status

```
hbase(main):006:0> status
1 servers, 0 dead, 5.0000 average load
hbase(main):002:0> help 'status'
hbase> status
hbase> status 'simple'
hbase> status 'summary'
hbase> status 'detailed'
hbase> status 'replication'
hbase> status 'replication', 'source'
hbase> status 'replication', 'sink'
```

whoami

```
hbase(main):011:0> whoami
datanode1 (auth:SIMPLE)
groups: datanode1
```

alter / alter_async

```
#   t1   t1  f1     f1        5.
#   t1  f1      f1     t1     f1       5.
hbase> alter 't1', NAME => 'f1', VERSIONS => 5
#
hbase> alter 't1', 'f1', {NAME => 'f2', IN_MEMORY => true}, {NAME =>
'f3', VERSIONS => 5}
#       ns1     t1     f1
hbase> alter 'ns1:t1', NAME => 'f1', METHOD => 'delete'
hbase> alter 'ns1:t1', 'delete' => 'f1'
#   t1 MAX_FILESIZE
hbase> alter 't1', MAX_FILESIZE => '134217728'
#    t1    f2
hbase> alter 't1', CONFIGURATION =>
{'hbase.hregion.scan.loadColumnFamiliesOnDemand' => 'true'}
hbase> alter 't1', {NAME => 'f2', CONFIGURATION =>
{'hbase.hstore.blockingStoreFiles' => '10'}}
#
hbase> alter 't1', METHOD => 'table_att_unset', NAME =>
'MAX_FILESIZE'
hbase> alter 't1', METHOD => 'table_att_unset', NAME =>
'coprocessor$1'
#
hbase> alter 't1', { NAME => 'f1', VERSIONS => 3 },
{ MAX_FILESIZE => '134217728' }, { METHOD => 'delete', NAME => 'f2'
},
OWNER => 'johndoe', METADATA => { 'mykey' => 'myvalue' }
hbase(main):014:0>
```

scan

```
#       hbase  meta     meta
hbase> scan 'hbase:meta'
#       hbase  meta    info  regioninfo    meta    info  regioninfo
hbase> scan 'hbase:meta', {COLUMNS => 'info:regioninfo'}
#       ns1  t1    'c1' 'c2'        ns1  t1    'c1' 'c2'
hbase> scan 'ns1:t1', {COLUMNS => ['c1', 'c2']}
#       ns1  t1    'c1' 'c2'        ns1  t1    'c1' 'c2'      10 rowkey
hbase> scan 'ns1:t1', {COLUMNS => ['c1', 'c2'], LIMIT => 10}
#       ns1  t1    'c1' 'c2'        ns1  t1    'c1' 'c2'      rowkey="xyz"    10 rowkey
hbase> scan 'ns1:t1', {COLUMNS => ['c1', 'c2'], LIMIT => 10,
STARTROW => 'xyz'}
#          t1   c1    '1303668804' '1303668904'
hbase> scan 't1', {COLUMNS => 'c1', TIMERANGE => [1303668804,
1303668904]}
#      t1
hbase> scan 't1', {REVERSED => true}
#      t1
hbase> scan 't1', {FILTER => "(PrefixFilter ('row2') AND
(QualifierFilter (>=, 'binary:xyz'))) AND (TimestampsFilter ( 123, 456))"}
# RAW true     t1
hbase> scan 't1', {RAW => true, VERSIONS => 10}
#  t1
hbase> t11 = get_table 't1'
hbase> t11.scan
```

get

```
#      ns1  t1 rowkey r1
hbase> get 'ns1:t1', 'r1'
#          t1 rowkey r1
hbase> get 't1', 'r1'
#          t1 rowkey r1       ts1 ts2
hbase> get 't1', 'r1', {TIMERANGE => [ts1, ts2]}
#          t1 rowkey r1 c1
hbase> get 't1', 'r1', {COLUMN => 'c1'}
#          t1 rowkey r1 c1,c2,c3
hbase> get 't1', 'r1', {COLUMN => ['c1', 'c2', 'c3']}
#          t1 rowkey r1 c1      ts1
hbase> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}
#          t1 rowkey r1 c1        ts1 ts2     4
```

get (cont)

```
hbase> get 't1', 'r1', {COLUMN => 'c1', TIMERANGE => [ts1, ts2],
VERSIONS => 4}
#
hbase> t.get 'r1'
hbase> t.get 'r1', {TIMERANGE => [ts1, ts2]}
hbase> t.get 'r1', {COLUMN => 'c1'}
hbase> t.get 'r1', {COLUMN => ['c1', 'c2', 'c3']}
hbase> t.get 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}
hbase> t.get 'r1', {COLUMN => 'c1', TIMERANGE => [ts1, ts2],
VERSIONS => 4}
hbase> t.get 'r1', {COLUMN => 'c1', TIMESTAMP => ts1, VERSIONS =>
4}
```

put

```
#      ns1  t1 rowkey r1  c1
hbase> put 'ns1:t1', 'r1', 'c1', 'value'
#          t1 rowkey r1  c1
hbase> put 't1', 'r1', 'c1', 'value'
#          t1 rowkey r1  c1            ts1
hbase> put 't1', 'r1', 'c1', 'value', ts1
#          t1 rowkey r1  c1            ts1
hbase> put 't1', 'r1', 'c1', 'value', ts1, {ATTRIBUTES=>
{'mykey'=>'myvalue'}}
#
t.put 'r1', 'c1', 'value', ts1, {ATTRIBUTES=>{'mykey'=>'myvalue'}}
```

version

```
hbase(main):010:0> version
1.0.3, rf1e1312f9790a7c40f6a4b5a1bab2ea1dd559890, Tue Jan 19
19:26:53 PST 2016
```

create

```
#     ns1     t1        f1 f1     5
hbase> create 'ns1:t1', {NAME => 'f1', VERSIONS => 5}
#           t1      f1,f2,f3
hbase> create 't1', {NAME => 'f1'}, {NAME => 'f2'}, {NAME => 'f3'}
#
hbase> create 't1', 'f1', 'f2', 'f3'
#   t1   f1    f1     1   TTL 2592000   BLOCKCACHE true
hbase> create 't1', {NAME => 'f1', VERSIONS => 1, TTL => 2592000,
BLOCKCACHE => true}
#    t1,  f1    f1   hbase.hstore.blockingStoreFiles   10
hbase> create 't1', {NAME => 'f1', CONFIGURATION =>
{'hbase.hstore.blockingStoreFiles' => '10'}}
#
hbase> create 'ns1:t1', 'f1', SPLITS => ['10', '20', '30', '40']
hbase> create 't1', 'f1', SPLITS => ['10', '20', '30', '40']
hbase> create 't1', 'f1', SPLITS_FILE => 'splits.txt', OWNER => 'johndoe'
```

create (cont)

```
hbase> create 't1', {NAME => 'f1', VERSIONS => 5}, METADATA => {
'mykey' => 'myvalue' }
hbase> # Optionally pre-split the table into NUMREGIONS, using
hbase> # SPLITALGO ("HexStringSplit", "UniformSplit" or classname)
#  Pre-splitting region
hbase> create 't1', 'f1', {NUMREGIONS => 15, SPLITALGO =>
'HexStringSplit'}
hbase> create 't1', 'f1', {NUMREGIONS => 15, SPLITALGO =>
'HexStringSplit', REGION_REPLICATION => 2, CONFIGURATION =>
{'hbase.hregion.scan.loadColumnFamiliesOnDemand' => 'true'}}
#        t2          t1 t1   t2          f1
hbase> t1 = create 't2', 'f1'
```

describe / desc

```
#       t1
hbase> describe 't3'
#
Table t3 is ENABLED
t3
COLUMN FAMILIES DESCRIPTION
{NAME => 'colfa', BLOOMFILTER => 'ROW', VERSIONS => '1',
IN_MEMORY => 'false', KEEP
_DELETED_CELLS => 'false', DATA_BLOCK_ENCODING => 'NONE',
COMPRESSION => 'NONE', TT
L => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => 'true',
BLOCKSIZE => '65536', RE
PLICATION_SCOPE => '0'}
1 row(s) in 0.0200 seconds
```

get_table

```
#   t1       t1d
hbase> t1d = get_table 't1'
#t1d
t1d.scan
t1d.describe
```

list

```
#
hbase> list
#     abc
hbase> list 'abc.*'
#       ns      abc
hbase> list 'ns:abc.*'
#       ns
hbase> list 'ns:.*'
```

show_filters

append

```
#   t1 rowkey r1  c1         value
hbase> append 't1', 'r1', 'c1', 'value'
# t1     t11  append
hbase> t11.append 'r1', 'c1', 'value'
```

count

```
#       ns1   t1 rowkey r1  c1     ts1
hbase> delete 'ns1:t1', 'r1', 'c1', ts1
#           t1 rowkey r1  c1     ts1
hbase> delete 't1', 'r1', 'c1', ts1
#
hbase> t.delete 'r1', 'c1', ts1
```

delete / deleteall

```
#       ns1   t1 rowkey r1  c1     ts1
hbase> delete 'ns1:t1', 'r1', 'c1', ts1
#           t1 rowkey r1  c1     ts1
hbase> delete 't1', 'r1', 'c1', ts1
#
hbase> t.delete 'r1', 'c1', ts1
#      ns1  t1 rowkey r1
hbase> deleteall 'ns1:t1', 'r1'
#          t1 rowkey r1
hbase> deleteall 't1', 'r1'
#      ns1  t1 rowkey r1  c1
hbase> deleteall 't1', 'r1', 'c1'
#           t1 rowkey r1  c1     ts1
hbase> deleteall 't1', 'r1', 'c1', ts1
#
hbase> t.deleteall 'r1'
hbase> t.deleteall 'r1', 'c1'
hbase> t.deleteall 'r1', 'c1', ts1
```

truncate

```
#   t3   disable
truncate 't3'
```