UnknownHost

Unknown Host

You get an Unknown Host Error -often wrapped in a Java `IOException`, when one machine on the network cannot determine the IP address of a host that it is trying to connect to by way of its hostname. This can happen during file upload (in which case the client machine is has the hostname problem), or inside the Hadoop cluster.

Some possible causes in approximately reverse order of likelihood (not an exclusive list):

1. DNS or hosts table misconfiguration:
    a. You are using DNS but the site's DNS server does not have an entry for the node. **Test:** do an `nslookup <hostname>` from the client machine.
    b. You are using `/etc/hosts` entries but the calling machine's hosts file lacks an entry for the host. FQDN entries in `/etc/hosts` files must contain a trailing dot. See the Using Hosts files section below. **Test:** do a `ping <hostname>.` from the client machine (note trailing 'dot').
    c. The hostname in the configuration files (such as `core-site.xml`) is misspelled.
2. The hostname in the configuration files (such as `core-site.xml`) is confused with the hostname of another service. For example, you are using the hostname of the YARN Resource Manager in the `fs.defaultFS` configuration option to define the namenode.
3. A worker node thinks it has a given name which it reports to the NameNode and JobTracker, but that isn't the name that the network team gave it, so it isn't resolvable.
4. If it is happening in service startup, it means the hostname of that service (HDFS, YARN, etc) cannot be found in `/etc/hosts`; the service will fail to start as it cannot determine which network card/address to use.
5. The calling machine is on a different subnet from the target machine, and short names are being used instead of fully qualified domain names (FQDNs).
6. You are running in a cloud infrastructure and the destination machine is no longer there. It may have been deleted from the DNS records, or, due to some race condition, something is trying to talk to a host that hasn't been created yet.

Less likely causes:

1. The client's network card is playing up (network timeouts, etc), the network is overloaded, or even the switch is dropping DNS packets.
2. The host's IP address has changed but a long-lived JVM is caching the old value. This is a known problem with JVMs (search for "java negative DNS caching" for the details and solutions). The quick solution: restart the JVMs
3. The site's DNS server is overloaded. This can happen in large clusters. Either move to host table entries or use caching DNS servers in every worker node.
4. Your ARP cache is corrupt, either accidentally or maliciously. If you don't know what that means, you won't be in a position to verify this is the problem -or fix it.

These are all network configuration/router issues. As it is your network, only you can find out and track down the problem. That said, any tooling to help Hadoop track down such problems in cluster would be welcome, as would extra diagnostics. If you have to extend Hadoop to track down these issues - submit your patches!

Some tactics to help solve the problem:

1. Look for configuration problems first (Hadoop XML files, hostnames, host tables), as these are easiest to fix and quite common.
2. Try and identify which client machine is playing up. If it is out-of-cluster, try the FQDN instead, and consider that it may not have access to the worker node.
3. If the client that does not work is one of the machines in the cluster, SSH to that machine and make sure it can resolve the hostname.
4. As well as `nslookup`, the `dig` command is invaluable for tracking down DNS problems, though it does assume you understand DNS records. Now is a good time to learn.
5. Restart the JVMs to see if that makes it go away.
6. Restart the servers to see if that makes it go away.
7. Reboot the network switches.

Unless the route cause has been identified, the problem may return.

Using hosts files

If you are using `/etc/hosts` files instead of DNS-based lookups and your hosts files have FQDNs, you must ensure that the FQDN includes a trailing dot. Thus a correct hosts file entry for `foo.example.com` may look like this.

```
1.2.3.4  foo.example.com foo.example.com. foo
```

The Hadoop host resolver ensures hostnames are terminated with a trailing dot prior to lookup to avoid the security issue described in RFC 1535.

Unknown Host Exception in HA HDFS

This exception surfaces when setting up HA HDFS.

As documented, HA HDFS requires you to list the namenode URLs of a cluster in the property `edfs.ha.namenodes.mycluster`, where "mycluster" is the name of your HA cluster.

```
<property>
  <name>dfs.ha.namenodes.mycluster</name>
  <value>nn1,nn2</value>
</property>
```

Then for the filesystem URL, you use the name of the cluster:

```
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://mycluster</value>
</property>
```

If you get an Unknown Host Exception, and the host is the name of your HA cluster, here `mycluster`, then it means that the HDFS client hasn't recognized that this is an HA cluster, and instead tried to connect to it directly on the default HDFS port.

The dfs.ha.namenodes.mycluster property is unset or the cluster name is inconsistent across the properties. Check your config and try again.

Finally, because this is a configuration problem, filing bug reports is not going to help. They will only be closed as Invalid Issues