# Running Hadoop on Amazon EC2

Amazon EC2 (Elastic Compute Cloud) is a computing service. One allocates a set of hosts, and runs one's application on them, then, when done, de-allocates the hosts. Billing is hourly per host. Thus EC2 permits one to deploy Hadoop on a cluster without having to own and operate that cluster, but rather renting it on an hourly basis. If you run Hadoop on EC2 you might consider using AmazonS3 for accessing job data (data transfer to and from S3 from EC2 instances is free). Initial input can be read from S3 when a cluster is launched, and the final output can be written back to S3 before the cluster is decomissioned. Intermediate, temporary data, only needed between MapReduce passes, is more efficiently stored in Hadoop's DFS. See AmazonS3 for more details.

This document assumes that you have already followed the steps in Amazon's Getting Started Guide. In particular, you should have run through the sections "Setting up an Account", "Setting up the Tools" and the "Generating a Keypair" section of "Running an Instance".

Note that the older, manual step-by-step guide to getting Hadoop running on EC2 can be found here.

**Version 0.17** of Hadoop includes a few changes that provide support for multiple simultaneous clusters, quicker startup times for large clusters, and includes a pre-configured Ganglia installation. These differences are noted below.

**Note:** Cloudera also provides their distribution for Hadoop as an EC2 AMI with single command deployment and support for Hive/Pig out of the box.

# Preliminaries

## Concepts

- **Amazon Machine Image (AMI)**, or *image*. A bootable Linux image, with software pre-installed. There are some public Hadoop AMIs that have everything you need to run Hadoop in a cluster.
- **instance**. A host running an AMI.

## Conventions

In this document, commands lines that start with '#' are executed on an Amazon instance, while command lines starting with a '%' are executed on your workstation.

## Security

Clusters of Hadoop instances are created in a security group. Instances within the group have unfettered access to one another. Machines outside the group (such as your workstation), can only access instance on port 22 (for SSH), port 50030 (for the JobTracker's web interface, permitting one to view job status), and port 50060 (for the TaskTracker's web interface, for more detailed debugging).

# Setting up

- Unpack the latest Hadoop distribution on your system
- Edit all relevant variables in *src/contrib/ec2/bin/hadoop-ec2-env.sh*.
    - Amazon Web Services variables (`AWS_ACCOUNT_ID`, `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`)
        - All need filling in - they can be found by logging in to http://aws.amazon.com/.
        - `AWS_ACCOUNT_ID` is your 12 digit account number.
    - Security variables (`EC2_KEYDIR`, `KEY_NAME`, `PRIVATE_KEY_PATH`, `SSH_`

OPTS)
    - The defaults should be OK if you followed Amazon Getting Started guide,
      except `PRIVATE_KEY_PATH` which needs changing if you don't store this with your other EC2 keys.
  - AMI selection (`HADOOP_VERSION`, `S3_BUCKET`)
    - These two variables control which AMI is used.
    - To see which versions are publicly available type:

```
% ec2-describe-images -x all | grep hadoop
```

    - The default value for `S3_BUCKET` (`hadoop-ec2-images`) is for public images. Images for Hadoop version 0.17.1 and later are in the `hadoop-images` bucket, so you should change this variable if you want to use one of these images. You also need to change this if you want to use a private image you have built yourself.

# Running a job on a cluster

- Open a command prompt in *src/contrib/ec2*.
- Launch a EC2 cluster and start Hadoop with the following command. You must supply a cluster name (test-cluster) and the number of slaves (2). After the cluster boots, the public DNS name will be printed to the console.

```
% bin/hadoop-ec2 launch-cluster test-cluster 2
```

- You can login to the master node from your workstation by typing:

```
% bin/hadoop-ec2 login test-cluster
```

- You will then be logged into the master node where you can start your job.
    - For example, to test your cluster, try

```
# cd /usr/local/hadoop-*
# bin/hadoop jar hadoop-*-examples.jar
pi 10 10000000
```

- You can check progress of your job at `http://<MASTER_HOST>:50030/`. Where MASTER_HOST is the host name returned after the cluster started, above.
- When you have finished, shutdown the cluster with the following:

```
% bin/hadoop-ec2 terminate-cluster test-
cluster
```

- Keep in mind that the master node is started first and configured, then all slaves nodes are booted simultaneously with boot parameters pointing to the master node. Even though the `launch-cluster` command has returned, the whole cluster may not have yet 'booted'. You should monitor the cluster via port 50030 to make sure all nodes are up.

## Running a job on a cluster from a remote machine

In some cases it's desirable to be able to submit a job to a hadoop cluster running in EC2 from a machine that's outside EC2 (for example a personal workstation). Similarly - it's convenient to be able to browse/cat files in HDFS from a remote machine. One of the advantages of this setup is that it obviates the need to create

custom AMIs that bundle stock Hadoop AMIs and user libraries/code. All the non-Hadoop code can be kept on the remote machine and can be made available to Hadoop during job submission time (in the form of jar files and other files that are copied into Hadoop's distributed cache). The only downside being the cost of copying these data sets into EC2 and the latency involved in doing so.

The recipe for doing this is well documented in this Cloudera blog post and involves configuring hadoop to use a ssh tunnel through the master hadoop node. In addition - this recipe only works when using EC2 scripts from versions of Hadoop that have the fix for HADOOP-5839 incorporated. (Alternatively, users can apply patches from this JIRA to older versions of Hadoop that do not have this fix).

# Troubleshooting

Running Hadoop on EC2 involves a high level of configuration, so it can take a few goes to get the system working for your particular set up.

If you are having problems with the Hadoop EC2 `launch-cluster` command then you can run the following in turn, which have the same effect but may help you to see where the problem is occurring:

```
% bin/hadoop-ec2 launch-master <cluster-name>
% bin/hadoop-ec2 launch-slaves <cluster-name>
<num slaves>
```

Note you can call the `launch-slaves` command as many times as necessary to grow your cluster. Shrinking a cluster is more tricky and should be done by hand (after balancing file replications etc).

To browse all your nodes via a web browser, starting at the 50030 status page, start the following command in a new shell window:

```
% bin/hadoop-ec2 proxy <cluster-name>
```

This command will start a SOCKS tunnel through your master node, and print out all the URLs you can reach from you web browser. For this to work, you must configure your browser to send requests over SOCKS to the local proxy on port 6666.
The FireFox plugin FoxyProxy is great for this.
Currently, the scripts don't have much in the way of error detection or handling. If a script produces an error, then you may need to use the Amazon EC2 tools for interacting with instances directly - for example, to shutdown an instance that is mis-configured.
Another technique for debugging is to manually run the scripts line-by-line until the error occurs. If you have feedback or suggestions, or need help then please use the Hadoop mailing lists.
If you are finding that all your nodes are not showing up, you can point your browser to the Ganglia status page for your cluster at `http://<MASTER_HOST>/ganglia/`, after starting the `proxy`command.

# Building your own Hadoop image

The public images should be sufficient for most needs, however there are circumstances where you would like to build your own images, perhaps because an image with the version of Hadoop you want isn't available (an older version, the latest trunk version, or a patched version), or because you want to run extra software on your instances.

## Design

Here is a high-level outline of how the scripts for creating a Hadoop AMI work. For details, please see the scripts' sources (linked to below).

1. The main script, create-hadoop-image starts a Fedora core Amazon AMI.
2. Once the Fedora instance has launched *create-hadoop-*

*image* copies the environment variables file (hadoop-ec2-env.sh) and scripts to run on the Fedora instance (create-hadoop-image-remote and hadoop-init) then it logs into the Fedora instance and runs *create-hadoop-image-remote*.

3. The script *create-hadoop-image-remote* then installs Java, tools required to run Hadoop, and Hadoop itself. Then it configures Hadoop:
    - In EC2, the local data volume is mounted as */mnt*, so logs are written under here.
    - *hadoop-init* is installed as an init script to be run on instance start up. This takes advantage of an EC2 feature called parameterized launches. For Hadoop, this allows the master hostname and the cluster size to be retrieved when the Hadoop instance starts - this information is used to finish the Hadoop configuration.
4. Finally, *create-hadoop-image-remote* bundles the machine as an AMI, and uploads it to S3. (Particular care has to be taken to ensure that no secrets, such as private keys, are bundled in the AMI. See here for more details.) The AMI is stored in a bucket named by the variable `$S3_BUCKET` and with the name `hadoop-$HADOOP_VERSION`.
5. Control then returns to *create-hadoop-image* which registers the image with EC2.

## Building a stock Hadoop image

- Edit all relevant variables in *src/contrib/ec2/bin/hadoop-ec2-env.sh*.
    - AMI selection (`HADOOP_VERSION`, `S3_BUCKET`)
        - When creating an AMI, `HADOOP_VERSION` is used to select which version of Hadoop to download and install from http://hadoop.apache.org/common/releases.html.
        - Change `S3_BUCKET` to be a bucket you own that

you want to store the Hadoop AMI in.
    - (**0.17**) AMI size selection (`INSTANCE_TYPE`)
        - When creating an AMI, `INSTANCE_TYPE` denotes the instance size the image will be run on (small, large, or xlarge). Ultimately this decides if the image is `i386` or `x86_64`, so this value is also used on cluster startup.
    - Java variables
        - `JAVA_BINARY_URL` is the download URL for a Sun JDK. Visit the Oracle Java downloads page, select a stable JDK, and get the URL for the JDK (not JRE) labelled "Linux self-extracting file".
        - `JAVA_VERSION` is the version number of the JDK to be installed.
    - All other variables should be set as above.
- Type

```
% bin/hadoop-ec2 create-image
```

- Accept the Java license terms.
- The script will create a new image, then bundle, upload and register it. This may take some time (20 minutes or more). Be patient - don't assume it's crashed!
- Terminate your instance using the command given by the script.

If you need to repeat this procedure to re-create an AMI then you will need to run `ec2-deregister` to de-register the existing AMI. You might also want to use `ec2-delete-bundle` command to remove the AMI from S3 if you no longer need it.

## Building a customized Hadoop image

If you want to build an image with a version of Hadoop that is not available from the Apache distribution site (e.g. trunk, or a patched version) then you will need to alter the *create-hadoop-image-*

*remote* script to retrieve and install your required version of Hadoop. Similarly, if you wish to install other software on your image then the same script is the place to do it.

### Making an image public

Since there are already public Hadoop AMIs available you shouldn't need to do this. (At least consider discussing it on the developer mailing list first, please.) Furthermore, you should only do this if you are sure you have produced a secure AMI.

```
% ec2-modify-image-attribute AMI -l -a all
```

where `AMI` is the ID of the AMI you want to publish.
See Introduction to Sharing AMIs for more details.

## Resources

- Amazon EC2 Homepage, Getting Started Guide, Developer Guide, Articles and Tutorials.
- AWS blog
- Running Hadoop MapReduce on Amazon EC2 and Amazon S3 by Tom White, Amazon Web Services Developer Connection, July 2007
- Notes on Using EC2 and S3 Details on FoxyProxy setup, and other things to watch out for.