HBaseFsck (hbck) is a tool for checking for **region consistency** and **table integrity** problems and repairing a corrupted HBase. It works in two basic modes -- a read-only inconsistency identifying mode and a multi-phase read-write repair mode.

Running hbck to identify inconsistencies

To check to see if your HBase cluster has corruptions, run hbck against your HBase cluster:

```
$ ./bin/hbase hbck
```

At the end of the commands output it prints *OK* or tells you the number of *INCONSISTENCIES* present. You may also want to run run **hbck** a few times because some inconsistencies can be transient (e.g. cluster is starting up or a region is splitting). Operationally you may want to run hbck regularly and setup alert (e.g. via nagios) if it repeatedly reports inconsistencies .

A run of hbck will report a list of inconsistencies along with a brief description of the regions and tables affected. The using the **-details** option will report more details including a representative listing of all the splits present in all the tables.

```
$ ./bin/hbase hbck -details
```

Inconsistencies

If after several runs, inconsistencies continue to be reported, you may have encountered a corruption. These should be rare, but in the event they occur newer versions of HBase include the hbck tool enabled with automatic repair options.

There are two invariants that when violated create inconsistencies in HBase:

- HBase's **region consistency** invariant is satisfied if every region is assigned and deployed on exactly one region server, and all places where this state kept is in accordance.
- HBase's **table integrity** invariant is satisfied if for each table, every possible row key resolves to exactly one region.

Repairs generally work in three phases -- a read-only information gathering phase that identifies inconsistencies, a table integrity repair phase that restores the table integrity invariant, and then finally a region consistency repair phase that restores the region consistency invariant.

Starting from version 0.90.0, hbck could detect region consistency problems report on a subset of possible table integrity problems. It also included the ability to automatically fix the most common inconsistency, region assignment and deployment consistency problems. This repair

could be done by using the **-fix** command line option. These problems close regions if they are open on the wrong server or on multiple region servers and also assigns regions to region servers if they are not open.

Starting from HBase versions 0.90.7, 0.92.2 and 0.94.0, several new command line options are introduced to aid repairing a corrupted HBase. This hbck sometimes goes by the nickname "uberhbck". Each particular version of uber hbck is compatible with the HBase's of the same major version (0.90.7 uberhbck can repair a 0.90.4). However, versions <=0.90.6 and versions <=0.92.1 may require restarting the master or failing over to a backup master.

Localized repairs

When repairing a corrupted HBase, it is best to repair the lowest risk inconsistencies first. These are generally region consistency repairs -- localized single region repairs, that only modify in-memory data, ephemeral zookeeper data, or patch holes in the META table.

Region consistency requires that the HBase instance has the state of the region's data in HDFS (.regioninfo files), the region's row in the .META. table., and region's deployment/assignments on region servers and the master in accordance. Options for repairing region consistency include:

- **-fixAssignments** (equivalent to the 0.90 **-fix** option) repairs unassigned, incorrectly assigned or multiply assigned regions.
- **-fixMeta** which removes meta rows when corresponding regions are not present in HDFS and adds new meta rows if they regions are present in HDFS while not in META.

To fix deployment and assignment problems you can run this command:
$ ./bin/hbase hbck -fixAssignments

To fix deployment and assignment problems as well as repairing incorrect meta rows you can run this command:.
$ ./bin/hbase hbck -fixAssignments -fixMeta

There are a few classes of table integrity problems that are low risk repairs. The first two are degenerate (startkey == endkey) regions and backwards regions (startkey > endkey). These are automatically handled by sidelining the data to a temporary directory (/hbck/xxxx).

The third low-risk class is hdfs region holes. This can be repaired by using the:

- **-fixHdfsHoles** option for fabricating new empty regions on the file system.

If holes are detected you can use -fixHdfsHoles and should include -fixMeta and -fixAssignments to make the new region consistent.
$ ./bin/hbase hbck -fixAssignments -fixMeta -fixHdfsHoles

Since this is a common operation, we've added a the **-repairHoles** flag that is equivalent to the previous command:
$ ./bin/hbase hbck -repairHoles

If inconsistencies still remain after these steps, you most likely have table integrity problems related to orphaned or overlapping regions.

Region Overlap Repairs

Table integrity problems can require repairs that deal with overlaps. This is a riskier operation because it requires modifications to the file system, requires some decision making, and may require some manual steps. For these repairs it is best to analyze the output of a **hbck -details** run so that you isolate repairs attempts only upon problems the checks identify. Because this is riskier, there are safeguard that should be used to limit the scope of the repairs.

WARNING: This is a relatively new and have only been tested on online but idle HBase instances (no reads/writes). Use at your own risk in an active production environment!

The options for repairing table integrity violations include:

- **-fixHdfsOrphans** option for "adopting" a region directory that is missing a region metadata file (the .regioninfo file).
- **-fixHdfsOverlaps** ability for fixing overlapping regions

When repairing overlapping regions, a region's data can be modified on the file system in two ways: 1) by *merging* regions into a larger region or 2) by *sidelining* regions by moving data to "sideline" directory where data could be restored later. Merging a large number of regions is technically correct but could result in an extremely large region that requires series of costly compactions and splitting operations. In these cases, it is probably better to sideline the regions that overlap with the most other regions (likely the largest ranges) so that merges can happen on a more reasonable scale. Since these sidelined regions are already laid out in HBase's native directory and HFile format, they can be restored by using HBase's bulk load mechanism.

The default safeguard thresholds are conservative. These options let you override the default thresholds and to enable the large region sidelining feature.

- **-maxMerge <n>** maximum number of overlapping regions to merge
- **-sidelineBigOverlaps** if more than maxMerge regions are overlapping, sideline attempt to sideline the regions overlapping with the most other regions.
- **-maxOverlapsToSideline <n>** if sidelining large overlapping regions, sideline at most n regions.

Since often times you would just want to get the tables repaired, you can use this option to turn on all repair options:

- **-repair** includes all the region consistency options and only the hole repairing table integrity options.

Finally, there are safeguards to limit repairs to only specific tables. For example the following command would only attempt to repair table TableFoo and TableBar.

$ ./bin/hbase/ hbck -repair TableFoo TableBar

Special cases: Meta is not properly assigned

There are a few special cases that hbck can handle as well.

Sometimes the meta table's only region is inconsistently assigned or deployed. In this case there is a special **-fixMetaOnly** option that can try to fix meta assignments.

$ ./bin/hbase hbck -fixMetaOnly -fixAssignments

Special cases: HBase version file is missing.

HBase's data on the file system requires a version file in order to start. If this flie is missing, you can use the **-fixVersionFile** option to fabricating a new HBase version file. This assumes that the version of hbck you are running is the appropriate version for the HBase cluster.

Special case: Root and META are corrupt.

The most drastic corruption scenario is the case where the ROOT or META is corrupted and HBase will not start. In this case you can use the **OfflineMetaRepair** tool create new ROOT and META regions and tables.

This tool assumes that HBase is offline. It then marches through the existing HBase home directory, loads as much information from region metadata files (.regioninfo files) as possible from the file system. If the region metadata has proper table integrity, it sidelines the original root and meta table directories, and builds new ones with pointers to the region directories and their data.

$ ./bin/hbase org.apache.hadoop.hbase.util.OfflineMetaRepair

NOTE: This tool is not as clever as uberhbck but can be used to bootstrap repairs that uberhbck can complete.

If the tool succeeds you should be able to start hbase and run online repairs if necessary.