefox) as described in http://www.apache.org/dev/project-site.html.

Remember to update download page accordingly.

As said in **BIGTOP-1162** - Please delete old releases from mirroring system RESOLVED , remove old releases to reduce the loading for apache mirror. The older releases are available in Apache archive server(older versions are archived automatically, see http://archive.apache.org/dist/bigtop/ and http://www.apache.org/dev/mirrors.html).

```
VERSION=X.Y.Z #Example: 1.2.1
svn co https://dist.apache.org/repos/dist/release/bigtop
cd bigtop
svn delete bigtop-$VERSION
svn commit
```

11. Announce the Release

TODO: Add a news section to the website.

Send an email to announce@apache.org (the from: address must be **@apache.org**). E.g.

```
To: announce@apache.org, user@bigtop.apache.org, dev@bigtop.apache.org
Subject: [ANNOUNCE] Apache Bigtop X.Y.Z released

The release is available here:
    https://bigtop.apache.org/download.html#releases

A few highlights of this release include:
    <LIST-OF-ITEMS-TO-BE-HIGHLIGHTED>


With Bigtop X.Y.Z the community continues to deliver the most advanced big data stack to date. More details
about X.Y.Z release are here:
    https://bigtop.apache.org/release-notes.html

Deploying Bigtop is easy: grab the repo/list file for your favorite Linux distribution:
  https://www.apache.org/dyn/closer.lua/bigtop/bigtop-X.Y.Z/repos/
and you'll be running your very own bigdata cluster in no time!

We welcome your help and feedback. For more information on how to report problems, and to get involved, visit
the project website at:
https://bigtop.apache.org
```

Please be noted that for download page it should be updated with following compliance:

- It should contain links to all current and archived releases along with links to KEYS, checksums, and signatures for all releases.
- The links on the page need to be links to release artifacts, not links to directories.
- The links to the asc and sha files should refer to archives only for non-current releases. The current release links should be to apache.org/dist, not archive.apache.org/dist.

12. Add the Next Release to JIRA

Add the next version number (e.g. 0.2.0 after 0.1.0) to JIRA using this link: https://issues.apache.org/jira/plugins/servlet/project-config/BIGTOP/administer-versions?status=unreleased

In JIRA mark the released version as "released" on the "manage versions" page. Be sure to fill in a date if not already specified.