# Bigtop CI Setup Guide

This document is for you to quickly setup CI jobs to ensure the quality of your own customized Bigtop distribution.

Notice that, the document assume you already know well how to use Jenkins, hence the instructions are brief.

If you have questions, feel free to ask on Bigtop mailing list(user@bigtop.apache.org). 🙂

Apache Bigtop is managing its own setup of CI system. We have 1 master + 4 slaves total. There's one 1T storage assigned to master, and another 2 * 800G go to two of the slaves (06 and 07), respectively. Other two nodes (02 and 03) got 200G for each node to operate therefore they are reserved to run non storage sensitive jobs such as provisioner.



## Setup a Jenkins master

If you already have a managed Jenkins/Hudson, or you prefer your all installation, skip and go to **Setup Bigtop packages build matrix**

To setup a Jenkins master by leveraging Docker, do the following:

```
# create jenkins user on host machine with uid=1000 to map the jenkins uid inside jenkins image
sudo adduser jenkins -u 1000
sudo yum install -y docker git
sudo su - jenkins -c "git config --global user.email \"jenkins@bigtop.apache.org\""
sudo su - jenkins -c "git config --global user.name \"jenkins\""
sudo usermod -a -G docker jenkins
sudo service docker start
docker run -d --name jenkins-master -p 8080:8080 -v /home/jenkins:/var/jenkins_home jenkins/jenkins:latest
```

And the needed Plugin(s):

- git plugin

## Setup Jenkins slaves

Assuming that you are going to add or replace an Amazon EC2 instance which have already been created with Amazon Linux 2 AMI, execute the following commands in that node:

```
// Install Docker and Docker Compose (execute as ec2-user)
$ sudo yum update -y
$ sudo yum install -y docker git java ruby
$ sudo amazon-linux-extras install -y docker  # to make sure the latest version of docker is installed
$ sudo service docker start
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -
m)" -o /usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose

// Create jenkins user and setup SSH auth so that Jenkins master can connect to this node
$ sudo adduser jenkins
$ sudo usermod -aG docker jenkins
$ sudo -i
# su - jenkins
$ mkdir .ssh
$ cat << EOF > .ssh/authorized_keys
> (MASTER_PUBLIC_KEY_HERE)
> EOF
$ chmod 700 .ssh
$ chmod 400 .ssh/authorized_keys

// Make sure Docker and Docker Compose work for jenkins user, and login to
// Docker Hub so that Docker-related Jenkins jobs can push generated images
$ docker run hello-world
$ docker-compose version
$ docker login --username bigtop
```

Then add the node or update its configuration (e.g., FQDN) via Jenkins. In the latter case, you may have to disconnect and relaunch agent so that Jenkins recognizes the change.

NOTE: Some AMI (e.g., Amazon Linux 2) uses 1000 as a UID for the default user, but it conflicts our Docker image, which assigns Jenkins to UID 1000, and causes permission issue when building packages inside the container. In this case, reassign Jenkins to 1000 and the default user to another UID.

# Setup Bigtop packages build matrix

Create a new job:

- New Item -> Multi-configuration project

In Source Code Management section, fill in your repo and the branch, for example:

```
Repository URL: https://gitbox.apache.org/repos/asf/bigtop.git
Branch Specifier: master
```

In the Configuration Matrix section, add a user defined axis with following name and values:

```
Name: BUILD_ENVIRONMENTS

Values:
centos-6
centos-7
fedora-25
ubuntu-16.04
debian-8
opensuse-42.1
```

Add another user defined axis:

```
Name: COMPONENTS

Values:
bigtop-groovy
bigtop-jsvc
bigtop-tomcat
bigtop-utils
crunch
datafu
flume
giraph
hadoop
hama
hbase
hive
hue
ignite-hadoop
kafka
kite
mahout
oozie
phoenix
pig
solr
spark
sqoop
sqoop2
tachyon
tez
ycsb
zookeeper
```

Finally, add a shell build step with the following script:

```
docker run --rm -v `pwd`:/ws --workdir /ws bigtop/slaves:trunk-$BUILD_ENVIRONMENTS \
bash -l -c './gradlew allclean $COMPONENTS-pkg'
```

The result will be looked like this:

https://ci.bigtop.apache.org/view/Packages/job/Bigtop-trunk-packages/

However, do aware that full matrix build of packages is time consuming, and they need roughly 50GB disk space for each build.

# Setup Bigtop deployments build matrix

To setup a deployments build matrix, prepare a job like above in packages build matrix and add these defined axis:

```
Name: OS

Values:
centos6
centos7
debian8
ubuntu_xenial
```

See the sample configs in bigtop/provisioner/docker.

```
Name: COMPONENTS

Values:
alluxio
apex
crunch
flink
flume
giraph
ignite_hadoop
hbase
hcat
hive
httpfs
hue
mahout
mapred-app
oozie
pig
qfs
solrcloud
spark
sqoop
sqoop2
tez
yarn
zookeeper
ycsb
gpdb
ambari
```

See the available components for deployment in bigtop-deploy/puppet/hieradata/site.yaml.

Use the following shell script for builds:

```
# destroy previous cluster
./gradlew docker-provisioner-destroy

# setup configuration file
CONFIG="config_${OS}_Bigtop-trunk-deployments.yaml"
sed "s/components.*/components: [hdfs, yarn, mapreduce, ${COMPONENTS}]/g" \
  provisioner/docker/config_${OS}.yaml > provisioner/docker/${CONFIG}


# provision
# Since the puppet deployment will always return 0,
# we need to save the log and grep error to determine status
./gradlew -Pconfig=${CONFIG} -Pnum_instances=1 docker-provisioner > tmp.log 2>&1
cat tmp.log


# destroy provisioned cluster
./gradlew docker-provisioner-destroy


# Fail the build if Errors occur in tmp.log
grep Error tmp.log && exit 1
exit 0
```

The result will be looked like this:

https://ci.bigtop.apache.org/view/Provisioner/job/Bigtop-1.2.1-deployments/

You can replace the repo by your own repo by adding sed commend so that your own packages can be tested using the same deployment recipes.

The components is also configurable, choose what you'd like to deploy and test as what you want.

# Setup Bigtop smoke tests build matrix

Create a new job:

- New Item -> Multi-configuration project

In Source Code Management section, fill in your repo and the branch, for example:

```
Repository URL: https://gitbox.apache.org/repos/asf/bigtop.git
Branch Specifier: master
```

In the Configuration Matrix section, add axises with following name and values:

User-defined axis:

```
Name: COMPONENTS

Values:
hdfs.alluxio@alluxio
ambari.bigtop-utils@ambari
hdfs.yarn.apex@apex
flume@flume
hdfs.hbase@hbase
hdfs.yarn@hdfs.hcfs
hdfs.yarn.hive@hive
hdfs.yarn.ignite_hadoop@ignite-hadoop
hdfs.yarn.mahout@mahout
hdfs.yarn@mapreduce
hdfs.yarn.qfs@qfs
hdfs.zookeeper.solrcloud@solr
hdfs.yarn.spark@spark
sqoop@sqoop
hdfs.yarn@yarn
zookeeper@zookeeper
```

For each value, the format is defined as DEPLOY_COMPONENTS@TEST_COMPONENETS, where DEPLOY_COMPONENTS cam be period separated string.

For example,

`hdfs.alluxio@alluxio`

Component hdfs and alluxio will be deployed, and alluxio will be tested.

Note: Why use period instead of comma as separator? Because Jenkins does not allow comma to be used in user defined axis.

Slaves aixs:

You got check boxes to check. To add options, go to **Manage Jenkins  Manage Nodes** and add labels for your slaves. The labels will be available here.

The following labels are bigtop's configuration for your reference.

```
Name: OS

Values:
centos-7-x86_64-deploy
fedora-26-x86_64-deploy
debian-9-amd64-deploy
ubuntu-16.04-amd64-deploy
debian-9-arm64-deploy
ubuntu-16.04-arm64-deploy
fedora-26-aarch64-deploy
```

In build section, add execute shell:

```
#!/bin/bash -x

# Setup configuration file
CONFIG="config_${OS}_Bigtop-smoke-tests.yaml"
DEPLOY_COMPONENTS=`echo ${COMPONENTS} | cut -d '@' -f 1 | sed "s/\\./,/g"`
TEST_COMPONENTS=`echo ${COMPONENTS} | cut -d '@' -f 2 | sed "s/\\./,/g"`
OS_FOR_URL=`echo ${OS} | awk -F"-" '{ print $1"-"$2"-"$3}' | sed "s@-@/@g"`

# Adjust mem based on test target
if [[ $DEPLOY_COMPONENTS == *"mahout"* ]]; then
  MEM=6g
fi

# The combination of DISTRO X ARCH is complicated. Resolved in the following logic
OS_WO_ARCH=`echo ${OS} | awk -F"-" '{ print $1"-"$2}'`
ARCH=`echo ${OS} | awk -F"-" '{ print $3}'`
case "${ARCH}" in
  aarch64|arm64)
    ARCH_SUFFIX="-aarch64"
    ;;
  ppc64le|ppc64el)
    ARCH_SUFFIX="-ppc64le"
    ;;
  x86_64|amd64)
    ARCH_SUFFIX=""
    ;;
  *)
    echo "Unsupported arch [${ARCH}]"
    exit 1
    ;;
esac

# Determine distro for config.yaml
case "${OS}" in
  centos-*|fedora-*|opensuse-*)
    DISTRO=centos
    ;;
  debian-*|ubuntu-*)
    DISTRO=debian
    ;;
  *)
    echo "Unsupported distro [${OS}]"
    exit 1
    ;;
esac

cat > provisioner/docker/${CONFIG} <<-__EOT__
docker:
        memory_limit: "${MEM:-4g}"
        image: "bigtop/puppet:trunk-${OS_WO_ARCH}"

# Package CI is currently broken
# repo: https://ci.bigtop.apache.org/view/Packages/job/Bigtop-trunk-repos/OS=${OS},label=docker-slave/ws/output
repo: http://repos.bigtop.apache.org/releases/1.3.0/${OS_FOR_URL}
distro: ${DISTRO}
components: [${DEPLOY_COMPONENTS}]
enable_local_repo: false
smoke_test_components: [${TEST_COMPONENTS}]
__EOT__

cat provisioner/docker/${CONFIG}

# Workaround that files written inside containers are own by root
docker run --rm -v ${PWD}:/bigtop-home "bigtop/puppet:trunk-${OS_WO_ARCH}${ARCH_SUFFIX}" bash -c "chown -R
${UID}:${UID} /bigtop-home"

# Provision
(cd provisioner/docker && ./docker-hadoop.sh -d -C ${CONFIG} -c 3 -s)
```

```
# Destroy provisinoed cluster
(cd provisioner/docker && ./docker-hadoop.sh -d)

# Workaround that files written inside containers are own by root
docker run --rm -v ${PWD}:/bigtop-home "bigtop/puppet:trunk-${OS_WO_ARCH}${ARCH_SUFFIX}" bash -c "chown -R
${UID}:${UID} /bigtop-home"
```

The result will be looked like this:

https://ci.bigtop.apache.org/view/Test/job/Bigtop-trunk-smoke-tests/

There're some limitations for the current implementation hence the script is a bit complicated.

1. Smoke tests running inside docker write files as root, hence files written by smoke tests can't be edited by Jenkins. Currently we work around this by spinning up a docker to chown files back to be owned by jenkins.
2. Provisioner should be able to decide which distro (centos or debian), however currently is externalized for user to configure it.

# Advanced part: Setup a SSL secured Jenkins master

First you need a SSL certificate.

## Generating a SSL cert

Here we will use a certificate created by the http://letsencrypt.org service.

Most convient way to run it is as a docker container, so pull the image:

```
# docker pull quay.io/letsencrypt/letsencrypt:latest
```

Now use this container to sign into letsencrypt. You have to make sure that no other service is running on port 80 and 443. This starts an intermediate server an handles a challenge response handshake in order to prove that we actually are running http://ci.bigtop.apache.org

```
# docker run --rm -i -t -p 80:80 -p 443:443  -v "/etc/letsencrypt:/etc/letsencrypt" quay.io/letsencrypt
/letsencrypt certonly --standalone --email dev@bigtop.apache.org -d ci.bigtop.apache.org
```

Please note that this writes into /etc/letsencrypt as root on the host.

## Generating first cert

The command does not generate a certificate on first run. It generates new certificate (renewals) each new run. Please be sure that any httpd server is stopped

```
# docker run --rm -i -t -p 80:80 -p 443:443  -v "/etc/letsencrypt:/etc/letsencrypt" quay.io/letsencrypt
/letsencrypt certonly --standalone --email dev@bigtop.apache.org -d ci.bigtop.apache.org
```

Please note it created  /etc/letsencrypt/live/ci.bigtop.apache.org containing cert and key.

## Enabling of SSL

Since the lifetime of the letsencrypt certs is quite short on needs a flexible infrastructure to use the certs as is.

The basic design here is to use a reverse proxy terminating the SSL at the proxy. I.e. All handling of https:// is done by the reverse proxy and requests are proxied to the web application and sent back to the client.

A quick setup on a RPM based system is:

```
# yum install httpd mod_ssl
```

in /etc/http/conf.d/ssl.conf add this block after the line

```
<VirtualHost _default_:443>
```

(see jenkins documentation about deeper insights into this glibberish)

```
<Proxy *>
        Order deny,allow
        Allow from all
</Proxy>

ProxyRequests           Off

# Because of JENKINS-22539
ProxyPreserveHost On
Header edit Location ^http://ci.bigtop.apache.org/ https://ci.bigtop.apache.org/
ProxyPass         /  http://localhost:8080/ nocanon
ProxyPassReverse  /  http://localhost:8080/
ProxyRequests     Off

AllowEncodedSlashes NoDecode

<Proxy http://localhost:8080/*>
  Order deny,allow
  Allow from all
</Proxy>
```

This enables the reverse proxy mode of port 433 to port 8080 and setting Jenkins specific parameters.

And the location of the certificates have to be aligned with letsencrypt in /etc/http/conf.d/ssl.conf

```
#   Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate.  If
# the certificate is encrypted, then you will be prompted for a
# pass phrase.  Note that a kill -HUP will prompt again.  A new
# certificate can be generated using the genkey(1) command.

SSLCertificateFile /etc/letsencrypt/live/ci.bigtop.apache.org/cert.pem

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)

SSLCertificateKeyFile  /etc/letsencrypt/live/ci.bigtop.apache.org/privkey.pem

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convinience.

SSLCertificateChainFile /etc/letsencrypt/live/ci.bigtop.apache.org/chain.pem
```

The last thing is to change jenkins to port 8080 and start apache httpd. Please note, we are running the latest Jenkins LTS to keep up with the security updates, etc.

```
# docker run -d --name jenkins-master-2.6 -p 8080:8080 -v /home/jenkins:/var/jenkins_home jenkins/jenkins:lts
# systemctl start httpd
```

In order to redirect the browser from http://ci.bigtop.apache.org to https://ci.bigtop.apache.org place a file into /var/www/html/index.html

```
<META HTTP-EQUIV="refresh" CONTENT="1; URL=https://ci.bigtop.apache.org">
```

## Renewing the cert

The command does not generate a certificate on first run. It generates new certificate (renewals) each new run. Please be sure that any httpd server is stopped

```
# service httpd stop
# docker stop jenkins-master
# docker run --rm -i -t -p 80:80 -p 443:443  -v "/etc/letsencrypt:/etc/letsencrypt" certbot/certbot renew
# docker start jenkins-master
# service httpd start
```