Vim

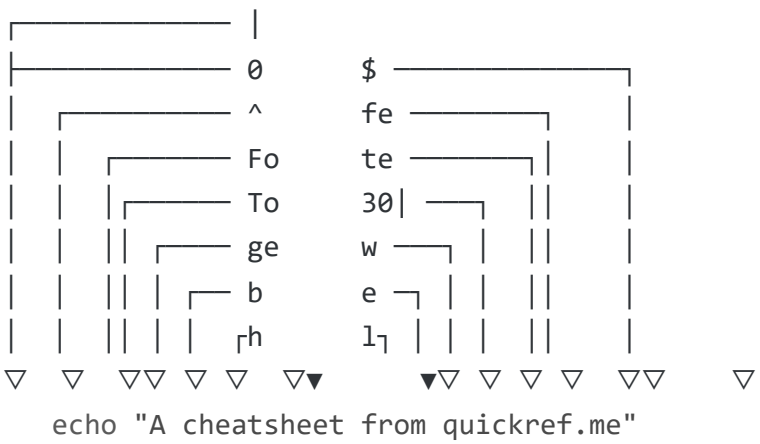
A useful collection of Vim 8.2 quick reference cheat sheets to help you learn vim editor faster.

Getting Started

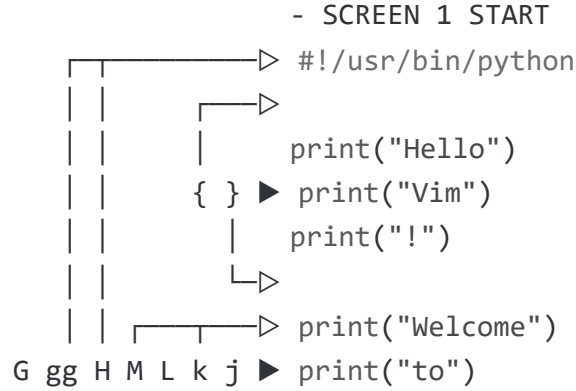
Motion Diagrams

▼/▶ Cursor ▽/▷ Target

Left-right motions



Up-down motions



```
|         |    └▷ print("quickref.me")
|         |       print("/vim")
|         |
|         └─────▷
|                 - SCREEN 1 END
└─────────────────▷ print("SCREEN 2")
```

Motions

h | j | k | l	Arrow keys
<C-u> / <C-d>	Half-page up/down
<C-b> / <C-f>	Page up/down
Words	
b / w	Previous/Next word
ge / e	Previous/Next end of word
Line	
0 (zero) / $	Start/End of line
^	Start of line (non-blank)
Character	
Fe / fe	Move to previous/next e
To / to	Move before/after previous/next o
| / n|	Go to first/nth column
Document	
gg / G	First/Last line
:n | nG	Go to line n
} / {	Next/Previous empty line
Window	
H / M / L	Top/Middle/Bottom screen
zt / zz / zb	Top/Center/Bottom this line

Insert Mode

`i` / `a`	Insert before/after cursor
`I` / `A`	Insert start/end of line
`o` / `O` (letter)	Insert new line below/above
`s` / `S`	Delete char/line and insert
`C` / `cc`	Change to end of/current line
`gi`	Insert at last insert point
`Esc` | `<C-[>`	Exit insert mode

Saving and Exiting

`:w`	Save
`:q`	Close file
`:wq` | `:x` | `ZZ`	Save and quit
`:wqa`	Save and quit all files
`:q!` | `ZQ`	Force quit
`:qa`	Close all files
`:qa!`	Force quit all files
`:w` now.txt	Write to `now.txt`
`:sav` new.txt	Save and edit `new.txt`
`:w` !sudo tee %	Write to readonly file

Normal Mode

`r`	Replace one character
`R`	Enter Replace mode
`u` / `3u`	Undo changes 1 / 3 times
`U`	Undo changes on one line

`J`	Join with next line
`<C-r>` / 5 `<C-r>`	Redo changes 1 / 5 times

Cut and paste

`x`	Delete character (Cut)
`p` / `P`	Paste after/before
`xp`	Swap two characters
`D`	Delete to end of line (Cut)
`dw`	Delete word (Cut)
`dd`	Delete line (Cut)
`ddp`	Swap two lines
`yy`	Yank line (Copy)
`"*p` | `"+p`	Paste from system clipboard
`"*y` | `"+y`	Paste to system clipboard

In visual mode

`d` | `x`	Delete selection (Cut)
`s`	Replace selection
`y`	Yank selection (Copy)

Repeating

`.`	Repeat last command
`;`	Repeat latest `f`, `t`, `F` or `T`
`,`	Repeat latest `f`, `t`, `F` or `T` reversed
`&`	Repeat last `:s`
`@:`	Repeat a command-line command

Visual mode	
`v`	Enter visual mode
`V`	Enter visual line mode
`<C-v>`	Enter visual block mode
`ggVG`	Select all text
`>` / `<`	Shift text right/left

Macros	
`qi`	Record macro `i`
`q`	Stop recording macro
`@i`	Run macro `i`
`7@i`	Run macro `i` 7 times
`@@`	Repeat last macro

You can save macro for any letters not just `i`

Vim Operators

Usage	
d	w
Operator	Motion

Combine operators with motions to use them

Available Operators	
d	Delete
y	Yank (copy)
c	Change (delete then insert)

p	Paste
=	Formats code
g~	Toggle case
gU	Uppercase
gu	Lowercase
>	Indent right
<	Indent left
!	Filter through external program

Examples

Combination	Description
dd	Delete current line
dj	Delete two lines
dw	Delete to next word
db	Delete to beginning of word
dfa	Delete until a char
d/hello	Delete until `hello`
cc	Change current line, synonym with `S`
yy	Copy current line
>j	Indent 2 lines
ggdG	Delete a complete document
gg=G	Indent a complete document
ggyG	Copy a whole document

Counts

```
[count] <operator> <motion>
<operator> [count] <motion>
```

2dd	Delete 2 lines
6yy	Copy 6 lines
d3w	Delete 3 words
d5j	Delete 5 lines downwards
>4k	Indent 4 lines upwards

Vim Text objects

Usage

v	i / a	p
Operator	inner / around	Text object

Operate with an operator inner or around text blocks

Text objects

p	Paragraph
w	Word
W	WORD (surrounded by whitespace)
s	Sentence
[({ <	A [], (), or {} block
]) } >	A [], (), or {} block
' " `	A quoted string
b	A block [(
B	A block in [{
t	A HTML tag block

See `:help text-objects`

Delete

diw	Delete inner word
dis	Delete inner sentence
di"	Delete in quotes
da"	Delete in quotes (including quotes)
dip	Delete a paragraph

Selections

vi"	Select inner quotes "..."
va"	Select quotes "..."
vi[	Select inner brackets [...]
va[	Select brackets [...]
viw	Select inner word
vip	Select inner paragraph
vipip	Select more paragraph

Misc

ciw	Change inner word
ci"	Change inner quotes
cit	Change inner tags (HTML)
cip	Change inner paragraph
yip	Yank inner paragraph
yap	Yank paragraph (including newline)

Vim Multiple files

Buffers

`:e file`	Edit a file in a new buffer
`:bn`	Go to the next buffer
`:bp`	Go to the previous buffer
`:bd`	Remove file from buffer list
`:b 5`	Open buffer #5
`:b file`	Go to a buffer by file
`:ls`	List all open buffers
`:sp file`	Open and split window
`:vs file`	Open and vertically split window
`:hid`	Hide this buffer
`:wn`	Write file and move to next
`:tab ba`	Edit all buffers as tabs

Windows

`<C-w>` `s`	Split window
`<C-w>` `v`	Split window vertically
`<C-w>` `w`	Switch windows
`<C-w>` `q`	Quit a window
`<C-w>` `T`	Break out into a new tab
`<C-w>` `x`	Swap current with next
`<C-w>` `-` / `+`	Decrease/Increase height
`<C-w>` `<` / `>`	Decrease/Increase width
`<C-w>` `|`	Max out the width

`<C-w>` `=`	Equally high and wide
`<C-w>` `h` / `l`	Go to the left/right window
`<C-w>` `j` / `k`	Go to the up/down window

Tabs

`:tabe [file]`	Edit file in a new tab
`:tabf [file]`	Open if exists in new tab
`:tabc`	Close current tab
`:tabo`	Close other tabs
`:tabs`	List all tabs
`:tabr`	Go to first tab
`:tabl`	Go to last tab
`:tabm 0`	Move to position `0`
`:tabn`	Go to next tab
`:tabp`	Go to previous tab

Normal mode

`gt`	Go to next tab
`gT`	Go to previous tab
`2gt`	Go to tab number 2

Vim Search and Replace

Search

`/foo`	Search forward
`/foo\c`	Search forward (case insensitive)

`?foo`	Search backward
`/\v\d+`	Search with regex
`n`	Next matching search pattern
`N`	Previous match
`*`	Search for current word forward
`#`	Search for current word backward

Replace LINE

`:[range]s/{pattern}/{str}/[flags]`

`:s/old/new`	Replace first
`:s/old/new/g`	Replace all
`:s/\vold/new/g`	Replace all with regex
`:s/old/new/gc`	replace all (Confirm)
`:s/old/new/i`	Ignore case replace first
`:2,6s/old/new/g`	Replace between lines 2-6

Replace FILE

`:%s/{pattern}/{str}/[flags]`

`:%s/old/new`	Replace first
`:%s/old/new/g`	Replace all
`:%s/old/new/gc`	Replace all (Confirm)
`:%s/old/new/gi`	Replace all (ignore case)
`:%s/\vold/new/g`	Replace all with regex

Ranges

`%`	Entire file

`'<,'>`	Current selection
`5`	Line 5
`5,10`	Lines 5 to 10
`$`	Last line
`2,$`	Lines 2 to Last
`.`	Current line
`,3`	Next 3 lines
`-3,`	Forward 3 lines

Global command

`:[range]g/{pattern}/[command]`

`:g/foo/d`	Delete lines containing `foo`
`:g!/foo/d`	Delete lines not containing `foo`
`:g/^\s*$/d`	Delete all blank lines
`:g/foo/t$`	Copy lines containing `foo` to EOF
`:g/foo/m$`	Move lines containing `foo` to EOF
`:g/^/m0`	Reverse a file
`:g/^/t.`	Duplicate every line

Inverse :g

`:[range]v/{pattern}/[command]`

`:v/foo/d`	Delete lines not containing `foo` (also `:g!/foo/d`)

Flags

`g`	Replace all occurrences
`i`	Ignore case

I	Don't ignore case
c	Confirm each substitution

Substitute expression (magic)

& | \0	Replace with the whole matched
\1...\9	Replace with the group 0-9
\u	Uppercase next letter
\U	Uppercase following characters
\l	Lowercase next letter
\L	Lowercase following characters
\e	End of \u, \U, \l and \L
\E	End of \u, \U, \l and \L

Examples

```
:s/a\|b/xxx\0xxx/g              # Modifies "a b"      to "xxxaxxx xxxbxxx"
:s/test/\U& file/               # Modifies "test"     to "TEST FILE"
:s/\(test\)/\U\1\e file/        # Modifies "test"     to "TEST file"
:s/\v([abc])([efg])/\2\1/g      # Modifies "af fa bg" to "fa fa gb"
:s/\v\w+/\u\0/g                 # Modifies "bla bla"  to "Bla Bla"
:s/\v([ab])|([cd])/\1x/g        # Modifies "a b c d"  to "ax bx x x"
:%s/.*/\L&/                     # Modifies "HTML"     to "html"
:s/\v<(.)(\w*)/\u\1\L\2/g       # Make every first letter of a word uppercase
:%s/^\(.*\)\n\1/\1/             # Remove duplicate lines
:%s/<\/\=\(\w\+\)\>/\U&/g       # Convert HTML-Tags to uppercase
:g/^pattern/s/$/mytext          # Find and append text to the end
:g/pattern/norm! @i             # Run a macro on matching lines
/^\(.*\)\(\r\?\n\1\)\+$         # View the duplicates lines
/\v^(.*)(\r?\n\1)+$             # View the duplicates lines (very magic)
:v/./,/./-j                     # Compress blank lines into a blank line
:g/<p1>/,/<p2>/d                # Delete inclusively from <p1> to <p2>
```

Vimdiff

Usage

```
$ vimdiff file1 file2 [file3]
$ vim -d file1 file2 [file3]
```

Editing

```
:[range]diffget [bufspec]
:[range]diffput [bufspec]
```

`do` / `:diffget`	Obtain (get) difference
`dp` / `:diffput`	Put difference
`:dif`	Re-scan differences
`:diffo`	Switch off diff mode
`:1,$+1diffget`	Get all differences
`ZQ`	Quit without changes

See: Ranges

Folds

`zo` / `zO`	Open
`zc` / `zC`	Close
`za` / `zA`	Toggle
`zv`	Open folds for this line
`zM`	Close all
`zR`	Open all
`zm`	Fold more (foldlevel += 1)
`zr`	Fold less (foldlevel -= 1)
`zx`	Update folds

Jumping

`]c`	Next difference
`[c`	Previous difference

Miscellaneous

Case

`vU`	Uppercase character
`vu`	Lowercase character
`~`	Toggle case character
`viw` `U`	Uppercase word
`viw` `u`	Lowercase word
`viw` `~`	Toggle case word
`VU` / `gUU`	Uppercase line
`Vu` / `guu`	Lowercase line
`V~` / `g~~`	Toggle case line
`gggUG`	Uppercase all text
`ggguG`	Lowercase all text
`ggg~G`	Toggle case all text

Jumping

`<C-o>`	Go back to previous
`<C-i>`	Go forward

`gf`	Go to file in cursor
`ga`	Display hex, ascii value

Misc command-lines

`:h`	Help open help view
`:edit!`	Reload current file
`:2,8m0`	Move lines `2-8` to `0`
`:noh`	Clear search highlights
`:sort`	Sort lines
`:ter`	Open a terminal window
`:set paste`	Enable Insert Paste sub-mode
`:set nopaste`	disable Insert Paste sub-mode
`:cq`	Exiting with an error (aborting Git)

Navigating

`%`	Nearest/matching `{[()]}`
`[(` | `[{`	Previous (or {
`])` | `]{`	Next) or }
`[m`	Previous method start
`[M`	Previous method end

Counters

`<C-a>`	Increase number
`<C-x>`	Decrease number

Tags

`:tag Classname`	Jump to first definition of Classname

`<C-]>`	Jump to definition
`g]`	See all definitions
`<C-t>`	Go back to last tag
`<C-o> <C-i>`	Back/forward
`:tselect Classname`	Find definitions of Classname
`:tjump Classname`	Find definitions of Classname (auto-select 1st)

Formatting

`:ce 8`	Center lines between 8 columns
`:ri 4`	Right-align lines at 4 columns
`:le`	Left-align lines

See `:help formatting`

Marks

`` `^ ``	Last position of cursor in insert mode
`` `. ``	Last change in current buffer
`` `" ``	Last exited current buffer
`` `0 ``	In last file edited
`''`	Back to line in current buffer where jumped from
`` `` ```	Back to position in current buffer where jumped from
`` `[ ``	To beginning of previously changed or yanked text
`` `] ``	To end of previously changed or yanked text
`` `< ``	To beginning of last visual selection
`` `> ``	To end of last visual selection
`ma`	Mark this cursor position as a
`` `a ``	Jump to the cursor position a

`'a`	Jump to the beginning of the line with position a
`d'a`	Delete from current line to line of mark a
`d`a`	Delete from current position to position of mark a
`c'a`	Change text from current line to line of a
`y`a`	Yank text from current position to position of a
`:marks`	List all current marks
`:delm a`	Delete mark a
`:delm a-d`	Delete marks a, b, c, d
`:delm abc`	Delete marks a, b, c

Calculator

`<C-r>` `=` 7*7	Shows the result
`<C-r>` `=` 10/2	Shows the result

Do this in INSERT mode

Shell

`:!<shell>`	Interpret Shell Command
`:r!<shell>`	Read in output of shell
`:r!date`	Insert date
`:!!date`	Replace current line with date

Command line

`<C-r><C-w>`	Insert current word into the command line
`<C-r>"`	Paste from " register
`<C-x><C-f>`	Auto-completion of path in insert mode

Tricks

Remove duplicate lines

```
:sort | %!uniq -u
```

To number the lines in the file

```
:%!cat -n
```

Copy whole doc to clipboard

```
:%w !pbcopy             # Mac OS X
:%w !xclip -i -sel c    # GNU/Linux
:%w !xsel -i -b         # GNU/Linux
```

Also see

Devhints (devhints.io)
Vim cheatsheet (vim.rotrr.com)
Vim documentation (vimdoc.sourceforge.net)
Interactive Vim tutorial (openvim.com)

Related Cheatsheet

Emacs Cheatsheet
Quick Reference

VSCode Cheatsheet
Quick Reference

Sed Cheatsheet
Quick Reference

Awk Cheatsheet
Quick Reference

Recent Cheatsheet

Remote Work Revolution Cheatsheet

Homebrew Cheatsheet

Quick Reference

Quick Reference

PyTorch Cheatsheet

Quick Reference

Taskset Cheatsheet

Quick Reference

Git

This cheat sheet summarizes commonly used Git command line instructions for quick reference.

Getting Started

Create a Repository

Create a new local repository

```
$ git init [project name]
```

Clone a repository

```
$ git clone git_url
```

Clone a repository into a specified directory

```
$ git clone git_url my_directory
```

Make a change

Show modified files in working directory, staged for your next commit

```
$ git status
```

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to version history

```
$ git commit -m "commit message"
```

Commit all your tracked files to version history

```
$ git commit -am "commit message"
```

Discard changes in working directory which is not staged

```
$ git restore [file]
```

Unstage a staged file or file which is staged

```
$ git restore --staged [file]
```

Unstage a file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

Diff of what is changed but not staged

```
$ git diff
```

Diff of what is staged but not yet committed

```
$ git diff --staged
```

Apply any commits of current branch ahead of specified one

```
$ git rebase [branch]
```

Configuration

Set the name that will be attached to your commits and tags

```
$ git config --global user.name "name"
```

Set an email address that will be attached to your commits and tags

```
$ git config --global user.email "email"
```

Enable some colorization of Git output

```
$ git config --global color.ui auto
```

Edit the global configuration file in a text editor

```
$ git config --global --edit
```

Working with Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to my_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new_branch

```
$ git checkout -b new_branch
```

Delete the branch called my_branch

```
$ git branch -d my_branch
```

Merge branchA into branchB

```
$ git checkout branchB
$ git merge branchA
```

Tag the current commit

```
$ git tag my_tag
```

Observe your Repository

Show the commit history for the currently active branch

```
$ git log
```

Show the commits on branchA that are not on branchB

```
$ git log branchB..branchA
```

Show the commits that changed file, even across renames

```
$ git log --follow [file]
```

Show the diff of what is in branchA that is not in branchB

```
$ git diff branchB...branchA
```

Show any object in Git in human-readable format

```
$ git show [SHA]
```

Synchronize

Fetch down all the branches from that Git remote

```
$ git fetch [alias]
```

Merge a remote branch into your current branch to bring it up to date

```
$ git merge [alias]/[branch]
# No fast-forward
$ git merge --no-ff [alias]/[branch]
# Only fast-forward
$ git merge --ff-only [alias]/[branch]
```

Transmit local branch commits to the remote repository branch

```
$ git push [alias] [branch]
```

Fetch and merge any commits from the tracking remote branch

```
$ git pull
```

Merge just one specific commit from another branch to your current branch

```
$ git cherry-pick [commit_id]
```

Remote

Add a git URL as an alias

```
$ git remote add [alias] [url]
```

Show the names of the remote repositories you've set up

```
$ git remote
```

Show the names and URLs of the remote repositories

```
$ git remote -v
```

Remove a remote repository

```
$ git remote rm [remote repo name]
```

Change the URL of the git repo

```
$ git remote set-url origin [git_url]
```

Temporary Commits

Save modified and staged changes

```
$ git stash
```

List stack-order of stashed file changes

```
$ git stash list
```

Write working from top of stash stack

```
$ git stash pop
```

Discard the changes from top of stash stack

```
$ git stash drop
```

Tracking path Changes

Delete the file from project and stage the removal for commit

```
$ git rm [file]
```

Change an existing file path and stage the move

```
$ git mv [existing-path] [new-path]
```

Show all commit logs with indication of any paths that moved

```
$ git log --stat -M
```

Ignoring Files

```
/logs/*

# "!" means don't ignore
!logs/.gitkeep

/# Ignore Mac system files
.DS_store

# Ignore node_modules folder
node_modules

# Ignore SASS config files
.sass-cache
```

A `.gitignore` file specifies intentionally untracked files that Git should ignore

Git Tricks

Rename branch

1. **Renamed** to `new_name`

```
$ git branch -m <new_name>
```

2. **Push** and reset

```
$ git push origin -u <new_name>
```

3. **Delete** remote branch

```shell script
$ git push origin --delete <old>
```

Log

Search change by content

```
$ git log -S'<a term in the source>'
```

Show changes over time for specific file

```
$ git log -p <file_name>
```

Print out a cool visualization of your log

```
$ git log --pretty=oneline --graph --decorate --all
```

Branch

List all branches and their upstreams

```
$ git branch -vv
```

Quickly switch to the previous branch

```
$ git checkout -
```

Get only remote branches

```
$ git branch -r
```

Checkout a single file from another branch

```
$ git checkout <branch> -- <file>
```

Rewriting history

Rewrite last commit message

```
$ git commit --amend -m "new message"
```

Amend the latest commit without changing the commit message.

```
$ git commit --amend --no-edit
```

See also: Rewriting history

Git Aliases

```
git config --global alias.co checkout
git config --global alias.br branch
git config --global alias.ci commit
git config --global alias.st status
```

See also: More Aliases

Advanced Git

Submodules

Create a new submodule within your repository:

```
$ git submodule add <repository_url> <path>
```

Clone a repository and initialize its submodules:

```
$ git clone --recursive <repository_url>
```

Update all the submodules in your repository to the latest commit of their respective branches:

```
$ git submodule update
```

Pull the latest changes from the remote repositories of the submodules and update them in your main repository:

```
$ git submodule update --remote
```

Remove a submodule from your repository:

```
$ git submodule deinit <path>
$ git rm <path>
$ git commit -m "Removed submodule"
```

Cherry-picking

Cherry-picking allows you to apply a specific commit from one branch to another branch.

```
$ git cherry-pick <commit_hash>
```

Reflog

Display the reflog, showing the history of HEAD and branch movements:

```
$ git reflog
```

Find the hash of the lost commit or branch using the reflog and then checkout to that hash to restore it:

```
$ git checkout <commit_or_branch_hash>
```

Top Cheatsheet

Python Cheatsheet
Quick Reference

Vim Cheatsheet
Quick Reference

JavaScript Cheatsheet
Quick Reference

Bash Cheatsheet
Quick Reference

Recent Cheatsheet

Gnome Desktop Cheatsheet
Quick Reference

Arduino Programming Cheatsheet
Quick Reference

HTML Canvas Cheatsheet
Quick Reference

TypeScript Cheatsheet
Quick Reference

MySQL

The SQL cheat sheet provides you with the most commonly used SQL statements for your reference.

Getting Started

Connect MySQL

```
mysql -u <user> -p

mysql [db_name]

mysql -h <host> -P <port> -u <user> -p [db_name]

mysql -h <host> -u <user> -p [db_name]
```

Commons

Database	
`CREATE DATABASE db ;`	Create database
`SHOW DATABASES;`	List databases
`USE db;`	Switch to db
`CONNECT db ;`	Switch to db
`DROP DATABASE db;`	Delete db
Table	
`SHOW TABLES;`	List tables for current db
`SHOW FIELDS FROM t;`	List fields for a table
`DESC t;`	Show table structure

`SHOW CREATE TABLE t;`	Show create table sql
`TRUNCATE TABLE t;`	Remove all data in a table
`DROP TABLE t;`	Delete table
Process	
`show processlist;`	List processes
`kill pid;`	kill process
Other	
`exit` or `\q`	Exit MySQL session

Backups

Create a backup

```
mysqldump -u user -p db_name > db.sql
```

Export db without schema

```
mysqldump -u user -p db_name --no-data=true --add-drop-table=false > db.sql
```

Restore a backup

```
mysql -u user -p db_name < db.sql
```

MySQL Examples

Managing tables

Create a new table with three columns

```
CREATE TABLE t (
     id    INT,
     name  VARCHAR DEFAULT NOT NULL,
     price INT DEFAULT 0
     PRIMARY KEY(id)
);
```

Delete the table from the database

```
DROP TABLE t ;
```

Add a new column to the table

```
ALTER TABLE t ADD column;
```

Drop column c from the table

```
ALTER TABLE t DROP COLUMN c ;
```

Add a constraint

```
ALTER TABLE t ADD constraint;
```

Drop a constraint

```
ALTER TABLE t DROP constraint;
```

Rename a table from t1 to t2

```
ALTER TABLE t1 RENAME TO t2;
```

Rename column c1 to c2

```
ALTER TABLE t1 RENAME c1 TO c2 ;
```

Remove all data in a table

```
TRUNCATE TABLE t;
```

Querying data from a table

Query data in columns c1, c2 from a table

```
SELECT c1, c2 FROM t
```

Query all rows and columns from a table

```
SELECT * FROM t
```

Query data and filter rows with a condition

```
SELECT c1, c2 FROM t
WHERE condition
```

Query distinct rows from a table

```
SELECT DISTINCT c1 FROM t
WHERE condition
```

Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC]
```

Skip offset of rows and return the next n rows

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset
```

Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
```

Filter groups using HAVING clause

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition
```

Querying from multiple tables

Inner join t1 and t2

```
SELECT c1, c2
FROM t1
```

```
INNER JOIN t2 ON condition
```

Left join t1 and t1

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition
```

Right join t1 and t2

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition
```

Perform full outer join

```
SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition
```

Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1
CROSS JOIN t2
```

Another way to perform cross join

```
SELECT c1, c2
FROM t1, t2
```

Join t1 to itself using INNER JOIN clause

```
SELECT c1, c2
FROM t1 A
INNER JOIN t1 B ON condition
```

Using SQL Operators Combine rows from two queries

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2
```

Return the intersection of two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2
```

Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2
```

Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern
```

Query rows in a list

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list
```

Query rows between two values

```
SELECT c1, c2 FROM t
WHERE  c1 BETWEEN low AND high
```

Check if values in a table is NULL or not

```
SELECT c1, c2 FROM t
WHERE  c1 IS [NOT] NULL
```

Using SQL constraints

Set c1 and c2 as a primary key

```
CREATE TABLE t(
    c1 INT, c2 INT, c3 VARCHAR,
    PRIMARY KEY (c1,c2)
);
```

Set c2 column as a foreign key

```
CREATE TABLE t1(
    c1 INT PRIMARY KEY,
    c2 INT,
    FOREIGN KEY (c2) REFERENCES t2(c2)
);
```

Make the values in c1 and c2 unique

```
CREATE TABLE t(
    c1 INT, c1 INT,
    UNIQUE(c2,c3)
);
```

Ensure c1 > 0 and values in c1 >= c2

```
CREATE TABLE t(
  c1 INT, c2 INT,
  CHECK(c1> 0 AND c1 >= c2)
);
```

Set values in c2 column not NULL

```
CREATE TABLE t(
     c1 INT PRIMARY KEY,
     c2 VARCHAR NOT NULL
);
```

Modifying Data

Insert one row into a table

```
INSERT INTO t(column_list)
VALUES(value_list);
```

Insert multiple rows into a table

```
INSERT INTO t(column_list)
VALUES (value_list),
       (value_list), …;
```

Insert rows from t2 into t1

```
INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
```

Update new value in the column c1 for all rows

```
UPDATE t
SET c1 = new_value;
```

Update values in the column c1, c2 that match the condition

```
UPDATE t
SET c1 = new_value,
        c2 = new_value
WHERE condition;
```

Delete all data in a table

```
DELETE FROM t;
```

Delete subset of rows in a table

```
DELETE FROM t
WHERE condition;
```

Managing Views

Create a new view that consists of c1 and c2

```
CREATE VIEW v(c1,c2)
AS
SELECT c1, c2
FROM t;
```

Create a new view with check option

```
CREATE VIEW v(c1,c2)
AS
SELECT c1, c2
FROM t;
WITH [CASCADED | LOCAL] CHECK OPTION;
```

Create a recursive view

```
CREATE RECURSIVE VIEW v
AS
select-statement -- anchor part
UNION [ALL]
select-statement; -- recursive part
```

Create a temporary view

```
CREATE TEMPORARY VIEW v
AS
SELECT c1, c2
FROM t;
```

Delete a view

```
DROP VIEW view_name;
```

Managing triggers

Create or modify a trigger

```
CREATE OR MODIFY TRIGGER trigger_name
WHEN EVENT
ON table_name TRIGGER_TYPE
EXECUTE stored_procedure;
```

WHEN	
BEFORE	invoke before the event occurs
AFTER	invoke after the event occurs
EVENT	
INSERT	invoke for INSERT
UPDATE	invoke for UPDATE
DELETE	invoke for DELETE
TRIGGER_TYPE	
FOR EACH ROW	
FOR EACH STATEMENT	

Managing indexes

Create an index on c1 and c2 of the t table

```
CREATE INDEX idx_name
ON t(c1,c2);
```

Create a unique index on c3, c4 of the t table

```
CREATE UNIQUE INDEX idx_name
ON t(c3,c4)
```

Drop an index

```
DROP INDEX idx_name ON t;
```

MySQL Data Types

Strings

CHAR	String (0 - 255)
VARCHAR	String (0 - 255)
TINYTEXT	String (0 - 255)
TEXT	String (0 - 65535)
BLOB	String (0 - 65535)
MEDIUMTEXT	String (0 - 16777215)
MEDIUMBLOB	String (0 - 16777215)
LONGTEXT	String (0 - 4294967295)
LONGBLOB	String (0 - 4294967295)
ENUM	One of preset options
SET	Selection of preset options

Date & time

DATE	yyyy-MM-dd
TIME	hh:mm:ss
DATETIME	yyyy-MM-dd hh:mm:ss
TIMESTAMP	yyyy-MM-dd hh:mm:ss
YEAR	yyyy

Numeric

TINYINT x	Integer (-128 to 127)
SMALLINT x	Integer (-32768 to 32767)
MEDIUMINT x	Integer (-8388608 to 8388607)
INT x	Integer (-2147483648 to 2147483647)
BIGINT x	Integer (-9223372036854775808 to 9223372036854775807)
FLOAT	Decimal (precise to 23 digits)
DOUBLE	Decimal (24 to 53 digits)
DECIMAL	"DOUBLE" stored as string

MySQL Functions & Operators

Strings

ASCII()	BIN()
BIT_LENGTH()	CHAR()
CHARACTER_LENGTH()	CHAR_LENGTH()
CONCAT()	CONCAT_WS()
ELT()	EXPORT_SET()

FIELD()	FIND_IN_SET()
FORMAT()	FROM_BASE64()
HEX()	INSERT()
INSTR()	LCASE()
LEFT()	LENGTH()
LIKE	LOAD_FILE()
LOCATE()	LOWER()
LPAD()	LTRIM()
MAKE_SET()	MATCH
MID()	NOT LIKE
NOT REGEXP	OCT()
OCTET_LENGTH()	ORD()
POSITION()	QUOTE()
REGEXP	REGEXP_INSTR()
REGEXP_LIKE()	REGEXP_REPLACE()
REGEXP_SUBSTR()	REPEAT()
REPLACE()	REVERSE()
RIGHT()	RLIKE
RPAD()	RTRIM()
SOUNDEX()	SOUNDS LIKE
SPACE()	STRCMP()
SUBSTR()	SUBSTRING()
SUBSTRING_INDEX()	TO_BASE64()
TRIM()	UCASE()
UNHEX()	UPPER()

WEIGHT_STRING()	

Date and Time

ADDDATE()	ADDTIME()
CONVERT_TZ()	CURDATE()
CURRENT_DATE()	CURRENT_TIME()
CURRENT_TIMESTAMP()	CURTIME()
DATE()	DATE_ADD()
DATE_FORMAT()	DATE_SUB()
DATEDIFF()	DAY()
DAYNAME()	DAYOFMONTH()
DAYOFWEEK()	DAYOFYEAR()
EXTRACT()	FROM_DAYS()
FROM_UNIXTIME()	GET_FORMAT()
HOUR()	LAST_DAY
LOCALTIME()	LOCALTIMESTAMP()
MAKEDATE()	MAKETIME()
MICROSECOND()	MINUTE()
MONTH()	MONTHNAME()
NOW()	PERIOD_ADD()
PERIOD_DIFF()	QUARTER()
SEC_TO_TIME()	SECOND()
STR_TO_DATE()	SUBDATE()
SUBTIME()	SYSDATE()
TIME()	TIME_FORMAT()
TIME_TO_SEC()	TIMEDIFF()

TIMESTAMP()	TIMESTAMPADD()
TIMESTAMPDIFF()	TO_DAYS()
TO_SECONDS()	UNIX_TIMESTAMP()
UTC_DATE()	UTC_TIME()
UTC_TIMESTAMP()	WEEK()
WEEKDAY()	WEEKOFYEAR()
YEAR()	YEARWEEK()
GET_FORMAT()	

Numeric

%, MOD	*
+	-
-	/
ABS()	ACOS()
ASIN()	ATAN()
ATAN2(), ATAN()	CEIL()
CEILING()	CONV()
COS()	COT()
CRC32()	DEGREES()
DIV	EXP()
FLOOR()	LN()
LOG()	LOG10()
LOG2()	MOD()
PI()	POW()
POWER()	RADIANS()
RAND()	ROUND()

SIGN()	SIN()
SQRT()	TAN()
TRUNCATE()	

Aggregate

AVG()	BIT_AND()
BIT_OR()	BIT_XOR()
COUNT()	COUNT(DISTINCT)
GROUP_CONCAT()	JSON_ARRAYAGG()
JSON_OBJECTAGG()	MAX()
MIN()	STD()
STDDEV()	STDDEV_POP()
STDDEV_SAMP()	SUM()
VAR_POP()	VAR_SAMP()
VARIANCE()	

JSON

->
->>
JSON_ARRAY()
JSON_ARRAY_APPEND()
JSON_ARRAY_INSERT()
JSON_CONTAINS()
JSON_CONTAINS_PATH()
JSON_DEPTH()
JSON_EXTRACT()
JSON_INSERT()

JSON_KEYS()

JSON_LENGTH()

JSON_MERGE() (deprecated)

JSON_MERGE_PATCH()

JSON_MERGE_PRESERVE()

JSON_OBJECT()

JSON_OVERLAPS() (introduced 8.0.17)

JSON_PRETTY()

JSON_QUOTE()

JSON_REMOVE()

JSON_REPLACE()

JSON_SCHEMA_VALID() (introduced 8.0.17)

JSON_SCHEMA_VALIDATION_REPORT() (introduced 8.0.17)

JSON_SEARCH()

JSON_SET()

JSON_STORAGE_FREE()

JSON_STORAGE_SIZE()

JSON_TABLE()

JSON_TYPE()

JSON_UNQUOTE()

JSON_VALID()

JSON_VALUE() (introduced 8.0.21)

MEMBER OF() (introduced 8.0.17)

Cast

BINARY	CAST()
CONVERT()	

Flow Control

CASE	IF()
IFNULL()	NULLIF()

Information

BENCHMARK()	CHARSET()
COERCIBILITY()	COLLATION()
CONNECTION_ID()	CURRENT_ROLE()
CURRENT_USER()	DATABASE()
FOUND_ROWS()	ICU_VERSION()
LAST_INSERT_ID()	ROLES_GRAPHML()
ROW_COUNT()	SCHEMA()
SESSION_USER()	SYSTEM_USER()
USER()	VERSION()

Encryption and Compression

AES_DECRYPT()
AES_ENCRYPT()
COMPRESS()
MD5()
RANDOM_BYTES()
SHA1(), SHA()
SHA2()
STATEMENT_DIGEST()

STATEMENT_DIGEST_TEXT()
UNCOMPRESS()
UNCOMPRESSED_LENGTH()
VALIDATE_PASSWORD_STRENGTH()

Locking

GET_LOCK()
IS_FREE_LOCK()
IS_USED_LOCK()
RELEASE_ALL_LOCKS()
RELEASE_LOCK()

Bit

&	>>
<<	^
BIT_COUNT()	|
~	

Miscellaneous

ANY_VALUE()	BIN_TO_UUID()
DEFAULT()	GROUPING()
INET_ATON()	INET_NTOA()
INET6_ATON()	INET6_NTOA()
IS_IPV4()	IS_IPV4_COMPAT()
IS_IPV4_MAPPED()	IS_IPV6()
IS_UUID()	MASTER_POS_WAIT()
NAME_CONST()	SLEEP()

UUID()	UUID_SHORT()
UUID_TO_BIN()	VALUES()

Also see

Regex in MySQL (cheatsheets.zip)

Related Cheatsheet

PostgreSQL Cheatsheet
Quick Reference

MongoDB Cheatsheet
Quick Reference

Neo4j Cheatsheet
Quick Reference

Recent Cheatsheet

Gnome Desktop Cheatsheet
Quick Reference

Arduino Programming Cheatsheet
Quick Reference

HTML Canvas Cheatsheet
Quick Reference

TypeScript Cheatsheet
Quick Reference

MongoDB

The MongoDB cheat sheet provides you with the most commonly used MongoDB commands and queries for your reference. the cheatsheet is from mongodb developers website

Getting Started

Connect MongoDB Shell

```
mongo # connects to mongodb://127.0.0.1:27017 by default
```

```
mongo --host <host> --port <port> -u <user> -p <pwd> # omit the password if you wa
```

```
mongo "mongodb://192.168.1.1:27017"
```

```
mongo "mongodb+srv://cluster-name.abcde.mongodb.net/<dbname>" --username <username
```

Helpers

Show dbs :

```
db // prints the current database
```

Switch database :

```
use <database_name>
```

Show collections :

```
show collections
```

Run JavaScript file :

```
load("myScript.js")
```

Crud

Create

```
db.coll.insertOne({name: "Max"})
db.coll.insert([{name: "Max"}, {name:"Alex"}]) // ordered bulk insert
db.coll.insert([{name: "Max"}, {name:"Alex"}], {ordered: false}) // unordered bulk
db.coll.insert({date: ISODate()})
db.coll.insert({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 5000}
```

Delete

```
db.coll.remove({name: "Max"})
db.coll.remove({name: "Max"}, {justOne: true})
db.coll.remove({}) // WARNING! Deletes all the docs but not the collection itself
db.coll.remove({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 5000}
db.coll.findOneAndDelete({"name": "Max"})
```

Update

```
db.coll.update({"_id": 1}, {"year": 2016}) // WARNING! Replaces the entire documer
db.coll.update({"_id": 1}, {$set: {"year": 2016, name: "Max"}})
db.coll.update({"_id": 1}, {$unset: {"year": 1}})
db.coll.update({"_id": 1}, {$rename: {"year": "date"} })
db.coll.update({"_id": 1}, {$inc: {"year": 5}})
db.coll.update({"_id": 1}, {$mul: {price: NumberDecimal("1.25"), qty: 2}})
db.coll.update({"_id": 1}, {$min: {"imdb": 5}})
db.coll.update({"_id": 1}, {$max: {"imdb": 8}})
db.coll.update({"_id": 1}, {$currentDate: {"lastModified": true}})
db.coll.update({"_id": 1}, {$currentDate: {"lastModified": {$type: "timestamp"}}})
```

Array

```
db.coll.update({"_id": 1}, {$push :{"array": 1}})
db.coll.update({"_id": 1}, {$pull :{"array": 1}})
```

```
db.coll.update({"_id": 1}, {$addToSet :{"array": 2}})
db.coll.update({"_id": 1}, {$pop: {"array": 1}})  // last element
db.coll.update({"_id": 1}, {$pop: {"array": -1}}) // first element
db.coll.update({"_id": 1}, {$pullAll: {"array" :[3, 4, 5]}})
db.coll.update({"_id": 1}, {$push: {scores: {$each: [90, 92, 85]}}})
db.coll.updateOne({"_id": 1, "grades": 80}, {$set: {"grades.$": 82}})
db.coll.updateMany({}, {$inc: {"grades.$[]": 10}})
db.coll.update({}, {$set: {"grades.$[element]": 100}}, {multi: true, arrayFilters:
```

Update many

```
db.coll.update({"year": 1999}, {$set: {"decade": "90's"}}, {"multi":true})
db.coll.updateMany({"year": 1999}, {$set: {"decade": "90's"}})
```

FindOneAndUpdate

```
db.coll.findOneAndUpdate({"name": "Max"}, {$inc: {"points": 5}}, {returnNewDocumer
```

Upsert

```
db.coll.update({"_id": 1}, {$set: {item: "apple"}, $setOnInsert: {defaultQty: 100}
```

Replace

```
db.coll.replaceOne({"name": "Max"}, {"firstname": "Maxime", "surname": "Beugnet"})
```

Save

```
db.coll.save({"item": "book", "qty": 40})
```

Write concern

```
db.coll.update({}, {$set: {"x": 1}}, {"writeConcern": {"w": "majority", "wtimeout"
```

Find

```
db.coll.findOne() // returns a single document
db.coll.find()    // returns a cursor - show 20 results - "it" to display more
db.coll.find().pretty()
db.coll.find({name: "Max", age: 32}) // implicit logical "AND".
```

```
db.coll.find({date: ISODate("2020-09-25T13:57:17.180Z")})
db.coll.find({name: "Max", age: 32}).explain("executionStats") // or "queryPlanner
db.coll.distinct("name")
```

Count

```
db.coll.count({age: 32})          // estimation based on collection metadata
db.coll.estimatedDocumentCount()  // estimation based on collection metadata
db.coll.countDocuments({age: 32}) // alias for an aggregation pipeline - accurate
```

Comparison

```
db.coll.find({"year": {$gt: 1970}})
db.coll.find({"year": {$gte: 1970}})
db.coll.find({"year": {$lt: 1970}})
db.coll.find({"year": {$lte: 1970}})
db.coll.find({"year": {$ne: 1970}})
db.coll.find({"year": {$in: [1958, 1959]}})
db.coll.find({"year": {$nin: [1958, 1959]}})
```

Logical

```
db.coll.find({name:{$not: {$eq: "Max"}}})
db.coll.find({$or: [{"year" : 1958}, {"year" : 1959}]})
db.coll.find({$nor: [{price: 1.99}, {sale: true}]})
db.coll.find({
$and: [
    {$or: [{qty: {$lt :10}}, {qty :{$gt: 50}}]},
{$or: [{sale: true}, {price: {$lt: 5 }}]}
]
})
```

Element

```
db.coll.find({name: {$exists: true}})
db.coll.find({"zipCode": {$type: 2 }})
db.coll.find({"zipCode": {$type: "string"}})
```

Aggregation Pipeline

```
db.coll.aggregate([
{$match: {status: "A"}},
{$group: {_id: "$cust_id", total: {$sum: "$amount"}}},
```

```
{$sort: {total: -1}}
])
```

Text search with a "text" index

```
db.coll.find({$text: {$search: "cake"}}, {score: {$meta: "textScore"}}).sort({scor
```

Regex

```
db.coll.find({name: /^Max/}) // regex: starts by letter "M"
db.coll.find({name: /^Max$/i}) // regex case insensitive
```

Array

```
db.coll.find({tags: {$all: ["Realm", "Charts"]}})
db.coll.find({field: {$size: 2}}) // impossible to index - prefer storing the size
db.coll.find({results: {$elemMatch: {product: "xyz", score: {$gte: 8}}}})
```

Projections

```
db.coll.find({"x": 1}, {"actors": 1}) // actors + \_id
db.coll.find({"x": 1}, {"actors": 1, "\_id": 0}) // actors
db.coll.find({"x": 1}, {"actors": 0, "summary": 0}) // all but "actors" and "summa
```

Sort, skip, limit

```
db.coll.find({}).sort({"year": 1, "rating": -1}).skip(10).limit(3)
```

Read Concern

```
db.coll.find().readConcern("majority")
```

Databases and Collections

Drop

```
db.coll.drop()    // removes the collection and its index definitions
db.dropDatabase() // double check that you are *NOT* on the PROD cluster... :-)
```

Create Collection

```
db.createCollection("contacts", {
   validator: {$jsonSchema: {
      bsonType: "object",
      required: ["phone"],
      properties: {
         phone: {
            bsonType: "string",
            description: "must be a string and is required"
         },
         email: {
            bsonType: "string",
            pattern: "@mongodb\.com$",
            description: "must be a string and match the regular expression patter
         },
         status: {
            enum: [ "Unknown", "Incomplete" ],
            description: "can only be one of the enum values"
         }
      }
   }}
})
```

Other Collection Functions

```
db.coll.stats()
db.coll.storageSize()
db.coll.totalIndexSize()
db.coll.totalSize()
db.coll.validate({full: true})
db.coll.renameCollection("new_coll", true) // 2nd parameter to drop the target col
```

Indexes

Basics

List

```
db.coll.getIndexes()
db.coll.getIndexKeys()
```

Drop Indexes

```
db.coll.dropIndex("name_1")
```

Hide/Unhide Indexes

```
db.coll.hideIndex("name_1")
db.coll.unhideIndex("name_1")
```

Create Indexes

```
// Index Types
db.coll.createIndex({"name": 1})                // single field index
db.coll.createIndex({"name": 1, "date": 1})     // compound index
db.coll.createIndex({foo: "text", bar: "text"}) // text index
db.coll.createIndex({"$**": "text"})            // wildcard text index
db.coll.createIndex({"userMetadata.$**": 1})    // wildcard index
db.coll.createIndex({"loc": "2d"})              // 2d index
db.coll.createIndex({"loc": "2dsphere"})        // 2dsphere index
db.coll.createIndex({"_id": "hashed"})          // hashed index

// Index Options
db.coll.createIndex({"lastModifiedDate": 1}, {expireAfterSeconds: 3600})      // T
db.coll.createIndex({"name": 1}, {unique: true})
db.coll.createIndex({"name": 1}, {partialFilterExpression: {age: {$gt: 18}}}) // p
db.coll.createIndex({"name": 1}, {collation: {locale: 'en', strength: 1}})    // c
db.coll.createIndex({"name": 1 }, {sparse: true})
```

Others

Handy commands

```
use admin
db.createUser({"user": "root", "pwd": passwordPrompt(), "roles": ["root"]})
db.dropUser("root")
db.auth( "user", passwordPrompt() )

use test
```

```
db.getSiblingDB("dbname")
db.currentOp()
db.killOp(123) // opid

db.fsyncLock()
db.fsyncUnlock()

db.getCollectionNames()
db.getCollectionInfos()
db.printCollectionStats()
db.stats()

db.getReplicationInfo()
db.printReplicationInfo()
db.isMaster()
db.hostInfo()
db.printShardingStatus()
db.shutdownServer()
db.serverStatus()

db.setSlaveOk()
db.getSlaveOk()

db.getProfilingLevel()
db.getProfilingStatus()
db.setProfilingLevel(1, 200) // 0 == OFF, 1 == ON with slowms, 2 == ON

db.enableFreeMonitoring()
db.disableFreeMonitoring()
db.getFreeMonitoringStatus()

db.createView("viewName", "sourceColl", [{$project:{department: 1}}])
```

Replica Set

```
rs.status()
rs.initiate({"_id": "replicaTest",
  members: [
    { _id: 0, host: "127.0.0.1:27017" },
    { _id: 1, host: "127.0.0.1:27018" },
    { _id: 2, host: "127.0.0.1:27019", arbiterOnly:true }]
})
rs.add("mongodbd1.example.net:27017")
rs.addArb("mongodbd2.example.net:27017")
rs.remove("mongodbd1.example.net:27017")
```

```
rs.conf()
rs.isMaster()
rs.printReplicationInfo()
rs.printSlaveReplicationInfo()
rs.reconfig(<valid_conf>)
rs.slaveOk()
rs.stepDown(20, 5) // (stepDownSecs, secondaryCatchUpPeriodSecs)
```

Sharded Cluster

```
sh.status()
sh.addShard("rs1/mongodbd1.example.net:27017")
sh.shardCollection("mydb.coll", {zipcode: 1})

sh.moveChunk("mydb.coll", { zipcode: "53187" }, "shard0019")
sh.splitAt("mydb.coll", {x: 70})
sh.splitFind("mydb.coll", {x: 70})
sh.disableAutoSplit()
sh.enableAutoSplit()

sh.startBalancer()
sh.stopBalancer()
sh.disableBalancing("mydb.coll")
sh.enableBalancing("mydb.coll")
sh.getBalancerState()
sh.setBalancerState(true/false)
sh.isBalancerRunning()

sh.addTagRange("mydb.coll", {state: "NY", zip: MinKey }, { state: "NY", zip: MaxKe
sh.removeTagRange("mydb.coll", {state: "NY", zip: MinKey }, { state: "NY", zip: Ma
sh.addShardTag("shard0000", "NYC")
sh.removeShardTag("shard0000", "NYC")

sh.addShardToZone("shard0000", "JFK")
sh.removeShardFromZone("shard0000", "NYC")
sh.removeRangeFromZone("mydb.coll", {a: 1, b: 1}, {a: 10, b: 10})
```

Change Streams

```
watchCursor = db.coll.watch( [ { $match : {"operationType" : "insert" } } ] )

while (!watchCursor.isExhausted()){
   if (watchCursor.hasNext()){
      print(tojson(watchCursor.next()));
```

```
    }
}
```

Related Cheatsheet

- MySQL Cheatsheet
  Quick Reference
- Neo4j Cheatsheet
  Quick Reference
- PostgreSQL Cheatsheet
  Quick Reference

Recent Cheatsheet

- Gnome Desktop Cheatsheet
  Quick Reference
- Arduino Programming Cheatsheet
  Quick Reference
- HTML Canvas Cheatsheet
  Quick Reference
- TypeScript Cheatsheet
  Quick Reference

Redis

This is a redis quick reference cheat sheet that lists examples of redis commands

Getting Started

Getting started

Start Redis

```
$ redis-server
```

Connect to redis (Redis CLI client)

```
$ redis-cli
```

Connect to redis (telnet)

```
$ telnet 127.0.0.1 6379
```

Hello World

Ping

```
redis> PING
PONG
```

Hello World

```
redis> SET mykey "Hello world"
OK
redis> GET mykey
"Hello world"
```

Basic Data types

- Strings
- Lists
- Hashes
- Sets
- Sorted Sets

Redis supports 5 basic data types

Redis String command

APPEND

```
APPEND key value
```

Example

```
redis> EXISTS mykey
(integer) 0
redis> APPEND mykey "Hello"
(integer) 5
redis> APPEND mykey " World"
(integer) 11
redis> GET mykey
"Hello World"
```

Append a value to a key

BITCOUNT

```
BITCOUNT key [start end]
```

Example

```
redis> SET mykey "foobar"
"OK"
redis> BITCOUNT mykey
(integer) 26
redis> BITCOUNT mykey 0 0
```

```
(integer) 4
redis> BITCOUNT mykey 1 1
(integer) 6
```

Count set bits in a string

BITFIELD

```
BITFIELD key [GET type offset] [SET type offset value] [INCRBY type offset
increment] [OVERFLOW WRAP|SAT|FAIL]
```

Example

```
redis> BITFIELD mykey INCRBY i5 100 1 GET u4 0
1) (integer) 1
2) (integer) 0
```

Perform arbitrary bitfield integer operations on strings

BITOP

```
BITOP operation destkey key [key ...]
```

Example

```
redis> SET key1 "foobar"
"OK"
redis> SET key2 "abcdef"
"OK"
redis> BITOP AND dest key1 key2
(integer) 6
redis> GET dest
"`bc`ab"
```

Perform bitwise operations between strings

BITPOS

```
BITPOS key bit [start] [end]
```

Example

```
redis> SET mykey "\xff\xf0\x00"
"OK"
redis> BITPOS mykey 0
```

```
(integer) 12
redis> SET mykey "\x00\xff\xf0"
"OK"
redis> BITPOS mykey 1 0
(integer) 8
redis> BITPOS mykey 1 2
(integer) 16
redis> set mykey "\x00\x00\x00"
"OK"
redis> BITPOS mykey 1
(integer) -1
```

Find first bit set or clear in a string

DECR

```
DECR key
```

Example

```
redis> SET mykey "10"
"OK"
redis> DECR mykey
(integer) 9
redis> SET mykey "234293482390480948029348230948"
"OK"
redis> DECR mykey
ERR ERR value is not an integer or out of range
```

Decrement the integer value of a key by one

DECRBY

```
DECRBY key decrement
```

Example

```
redis> SET mykey "10"
"OK"
redis> DECRBY mykey 3
(integer) 7
```

Decrement the integer value of a key by the given number

GET

```
GET key
```

Example

```
redis> GET nonexisting
(nil)
redis> SET mykey "Hello"
"OK"
redis> GET mykey
"Hello"
```

Get the value of a key

GETBIT

```
GETBIT key offset
```

Example

```
redis> SETBIT mykey 7 1
(integer) 0
redis> GETBIT mykey 0
(integer) 0
redis> GETBIT mykey 7
(integer) 1
redis> GETBIT mykey 100
(integer) 0
```

Returns the bit value at offset in the string value stored at key

GETRANGE

```
GETRANGE key start end
```

Example

```
redis> SET mykey "This is a string"
"OK"
redis> GETRANGE mykey 0 3
"This"
redis> GETRANGE mykey -3 -1
"ing"
redis> GETRANGE mykey 0 -1
"This is a string"
redis> GETRANGE mykey 10 100
```

```
"string"
```

Get a substring of the string stored at a key

GETSET

```
GETSET key value
```

Example

```
redis> INCR mycounter
(integer) 1
redis> GETSET mycounter "0"
"1"
redis> GET mycounter
"0"
```

Set the string value of a key and return its old value

INCR

```
INCR key
```

Example

```
redis> SET mykey "10"
"OK"
redis> INCR mykey
(integer) 11
redis> GET mykey
"11"
```

Increment the integer value of a key by one

MSETNX

```
MSETNX key value [key value ...]
```

Example

```
redis> MSETNX key1 "Hello" key2 "there"
(integer) 1
redis> MSETNX key2 "new" key3 "world"
(integer) 0
redis> MGET key1 key2 key3
```

```
1) "Hello"
2) "there"
3) (nil)
```

Set multiple keys to multiple values, only if none of the keys exist

INCRBYFLOAT

```
INCRBYFLOAT key increment
```

Example

```
redis> SET mykey 10.50
"OK"
redis> INCRBYFLOAT mykey 0.1
"10.6"
redis> INCRBYFLOAT mykey -5
"5.6"
redis> SET mykey 5.0e3
"OK"
redis> INCRBYFLOAT mykey 2.0e2
"5200"
```

Increment the float value of a key by the given amount

MGET

```
MGET key [key ...]
```

Example

```
redis> SET key1 "Hello"
"OK"
redis> SET key2 "World"
"OK"
redis> MGET key1 key2 nonexisting
1) "Hello"
2) "World"
3) (nil)
```

Get the values of all the given keys

MSET

```
MSET key value [key value ...]
```

Example

```
redis> MSET key1 "Hello" key2 "World"
"OK"
redis> GET key1
"Hello"
redis> GET key2
"World"
```

Set multiple keys to multiple values

INCRBY

```
INCRBY key increment
```

Example

```
redis> SET mykey "10"
"OK"
redis> INCRBY mykey 5
(integer) 15
```

Increment the integer value of a key by the given amount

PSETEX

```
PSETEX key milliseconds value
```

Example

```
redis> PSETEX mykey 1000 "Hello"
"OK"
redis> PTTL mykey
(integer) 1000
redis> GET mykey
"Hello"
```

Set the value and expiration in milliseconds of a key

SET

```
SET key value [EX seconds|PX milliseconds|KEEPTTL] [NX|XX] [GET]
```

Example

```
redis> SET mykey "Hello"
"OK"
redis> GET mykey
"Hello"
redis> SET anotherkey "will expire in a minute" EX 60
"OK"
```

Set the string value of a key

SETBIT

```
SETBIT key offset value
```

Example

```
redis> SETBIT mykey 7 1
(integer) 0
redis> SETBIT mykey 7 0
(integer) 1
redis> GET mykey
"\u0000"
```

Sets or clears the bit at offset in the string value stored at key

SETEX

```
SETEX key seconds value
```

Example

```
redis> SETEX mykey 10 "Hello"
"OK"
redis> TTL mykey
(integer) 10
redis> GET mykey
"Hello"
```

Set the value and expiration of a key

SETNX

```
SETNX key value
```

Example

```
redis> SETNX mykey "Hello"
(integer) 1
redis> SETNX mykey "World"
(integer) 0
redis> GET mykey
"Hello"
```

Set the value of a key, only if the key does not exist

SETRANGE

```
SETRANGE key offset value
```

Example

```
redis> SET key1 "Hello World"
"OK"
redis> SETRANGE key1 6 "Redis"
(integer) 11
redis> GET key1
"Hello Redis"
```

Overwrite part of a string at key starting at the specified offset

STRLEN

```
STRLEN key
```

Example

```
redis> SET mykey "Hello world"
"OK"
redis> STRLEN mykey
(integer) 11
redis> STRLEN nonexisting
(integer) 0
```

Get the length of the value stored in a key

STRALGO

```
STRALGO LCS algo-specific-argument [algo-specific-argument ...]
```

Example

```
redis> STRALGO LCS KEYS key1 key2 IDX
1) "matches"
2) 1) 1) 1) (integer) 4
         2) (integer) 7
      2) 1) (integer) 5
         2) (integer) 8
   2) 1) 1) (integer) 2
         2) (integer) 3
      2) 1) (integer) 0
         2) (integer) 1
3) "len"
4) (integer) 6
```

Run algorithms (currently LCS) against strings

Redis Set command

SADD

```
SADD key member [member ...]
```

Example

```
redis> SADD myset "Hello"
(integer) 1
redis> SADD myset "World"
(integer) 1
redis> SADD myset "World"
(integer) 0
redis> SMEMBERS myset
1) "Hello"
2) "World"
```

Add one or more members to a set

SCARD

```
SCARD key
```

Example

```
redis> SADD myset "Hello"
(integer) 1
redis> SADD myset "World"
(integer) 1
redis> SCARD myset
(integer) 2
```

Get the number of members in a set

SDIFF

```
SDIFF key [key ...]
```

Example

```
redis> SADD key1 "a"
(integer) 1
redis> SADD key1 "b"
(integer) 1
redis> SADD key1 "c"
(integer) 1
redis> SADD key2 "c"
(integer) 1
redis> SADD key2 "d"
(integer) 1
redis> SADD key2 "e"
(integer) 1
redis> SDIFF key1 key2
1) "a"
2) "b"
```

Subtract multiple sets

SDIFFSTORE

```
SDIFFSTORE destination key [key ...]
```

Example

```
redis> SADD key1 "a"
(integer) 1
redis> SADD key1 "b"
(integer) 1
redis> SADD key1 "c"
(integer) 1
```

```
redis> SADD key2 "c"
(integer) 1
redis> SADD key2 "d"
(integer) 1
redis> SADD key2 "e"
(integer) 1
redis> SDIFFSTORE key key1 key2
(integer) 2
redis> SMEMBERS key
1) "a"
2) "b"
```

Subtract multiple sets and store the resulting set in a key

SINTER

```
SINTER key [key ...]
```

Example

```
redis> SADD key1 "a"
(integer) 1
redis> SADD key1 "b"
(integer) 1
redis> SADD key1 "c"
(integer) 1
redis> SADD key2 "c"
(integer) 1
redis> SADD key2 "d"
(integer) 1
redis> SADD key2 "e"
(integer) 1
redis> SINTER key1 key2
1) "c"
```

Intersect multiple sets

SINTERSTORE

```
SINTERSTORE destination key [key ...]
```

Example

```
redis> SADD key1 "a"
(integer) 1
```

```
redis> SADD key1 "b"
(integer) 1
redis> SADD key1 "c"
(integer) 1
redis> SADD key2 "c"
(integer) 1
redis> SADD key2 "d"
(integer) 1
redis> SADD key2 "e"
(integer) 1
redis> SINTERSTORE key key1 key2
(integer) 1
redis> SMEMBERS key
1) "c"
```

Intersect multiple sets and store the resulting set in a key

SISMEMBER

```
SISMEMBER key member
```

Example

```
redis> SADD myset "one"
(integer) 1
redis> SISMEMBER myset "one"
(integer) 1
redis> SISMEMBER myset "two"
(integer) 0
```

Determine if a given value is a member of a set

SMISMEMBER

```
SMISMEMBER key member [member ...]
```

Example

```
redis> SADD myset "one"
(integer) 1
redis> SADD myset "one"
(integer) 0
redis> SMISMEMBER myset "one" "notamember"
1) (integer) 1
2) (integer) 0
```

Returns the membership associated with the given elements for a set

SMEMBERS

```
SMEMBERS key
```

Example

```
redis> SADD myset "Hello"
(integer) 1
redis> SADD myset "World"
(integer) 1
redis> SMEMBERS myset
1) "Hello"
2) "World"
```

Get all the members in a set

SMOVE

```
SMOVE source destination member
```

Example

```
redis> SADD myset "one"
(integer) 1
redis> SADD myset "two"
(integer) 1
redis> SADD myotherset "three"
(integer) 1
redis> SMOVE myset myotherset "two"
(integer) 1
redis> SMEMBERS myset
1) "one"
redis> SMEMBERS myotherset
1) "two"
2) "three"
```

Move a member from one set to another

SPOP

```
SPOP key [count]
```

Example

```
redis> SADD myset "one"
(integer) 1
redis> SADD myset "two"
(integer) 1
redis> SADD myset "three"
(integer) 1
redis> SPOP myset
"two"
redis> SMEMBERS myset
1) "one"
2) "three"
redis> SADD myset "four"
(integer) 1
redis> SADD myset "five"
(integer) 1
redis> SPOP myset 3
1) "four"
2) "five"
3) "three"
redis> SMEMBERS myset
1) "one"
```

Remove and return one or multiple random members from a set

SRANDMEMBER

```
SRANDMEMBER key [count]
```

Example

```
redis> SADD myset one two three
(integer) 3
redis> SRANDMEMBER myset
"three"
redis> SRANDMEMBER myset 2
1) "two"
2) "three"
redis> SRANDMEMBER myset -5
1) "one"
2) "two"
3) "three"
4) "three"
5) "one"
```

Get one or multiple random members from a set

SREM

```
SREM key member [member ...]
```

Example

```
redis> SADD myset "one"
(integer) 1
redis> SADD myset "two"
(integer) 1
redis> SADD myset "three"
(integer) 1
redis> SREM myset "one"
(integer) 1
redis> SREM myset "four"
(integer) 0
redis> SMEMBERS myset
1) "two"
2) "three"
```

Remove one or more members from a set

SUNION

```
SUNION key [key ...]
```

Example

```
redis> SADD key1 "a"
(integer) 1
redis> SADD key1 "b"
(integer) 1
redis> SADD key1 "c"
(integer) 1
redis> SADD key2 "c"
(integer) 1
redis> SADD key2 "d"
(integer) 1
redis> SADD key2 "e"
(integer) 1
redis> SUNION key1 key2
1) "a"
2) "c"
3) "e"
4) "b"
```

```
5) "d"
```

Add multiple sets

SUNIONSTORE

```
SUNIONSTORE destination key [key ...]
```

Example

```
redis> SADD key1 "a"
(integer) 1
redis> SADD key1 "b"
(integer) 1
redis> SADD key1 "c"
(integer) 1
redis> SADD key2 "c"
(integer) 1
redis> SADD key2 "d"
(integer) 1
redis> SADD key2 "e"
(integer) 1
redis> SUNIONSTORE key key1 key2
(integer) 5
redis> SMEMBERS key
1) "a"
2) "c"
3) "e"
4) "b"
5) "d"
```

Add multiple sets and store the resulting set in a key

Redis List command

Misc

BRPOPLPUSH	Pop an element from a list, push it to another list and return it; or block until one is available
BLMOVE	Pop an element from a list, push it to another list and return it; or block until one is available

BLPOP

```
BLPOP key [key ...] timeout
```

Example

```
redis> DEL list1 list2
(integer) 0
redis> RPUSH list1 a b c
(integer) 3
redis> BLPOP list1 list2 0
1) "list1"
2) "a"
```

Remove and get the first element in a list, or block until one is available |

BRPOP

```
BRPOP key [key ...] timeout
```

Example

```
redis> DEL list1 list2
(integer) 0
redis> RPUSH list1 a b c
(integer) 3
redis> BRPOP list1 list2 0
1) "list1"
2) "c"
```

Remove and get the last element in a list, or block until one is available |

LINDEX

```
LINDEX key index
```

Example

```
redis> LPUSH mylist "World"
(integer) 1
redis> LPUSH mylist "Hello"
(integer) 2
redis> LINDEX mylist 0
"Hello"
```

```
redis> LINDEX mylist -1
"World"
redis> LINDEX mylist 3
(nil)
```

Get an element from a list by its index

LINSERT

```
LINSERT key BEFORE|AFTER pivot element
```

Example

```
redis> RPUSH mylist "Hello"
(integer) 1
redis> RPUSH mylist "World"
(integer) 2
redis> LINSERT mylist BEFORE "World" "There"
(integer) 3
redis> LRANGE mylist 0 -1
1) "Hello"
2) "There"
3) "World"
```

Insert an element before or after another element in a list

LLEN

```
LLEN key
```

Example

```
redis> LPUSH mylist "World"
(integer) 1
redis> LPUSH mylist "Hello"
(integer) 2
redis> LLEN mylist
(integer) 2
```

Get the length of a list

LPOP

```
LPOP key [count]
```

Example

```
redis> RPUSH mylist "one"
(integer) 1
redis> RPUSH mylist "two"
(integer) 2
redis> RPUSH mylist "three"
(integer) 3
redis> LPOP mylist
"one"
redis> LRANGE mylist 0 -1
1) "two"
2) "three"
```

Remove and get the first elements in a list

LPOS

```
LPOS key element [RANK rank] [COUNT num-matches] [MAXLEN len]
```

Example

```
redis> RPUSH mylist a b c d 1 2 3 4 3 3 3
(integer) 11
redis> LPOS mylist 3
(integer) 6
redis> LPOS mylist 3 COUNT 0 RANK 2
1) (integer) 8
2) (integer) 9
3) (integer) 10
```

Return the index of matching elements on a list

LPUSH

```
LPUSH key element [element ...]
```

Example

```
redis> LPUSH mylist "world"
(integer) 1
redis> LPUSH mylist "hello"
(integer) 2
redis> LRANGE mylist 0 -1
1) "hello"
2) "world"
```

Prepend one or multiple elements to a list

LPUSHX

```
LPUSHX key element [element ...]
```

Example

```
redis> LPUSH mylist "World"
(integer) 1
redis> LPUSHX mylist "Hello"
(integer) 2
redis> LPUSHX myotherlist "Hello"
(integer) 0
redis> LRANGE mylist 0 -1
1) "Hello"
2) "World"
redis> LRANGE myotherlist 0 -1
(empty list or set)
```

Prepend an element to a list, only if the list exists

LRANGE

```
LRANGE key start stop
```

Example

```
redis> RPUSH mylist "one"
(integer) 1
redis> RPUSH mylist "two"
(integer) 2
redis> RPUSH mylist "three"
(integer) 3
redis> LRANGE mylist 0 0
1) "one"
redis> LRANGE mylist -3 2
1) "one"
2) "two"
3) "three"
redis> LRANGE mylist -100 100
1) "one"
2) "two"
3) "three"
```

```
redis> LRANGE mylist 5 10
(empty list or set)
```

Get a range of elements from a list

LREM

```
LREM key count element
```

Example

```
redis> RPUSH mylist "hello"
(integer) 1
redis> RPUSH mylist "hello"
(integer) 2
redis> RPUSH mylist "foo"
(integer) 3
redis> RPUSH mylist "hello"
(integer) 4
redis> LREM mylist -2 "hello"
(integer) 2
redis> LRANGE mylist 0 -1
1) "hello"
2) "foo"
```

Remove elements from a list

LSET

```
LSET key index element
```

Example

```
redis> RPUSH mylist "one"
(integer) 1
redis> RPUSH mylist "two"
(integer) 2
redis> RPUSH mylist "three"
(integer) 3
redis> LSET mylist 0 "four"
"OK"
redis> LSET mylist -2 "five"
"OK"
redis> LRANGE mylist 0 -1
1) "four"
```

```
2) "five"
3) "three"
```

Set the value of an element in a list by its index

LTRIM

```
LTRIM key start stop
```

Example

```
redis> RPUSH mylist "one"
(integer) 1
redis> RPUSH mylist "two"
(integer) 2
redis> RPUSH mylist "three"
(integer) 3
redis> LTRIM mylist 1 -1
"OK"
redis> LRANGE mylist 0 -1
1) "two"
2) "three"
```

Trim a list to the specified range

RPOP

```
RPOP key [count]
```

Example

```
redis> RPUSH mylist "one"
(integer) 1
redis> RPUSH mylist "two"
(integer) 2
redis> RPUSH mylist "three"
(integer) 3
redis> RPOP mylist
"three"
redis> LRANGE mylist 0 -1
1) "one"
2) "two"
```

Remove and get the last elements in a list

RPOPLPUSH

```
RPOPLPUSH source destination
```

Example

```
redis> RPUSH mylist "one"
(integer) 1
redis> RPUSH mylist "two"
(integer) 2
redis> RPUSH mylist "three"
(integer) 3
redis> RPOPLPUSH mylist myotherlist
"three"
redis> LRANGE mylist 0 -1
1) "one"
2) "two"
redis> LRANGE myotherlist 0 -1
1) "three"
```

Remove the last element in a list, prepend it to another list and return it

LMOVE

```
LMOVE source destination LEFT|RIGHT LEFT|RIGHT
```

Example

```
redis> RPUSH mylist "one"
(integer) 1
redis> RPUSH mylist "two"
(integer) 2
redis> RPUSH mylist "three"
(integer) 3
redis> LMOVE mylist myotherlist RIGHT LEFT
"three"
redis> LMOVE mylist myotherlist LEFT RIGHT
"one"
redis> LRANGE mylist 0 -1
1) "two"
redis> LRANGE myotherlist 0 -1
1) "three"
2) "one"
```

Pop an element from a list, push it to another list and return it

RPUSH

```
RPUSH key element [element ...]
```

Example

```
redis> RPUSH mylist "hello"
(integer) 1
redis> RPUSH mylist "world"
(integer) 2
redis> LRANGE mylist 0 -1
1) "hello"
2) "world"
```

Append one or multiple elements to a list

RPUSHX

```
RPUSHX key element [element ...]
```

Example

```
redis> RPUSH mylist "Hello"
(integer) 1
redis> RPUSHX mylist "World"
(integer) 2
redis> RPUSHX myotherlist "World"
(integer) 0
redis> LRANGE mylist 0 -1
1) "Hello"
2) "World"
redis> LRANGE myotherlist 0 -1
(empty list or set)
```

Append an element to a list, only if the list exists

Redis Hash command

HDEL

```
HDEL key field [field ...]
```

Example

```
redis> HSET myhash field1 "foo"
(integer) 1
redis> HDEL myhash field1
(integer) 1
redis> HDEL myhash field2
(integer) 0
```

Delete one or more hash fields

HEXISTS

```
HEXISTS key field
```

Example

```
redis> HSET myhash field1 "foo"
(integer) 1
redis> HEXISTS myhash field1
(integer) 1
redis> HEXISTS myhash field2
(integer) 0
```

Determine if a hash field exists

HGET

```
HGET key field
```

Example

```
redis> HSET myhash field1 "foo"
(integer) 1
redis> HGET myhash field1
"foo"
redis> HGET myhash field2
(nil)
```

Get the value of a hash field

HGETALL

```
HGETALL key
```

Example

```
redis> HSET myhash field1 "Hello"
(integer) 1
redis> HSET myhash field2 "World"
(integer) 1
redis> HGETALL myhash
1) "field1"
2) "Hello"
3) "field2"
4) "World"
```

Get all the fields and values in a hash

HINCRBY

```
HINCRBY key field increment
```

Example

```
redis> HSET myhash field 5
(integer) 1
redis> HINCRBY myhash field 1
(integer) 6
redis> HINCRBY myhash field -1
(integer) 5
redis> HINCRBY myhash field -10
(integer) -5
```

Increment the integer value of a hash field by the given number

HINCRBYFLOAT

```
HINCRBYFLOAT key field increment
```

Example

```
redis> HSET mykey field 10.50
(integer) 1
redis> HINCRBYFLOAT mykey field 0.1
"10.6"
redis> HINCRBYFLOAT mykey field -5
"5.6"
redis> HSET mykey field 5.0e3
(integer) 0
redis> HINCRBYFLOAT mykey field 2.0e2
```

```
"5200"
```

Increment the float value of a hash field by the given amount

HKEYS

```
HKEYS key
```

Example

```
redis> HSET myhash field1 "Hello"
(integer) 1
redis> HSET myhash field2 "World"
(integer) 1
redis> HKEYS myhash
1) "field1"
2) "field2"
```

Get all the fields in a hash

HLEN

```
HLEN key
```

Example

```
redis> HSET myhash field1 "Hello"
(integer) 1
redis> HSET myhash field2 "World"
(integer) 1
redis> HLEN myhash
(integer) 2
```

Get the number of fields in a hash

HMGET

```
HMGET key field [field ...]
```

Example

```
redis> HSET myhash field1 "Hello"
(integer) 1
redis> HSET myhash field2 "World"
(integer) 1
```

```
redis> HMGET myhash field1 field2 nofield
1) "Hello"
2) "World"
3) (nil)
```

Get the values of all the given hash fields

HMSET

```
HMSET key field value [field value ...]
```

Example

```
redis> HMSET myhash field1 "Hello" field2 "World"
"OK"
redis> HGET myhash field1
"Hello"
redis> HGET myhash field2
"World"
```

Set multiple hash fields to multiple values

HSET

```
HSET key field value [field value ...]
```

Example

```
redis> HSET myhash field1 "Hello"
(integer) 1
redis> HGET myhash field1
"Hello"
```

Set the string value of a hash field

HSETNX

```
HSETNX key field value
```

Example

```
redis> HSETNX myhash field "Hello"
(integer) 1
redis> HSETNX myhash field "World"
(integer) 0
```

```
redis> HGET myhash field
"Hello"
```

Set the value of a hash field, only if the field does not exist

HSTRLEN

```
HSTRLEN key field
```

Example

```
redis> HMSET myhash f1 HelloWorld f2 99 f3 -256
"OK"
redis> HSTRLEN myhash f1
(integer) 10
redis> HSTRLEN myhash f2
(integer) 2
redis> HSTRLEN myhash f3
(integer) 4
```

Get the length of the value of a hash field

HVALS

```
HVALS key
```

Example

```
redis> HSET myhash field1 "Hello"
(integer) 1
redis> HSET myhash field2 "World"
(integer) 1
redis> HVALS myhash
1) "Hello"
2) "World"
```

Get all the values in a hash

Redis Sorted set command

BZPOPMIN

```
BZPOPMIN key [key ...] timeout
```

Example

```
redis> DEL zset1 zset2
(integer) 0
redis> ZADD zset1 0 a 1 b 2 c
(integer) 3
redis> BZPOPMIN zset1 zset2 0
1) "zset1"
2) "a"
3) "0"
```

Remove and return the member with the lowest score from one or more sorted sets, or block until one is available

BZPOPMAX

```
BZPOPMAX key [key ...] timeout
```

Example

```
redis> DEL zset1 zset2
(integer) 0
redis> ZADD zset1 0 a 1 b 2 c
(integer) 3
redis> BZPOPMAX zset1 zset2 0
1) "zset1"
2) "c"
3) "2"
```

Remove and return the member with the highest score from one or more sorted sets, or block until one is available

ZADD

```
ZADD key [NX|XX] [GT|LT] [CH] [INCR] score member [score member ...]
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 1 "uno"
(integer) 1
redis> ZADD myzset 2 "two" 3 "three"
```

```
(integer) 2
redis> ZRANGE myzset 0 -1 WITHSCORES
1) "one"
2) "1"
3) "uno"
4) "1"
5) "two"
6) "2"
7) "three"
8) "3"
```

Add one or more members to a sorted set, or update its score if it already exists

ZCARD

```
ZCARD key
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZCARD myzset
(integer) 2
```

Get the number of members in a sorted set

ZSCORE

```
ZSCORE key member
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZSCORE myzset "one"
"1"
```

Get the score associated with the given member in a sorted set

ZCOUNT

```
ZCOUNT key min max
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZCOUNT myzset -inf +inf
(integer) 3
redis> ZCOUNT myzset (1 3
(integer) 2
```

Count the members in a sorted set with scores within the given values

ZDIFF

```
ZDIFF numkeys key [key ...] [WITHSCORES]
```

Example

```
redis> ZADD zset1 1 "one"
(integer) 1
redis> ZADD zset1 2 "two"
(integer) 1
redis> ZADD zset1 3 "three"
(integer) 1
redis> ZADD zset2 1 "one"
(integer) 1
redis> ZADD zset2 2 "two"
(integer) 1
redis> ZDIFF 2 zset1 zset2
1) "three"
redis> ZDIFF 2 zset1 zset2 WITHSCORES
1) "three"
2) "3"
```

Subtract multiple sorted sets

ZDIFFSTORE

```
ZDIFFSTORE destination numkeys key [key ...]
```

Example

```
redis> ZADD zset1 1 "one"
```

```
(integer) 1
redis> ZADD zset1 2 "two"
(integer) 1
redis> ZADD zset1 3 "three"
(integer) 1
redis> ZADD zset2 1 "one"
(integer) 1
redis> ZADD zset2 2 "two"
(integer) 1
redis> ZDIFFSTORE out 2 zset1 zset2
(integer) 1
redis> ZRANGE out 0 -1 WITHSCORES
1) "three"
2) "3"
```

Subtract multiple sorted sets and store the resulting sorted set in a new key

ZINCRBY

```
ZINCRBY key increment member
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZINCRBY myzset 2 "one"
"3"
redis> ZRANGE myzset 0 -1 WITHSCORES
1) "two"
2) "2"
3) "one"
4) "3"
```

Increment the score of a member in a sorted set

ZINTER

```
ZINTER numkeys key [key ...] [WEIGHTS weight [weight ...]] [AGGREGATE
SUM|MIN|MAX] [WITHSCORES]
```

Example

```
redis> ZADD zset1 1 "one"
```

```
(integer) 1
redis> ZADD zset1 2 "two"
(integer) 1
redis> ZADD zset2 1 "one"
(integer) 1
redis> ZADD zset2 2 "two"
(integer) 1
redis> ZADD zset2 3 "three"
(integer) 1
redis> ZINTER 2 zset1 zset2
1) "one"
2) "two"
redis> ZINTER 2 zset1 zset2 WITHSCORES
1) "one"
2) "2"
3) "two"
4) "4"
```

Intersect multiple sorted sets

ZINTERSTORE

```
ZINTERSTORE destination numkeys key [key ...] [WEIGHTS weight [weight ...]]
[AGGREGATE SUM|MIN|MAX]
```

Example

```
redis> ZADD zset1 1 "one"
(integer) 1
redis> ZADD zset1 2 "two"
(integer) 1
redis> ZADD zset2 1 "one"
(integer) 1
redis> ZADD zset2 2 "two"
(integer) 1
redis> ZADD zset2 3 "three"
(integer) 1
redis> ZINTERSTORE out 2 zset1 zset2 WEIGHTS 2 3
(integer) 2
redis> ZRANGE out 0 -1 WITHSCORES
1) "one"
2) "5"
3) "two"
4) "10"
```

Intersect multiple sorted sets and store the resulting sorted set in a new key

ZLEXCOUNT

```
ZLEXCOUNT key min max
```

Example

```
redis> ZADD myzset 0 a 0 b 0 c 0 d 0 e
(integer) 5
redis> ZADD myzset 0 f 0 g
(integer) 2
redis> ZLEXCOUNT myzset - +
(integer) 7
redis> ZLEXCOUNT myzset [b [f
(integer) 5
```

Count the number of members in a sorted set between a given lexicographical range

ZPOPMAX

```
ZPOPMAX key [count]
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZPOPMAX myzset
1) "three"
2) "3"
```

Remove and return members with the highest scores in a sorted set

ZPOPMIN

```
ZPOPMIN key [count]
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
```

```
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZPOPMIN myzset
1) "one"
2) "1"
```

Remove and return members with the lowest scores in a sorted set

ZRANGE

```
ZRANGE key start stop [WITHSCORES]
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZRANGE myzset 0 -1
1) "one"
2) "two"
3) "three"
redis> ZRANGE myzset 2 3
1) "three"
redis> ZRANGE myzset -2 -1
1) "two"
2) "three"
```

Return a range of members in a sorted set, by index

ZRANGEBYLEX

```
ZRANGEBYLEX key min max [LIMIT offset count]
```

Example

```
redis> ZADD myzset 0 a 0 b 0 c 0 d 0 e 0 f 0 g
(integer) 7
redis> ZRANGEBYLEX myzset - [c
1) "a"
2) "b"
3) "c"
```

```
redis> ZRANGEBYLEX myzset - (c
1) "a"
2) "b"
redis> ZRANGEBYLEX myzset [aaa (g
1) "b"
2) "c"
3) "d"
4) "e"
5) "f"
```

Return a range of members in a sorted set, by lexicographical range

ZREVRANGEBYLEX

```
ZREVRANGEBYLEX key max min [LIMIT offset count]
```

Example

```
redis> ZADD myzset 0 a 0 b 0 c 0 d 0 e 0 f 0 g
(integer) 7
redis> ZREVRANGEBYLEX myzset [c -
1) "c"
2) "b"
3) "a"
redis> ZREVRANGEBYLEX myzset (c -
1) "b"
2) "a"
redis> ZREVRANGEBYLEX myzset (g [aaa
1) "f"
2) "e"
3) "d"
4) "c"
5) "b"
```

Return a range of members in a sorted set, by lexicographical range, ordered from higher to lower strings.

ZRANGEBYSCORE

```
ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT offset count]
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
```

```
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZRANGEBYSCORE myzset -inf +inf
1) "one"
2) "two"
3) "three"
redis> ZRANGEBYSCORE myzset 1 2
1) "one"
2) "two"
redis> ZRANGEBYSCORE myzset (1 2
1) "two"
redis> ZRANGEBYSCORE myzset (1 (2
(empty list or set)
```

Return a range of members in a sorted set, by score

ZRANK

```
ZRANK key member
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZRANK myzset "three"
(integer) 2
redis> ZRANK myzset "four"
(nil)
```

Determine the index of a member in a sorted set

ZREM

```
ZREM key member [member ...]
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
```

```
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZREM myzset "two"
(integer) 1
redis> ZRANGE myzset 0 -1 WITHSCORES
1) "one"
2) "1"
3) "three"
4) "3"
```

Remove one or more members from a sorted set

ZREMRANGEBYLEX

```
ZREMRANGEBYLEX key min max
```

Example

```
redis> ZADD myzset 0 aaaa 0 b 0 c 0 d 0 e
(integer) 5
redis> ZADD myzset 0 foo 0 zap 0 zip 0 ALPHA 0 alpha
(integer) 5
redis> ZRANGE myzset 0 -1
1) "ALPHA"
 2) "aaaa"
 3) "alpha"
 4) "b"
 5) "c"
 6) "d"
 7) "e"
 8) "foo"
 9) "zap"
10) "zip"
redis> ZREMRANGEBYLEX myzset [alpha [omega
(integer) 6
redis> ZRANGE myzset 0 -1
1) "ALPHA"
2) "aaaa"
3) "zap"
4) "zip"
```

Remove all members in a sorted set between the given lexicographical range

ZREMRANGEBYRANK

```
ZREMRANGEBYRANK key start stop
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZREMRANGEBYRANK myzset 0 1
(integer) 2
redis> ZRANGE myzset 0 -1 WITHSCORES
1) "three"
2) "3"
```

Remove all members in a sorted set within the given indexes

ZREMRANGEBYSCORE

```
ZREMRANGEBYSCORE key min max
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZREMRANGEBYSCORE myzset -inf (2
(integer) 1
redis> ZRANGE myzset 0 -1 WITHSCORES
1) "two"
2) "2"
3) "three"
4) "3"
```

Remove all members in a sorted set within the given scores

ZREVRANGE

```
ZREVRANGE key start stop [WITHSCORES]
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZREVRANGE myzset 0 -1
1) "three"
2) "two"
3) "one"
redis> ZREVRANGE myzset 2 3
1) "one"
redis> ZREVRANGE myzset -2 -1
1) "two"
2) "one"
```

Return a range of members in a sorted set, by index, with scores ordered from high to low

ZREVRANGEBYSCORE

```
ZREVRANGEBYSCORE key max min [WITHSCORES] [LIMIT offset count]
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZREVRANGEBYSCORE myzset +inf -inf
1) "three"
2) "two"
3) "one"
redis> ZREVRANGEBYSCORE myzset 2 1
1) "two"
2) "one"
redis> ZREVRANGEBYSCORE myzset 2 (1
1) "two"
redis> ZREVRANGEBYSCORE myzset (2 (1
(empty list or set)
```

Return a range of members in a sorted set, by score, with scores ordered from high to low

ZREVRANK

```
ZREVRANK key member
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZADD myzset 3 "three"
(integer) 1
redis> ZREVRANK myzset "one"
(integer) 2
redis> ZREVRANK myzset "four"
(nil)
```

Determine the index of a member in a sorted set, with scores ordered from high to low

ZUNION

```
ZUNION numkeys key [key ...] [WEIGHTS weight [weight ...]] [AGGREGATE
SUM|MIN|MAX] [WITHSCORES]
```

Example

```
redis> ZADD zset1 1 "one"
(integer) 1
redis> ZADD zset1 2 "two"
(integer) 1
redis> ZADD zset2 1 "one"
(integer) 1
redis> ZADD zset2 2 "two"
(integer) 1
redis> ZADD zset2 3 "three"
(integer) 1
redis> ZUNION 2 zset1 zset2
1) "one"
2) "three"
3) "two"
redis> ZUNION 2 zset1 zset2 WITHSCORES
1) "one"
2) "2"
3) "three"
4) "3"
```

```
5) "two"
6) "4"
```

Add multiple sorted sets

ZMSCORE

```
ZMSCORE key member [member ...]
```

Example

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 2 "two"
(integer) 1
redis> ZMSCORE myzset "one" "two" "nofield"
1) "1"
2) "2"
3) (nil)
```

Get the score associated with the given members in a sorted set

ZUNIONSTORE

```
ZUNIONSTORE destination numkeys key [key ...] [WEIGHTS weight [weight ...]]
[AGGREGATE SUM|MIN|MAX]
```

Example

```
redis> ZADD zset1 1 "one"
(integer) 1
redis> ZADD zset1 2 "two"
(integer) 1
redis> ZADD zset2 1 "one"
(integer) 1
redis> ZADD zset2 2 "two"
(integer) 1
redis> ZADD zset2 3 "three"
(integer) 1
redis> ZUNIONSTORE out 2 zset1 zset2 WEIGHTS 2 3
(integer) 3
redis> ZRANGE out 0 -1 WITHSCORES
1) "one"
2) "5"
3) "three"
```

```
4) "9"
5) "two"
6) "10"
```

Add multiple sorted sets and store the resulting sorted set in a new key

Redis Geo command

GEOADD

```
GEOADD key longitude latitude member [longitude latitude member ...]
```

Example

```
redis> GEOADD Sicily 13.361389 38.115556 "Palermo" 15.087269 37.502669 "Catania"
(integer) 2
redis> GEODIST Sicily Palermo Catania
"166274.1516"
redis> GEORADIUS Sicily 15 37 100 km
1) "Catania"
redis> GEORADIUS Sicily 15 37 200 km
1) "Palermo"
2) "Catania"
```

Add one or more geospatial items in the geospatial index represented using a sorted set

GEOHASH

```
GEOHASH key member [member ...]
```

Example

```
redis> GEOADD Sicily 13.361389 38.115556 "Palermo" 15.087269 37.502669 "Catania"
(integer) 2
redis> GEOHASH Sicily Palermo Catania
1) "sqc8b49rny0"
2) "sqdtr74hyu0"
```

Returns members of a geospatial index as standard geohash strings

GEOPOS

```
GEOPOS key member [member ...]
```

Example

```
redis> GEOADD Sicily 13.361389 38.115556 "Palermo" 15.087269 37.502669 "Catania"
(integer) 2
redis> GEOPOS Sicily Palermo Catania NonExisting
1) 1) "13.36138933897018433"
   2) "38.11555639549629859"
2) 1) "15.08726745843887329"
   2) "37.50266842333162032"
3) (nil)
```

Returns longitude and latitude of members of a geospatial index

GEODIST

```
GEODIST key member1 member2 [m|km|ft|mi]
```

Example

```
redis> GEOADD Sicily 13.361389 38.115556 "Palermo" 15.087269 37.502669 "Catania"
(integer) 2
redis> GEODIST Sicily Palermo Catania
"166274.1516"
redis> GEODIST Sicily Palermo Catania km
"166.2742"
redis> GEODIST Sicily Palermo Catania mi
"103.3182"
redis> GEODIST Sicily Foo Bar
(nil)
```

Returns the distance between two members of a geospatial index

GEORADIUS

```
GEORADIUS key longitude latitude radius m|km|ft|mi [WITHCOORD] [WITHDIST]
[WITHHASH] [COUNT count] [ASC|DESC] [STORE key] [STOREDIST key]
```

Example

```
redis> GEOADD Sicily 13.361389 38.115556 "Palermo" 15.087269 37.502669 "Catania"
(integer) 2
redis> GEORADIUS Sicily 15 37 200 km WITHDIST
1) 1) "Palermo"
```

```
   2) "190.4424"
2) 1) "Catania"
   2) "56.4413"
redis> GEORADIUS Sicily 15 37 200 km WITHCOORD
1) 1) "Palermo"
   2) 1) "13.36138933897018433"
      2) "38.11555639549629859"
2) 1) "Catania"
   2) 1) "15.08726745843887329"
      2) "37.50266842333162032"
redis> GEORADIUS Sicily 15 37 200 km WITHDIST WITHCOORD
1) 1) "Palermo"
   2) "190.4424"
   3) 1) "13.36138933897018433"
      2) "38.11555639549629859"
2) 1) "Catania"
   2) "56.4413"
   3) 1) "15.08726745843887329"
      2) "37.50266842333162032"
```

Query a sorted set representing a geospatial index to fetch members matching a given maximum distance from a point

GEORADIUSBYMEMBER

```
GEORADIUSBYMEMBER key member radius m|km|ft|mi [WITHCOORD] [WITHDIST] [WITHHASH]
[COUNT count] [ASC|DESC] [STORE key] [STOREDIST key]
```

Example

```
redis> GEOADD Sicily 13.583333 37.316667 "Agrigento"
(integer) 1
redis> GEOADD Sicily 13.361389 38.115556 "Palermo" 15.087269 37.502669 "Catania"
(integer) 2
redis> GEORADIUSBYMEMBER Sicily Agrigento 100 km
1) "Agrigento"
2) "Palermo"
```

Query a sorted set representing a geospatial index to fetch members matching a given maximum distance from a member

GEOSEARCH

```
GEOSEARCH key [FROMMEMBER member] [FROMLONLAT longitude latitude] [BYRADIUS
radius m|km|ft|mi] [BYBOX width height m|km|ft|mi] [ASC|DESC] [COUNT count]
```

```
[WITHCOORD] [WITHDIST] [WITHHASH]
```

Example

```
redis> GEOADD Sicily 13.361389 38.115556 "Palermo" 15.087269 37.502669 "Catania"
(integer) 2
redis> GEOADD Sicily 12.758489 38.788135 "edge1"   17.241510 38.788135 "edge2"
(integer) 2
redis> GEOSEARCH Sicily FROMLONLAT 15 37 BYRADIUS 200 km ASC
1) "Catania"
2) "Palermo"
redis> GEOSEARCH Sicily FROMLONLAT 15 37 BYBOX 400 400 km ASC
1) "Catania"
2) "Palermo"
3) "edge2"
4) "edge1"
```

Query a sorted set representing a geospatial index to fetch members inside an area of a box or a circle.

Misc

GEOSEARCHSTORE	Query a sorted set representing a geospatial index to fetch members inside an area of a box or a circle, and store the result in another key.

Redis Hyperloglog command

PFADD

```
PFADD key element [element ...]
```

Example

```
redis> PFADD hll a b c d e f g
(integer) 1
redis> PFCOUNT hll
(integer) 7
```

Adds the specified elements to the specified HyperLogLog.

PFCOUNT

```
PFCOUNT key [key ...]
```

Example

```
redis> PFADD hll foo bar zap
(integer) 1
redis> PFADD hll zap zap zap
(integer) 0
redis> PFADD hll foo bar
(integer) 0
redis> PFCOUNT hll
(integer) 3
redis> PFADD some-other-hll 1 2 3
(integer) 1
redis> PFCOUNT hll some-other-hll
(integer) 6
```

Return the approximated cardinality of the set(s) observed by the HyperLogLog at key(s).

PFMERGE

```
PFMERGE destkey sourcekey [sourcekey ...]
```

Example

```
redis> PFADD hll1 foo bar zap a
(integer) 1
redis> PFADD hll2 a b c foo
(integer) 1
redis> PFMERGE hll3 hll1 hll2
"OK"
redis> PFCOUNT hll3
(integer) 6
```

Merge N different HyperLogLogs into a single one.

Redis Server command

COMMAND

```
COMMAND
```

Example

```
redis> COMMAND
1) 1) "georadius_ro"
     2) (integer) -6
     3) 1) "readonly"
        2) "movablekeys"
     4) (integer) 1
     5) (integer) 1
     6) (integer) 1
     7) 1) "@read"
        2) "@geo"
        3) "@slow"
  2) 1) "zpopmin"
     2) (integer) -2
     3) 1) "write"
        2) "fast"
  ........
```

Get array of Redis command details

Misc

ACL LOAD	Reload the ACLs from the configured ACL file
ACL SAVE	Save the current ACL rules in the configured ACL file
ACL LIST	List the current ACL rules in ACL config file format
ACL USERS	List the username of all the configured ACL rules
ACL GETUSER	Get the rules for a specific ACL user
ACL SETUSER	Modify or create the rules for a specific ACL user
ACL DELUSER	Remove the specified ACL users and the associated rules
ACL CAT	List the ACL categories or the commands inside a category
ACL GENPASS	Generate a pseudorandom secure password to use for ACL users
ACL WHOAMI	Return the name of the user associated to the current connection
ACL LOG	List latest events denied because of ACLs in place
ACL HELP	Show helpful text about the different subcommands
BGREWRITEAOF	Asynchronously rewrite the append-only file

BGSAVE	Asynchronously save the dataset to disk
CONFIG GET	Get the value of a configuration parameter
CONFIG REWRITE	Rewrite the configuration file with the in memory configuration
CONFIG SET	Set a configuration parameter to the given value
CONFIG RESETSTAT	Reset the stats returned by INFO
DBSIZE	Return the number of keys in the selected database
DEBUG OBJECT	Get debugging information about a key
DEBUG SEGFAULT	Make the server crash
FLUSHALL	Remove all keys from all databases
FLUSHDB	Remove all keys from the current database
LOLWUT	Display some computer art and the Redis version
LASTSAVE	Get the UNIX time stamp of the last successful save to disk
MEMORY DOCTOR	Outputs memory problems report
MEMORY HELP	Show helpful text about the different subcommands
MEMORY MALLOC-STATS	Show allocator internal stats
MEMORY PURGE	Ask the allocator to release memory
MEMORY STATS	Show memory usage details
MEMORY USAGE	Estimate the memory usage of a key
MODULE LIST	List all modules loaded by the server
MODULE LOAD	Load a module
MODULE UNLOAD	Unload a module
MONITOR	Listen for all requests received by the server in real time
SAVE	Synchronously save the dataset to disk
SHUTDOWN	Synchronously save the dataset to disk and then shut down the server

SLAVEOF	Make the server a replica of another instance, or promote it as master. Deprecated starting with Redis 5. Use REPLICAOF instead.
REPLICAOF	Make the server a replica of another instance, or promote it as master.
SLOWLOG	Manages the Redis slow queries log
SWAPDB	Swaps two Redis databases
SYNC	Internal command used for replication
PSYNC	Internal command used for replication
LATENCY DOCTOR	Return a human readable latency analysis report.
LATENCY GRAPH	Return a latency graph for the event.
LATENCY HISTORY	Return timestamp-latency samples for the event.
LATENCY LATEST	Return the latest latency samples for all events.
LATENCY RESET	Reset latency data for one or more events.
LATENCY HELP	Show helpful text about the different subcommands.

COMMAND COUNT

```
COMMAND COUNT
```

Example

```
redis> COMMAND COUNT
(integer) 217
```

Get total number of Redis commands

COMMAND GETKEYS

```
COMMAND GETKEYS
```

Example

```
redis> COMMAND GETKEYS MSET a b c d e f
1) "a"
2) "c"
3) "e"
redis> COMMAND GETKEYS EVAL "not consulted" 3 key1 key2 key3 arg1 arg2 arg3 argN
```

```
1) "key1"
2) "key2"
3) "key3"
redis> COMMAND GETKEYS SORT mylist ALPHA STORE outlist
1) "mylist"
2) "outlist"
```

Extract keys given a full Redis command

COMMAND INFO

```
COMMAND INFO command-name [command-name ...]
```

Example

```
redis> COMMAND INFO get set eval
1) 1) "get"
   2) (integer) 2
   3) 1) "readonly"
      2) "fast"
   4) (integer) 1
   5) (integer) 1
   6) (integer) 1
   7) 1) "@read"
      2) "@string"
      3) "@fast"
2) 1) "set"
   2) (integer) -3
   3) 1) "write"
      2) "denyoom"
   4) (integer) 1
   5) (integer) 1
   6) (integer) 1
   7) 1) "@write"
      2) "@string"
      3) "@slow"
3) 1) "eval"
   2) (integer) -3
   3) 1) "noscript"
      2) "movablekeys"
   4) (integer) 0
   5) (integer) 0
   6) (integer) 0
   7) 1) "@slow"
      2) "@scripting"
```

Get array of specific Redis command details

INFO

```
INFO [section]
```

Example

```
redis> INFO
# Server
redis_version:6.1.240
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:a26db646ea64a07c
redis_mode:standalone
os:Linux 5.4.0-1017-aws x86_64
......
```

Get information and statistics about the server

ROLE

```
ROLE
```

Example

```
redis> ROLE
1) "master"
2) (integer) 0
3) (empty list or set)
```

Return the role of the instance in the context of replication

TIME

```
TIME
```

Example

```
redis> TIME
1) "1609040690"
2) "558952"
redis> TIME
1) "1609040690"
2) "559206"
```

Return the current server time

Redis Generic command

Misc

COPY	Copy a key
MIGRATE	Atomically transfer a key from a Redis instance to another one.
MOVE	Move a key to another database
OBJECT	Inspect the internals of Redis objects
RESTORE	Create a key using the provided serialized value, previously obtained using DUMP.
SORT	Sort the elements in a list, set or sorted set
WAIT	Wait for the synchronous replication of all the write commands sent in the context of the current connection
SCAN	Incrementally iterate the keys space

DEL

```
DEL key [key ...]
```

Example

```
redis> SET key1 "Hello"
"OK"
redis> SET key2 "World"
"OK"
redis> DEL key1 key2 key3
(integer) 2
```

Delete a key

DUMP

```
DUMP key
```

Example

```
redis> SET mykey 10
"OK"
redis> DUMP mykey
"\u0000\xC0\n\t\u0000\xBEm\u0006\x89Z(\u0000\n"
```

Return a serialized version of the value stored at the specified key.

EXISTS

```
EXISTS key [key ...]
```

Example

```
redis> SET key1 "Hello"
"OK"
redis> EXISTS key1
(integer) 1
redis> EXISTS nosuchkey
(integer) 0
redis> SET key2 "World"
"OK"
redis> EXISTS key1 key2 nosuchkey
(integer) 2
```

Determine if a key exists

EXPIRE

```
EXPIRE key seconds
```

Example

```
redis> SET mykey "Hello"
"OK"
redis> EXPIRE mykey 10
(integer) 1
redis> TTL mykey
(integer) 10
redis> SET mykey "Hello World"
"OK"
redis> TTL mykey
(integer) -1
```

Set a key's time to live in seconds

EXPIREAT

```
EXPIREAT key timestamp
```

Example

```
redis> SET mykey "Hello"
"OK"
redis> EXISTS mykey
(integer) 1
redis> EXPIREAT mykey 1293840000
(integer) 1
redis> EXISTS mykey
(integer) 0
```

Set the expiration for a key as a UNIX timestamp

KEYS

```
KEYS pattern
```

Example

```
redis> MSET firstname Jack lastname Stuntman age 35
"OK"
redis> KEYS *name*
1) "firstname"
2) "lastname"
redis> KEYS a??
1) "age"
redis> KEYS *
1) "firstname"
2) "age"
3) "lastname"
```

Find all keys matching the given pattern

PERSIST

```
PERSIST key
```

Example

```
redis> SET mykey "Hello"
```

```
"OK"
redis> EXPIRE mykey 10
(integer) 1
redis> TTL mykey
(integer) 10
redis> PERSIST mykey
(integer) 1
redis> TTL mykey
(integer) -1
```

Remove the expiration from a key

PEXPIRE

```
PEXPIRE key milliseconds
```

Example

```
redis> SET mykey "Hello"
"OK"
redis> PEXPIRE mykey 1500
(integer) 1
redis> TTL mykey
(integer) 1
redis> PTTL mykey
(integer) 1499
```

Set a key's time to live in milliseconds

PEXPIREAT

```
PEXPIREAT key milliseconds-timestamp
```

Example

```
redis> SET mykey "Hello"
"OK"
redis> PEXPIREAT mykey 1555555555005
(integer) 1
redis> TTL mykey
(integer) -2
redis> PTTL mykey
(integer) -2
```

Set the expiration for a key as a UNIX timestamp specified in milliseconds

PTTL

```
PTTL key
```

Example

```
redis> SET mykey "Hello"
"OK"
redis> EXPIRE mykey 1
(integer) 1
redis> PTTL mykey
(integer) 1000
```

Get the time to live for a key in milliseconds

RENAME

```
RENAME key newkey
```

Example

```
redis> SET mykey "Hello"
"OK"
redis> RENAME mykey myotherkey
"OK"
redis> GET myotherkey
"Hello"
```

Rename a key

RENAMENX

```
RENAMENX key newkey
```

Example

```
redis> SET mykey "Hello"
"OK"
redis> SET myotherkey "World"
"OK"
redis> RENAMENX mykey myotherkey
(integer) 0
redis> GET myotherkey
"World"
```

Rename a key, only if the new key does not exist

TOUCH

```
TOUCH key [key ...]
```

Example

```
redis> SET key1 "Hello"
"OK"
redis> SET key2 "World"
"OK"
redis> TOUCH key1 key2
(integer) 2
```

Alters the last access time of a key(s). Returns the number of existing keys specified.

TTL

```
TTL key
```

Example

```
redis> SET mykey "Hello"
"OK"
redis> EXPIRE mykey 10
(integer) 1
redis> TTL mykey
(integer) 10
```

Get the time to live for a key

TYPE

```
TYPE key
```

Example

```
redis> SET key1 "value"
"OK"
redis> LPUSH key2 "value"
(integer) 1
redis> SADD key3 "value"
(integer) 1
redis> TYPE key1
"string"
```

```
redis> TYPE key2
"list"
redis> TYPE key3
"set"
```

Determine the type stored at key

UNLINK

```
UNLINK key [key ...]
```

Example

```
redis> SET key1 "Hello"
"OK"
redis> SET key2 "World"
"OK"
redis> UNLINK key1 key2 key3
(integer) 2
```

Delete a key asynchronously in another thread. Otherwise it is just as DEL, but non blocking.

Redis Connection command

Misc

AUTH	Authenticate to the server
CLIENT CACHING	Instruct the server about tracking or not keys in the next request
CLIENT KILL	Kill the connection of a client
CLIENT LIST	Get the list of client connections
CLIENT GETNAME	Get the current connection name
CLIENT GETREDIR	Get tracking notifications redirection client ID if any
CLIENT PAUSE	Stop processing commands from clients for some time
CLIENT REPLY	Instruct the server whether to reply to commands
CLIENT SETNAME	Set the current connection name

CLIENT TRACKING	Enable or disable server assisted client side caching support
CLIENT UNBLOCK	Unblock a client blocked in a blocking command from a different connection
HELLO	switch Redis protocol
QUIT	Close the connection
RESET	Reset the connection
SELECT	Change the selected database for the current connection

CLIENT ID

```
CLIENT ID
```

Example

```
redis> CLIENT ID
ERR Unknown or disabled command 'CLIENT'
```

Returns the client ID for the current connection

CLIENT INFO

```
CLIENT INFO
```

Example

```
redis> CLIENT INFO
"id=55542 addr=127.0.0.1:58710 laddr=127.0.0.1:6379 fd=8 name= age=114920 idle=0 f
```

Returns information about the current client connection.

ECHO

```
ECHO message
```

Example

```
redis> ECHO "Hello World!"
"Hello World!"
```

Echo the given string

PING

```
PING [message]
```

Example

```
redis> PING
"PONG"
redis> PING "hello world"
"hello world"
```

Ping the server

Redis Stream command

Misc

XINFO	Get information on streams and consumer groups
XDEL	Removes the specified entries from the stream. Returns the number of items actually deleted, that may be different from the number of IDs passed in case certain IDs do not exist.
XREAD	Return never seen elements in multiple streams, with IDs greater than the ones reported by the caller for each stream. Can block.
XGROUP	Create, destroy, and manage consumer groups.
XREADGROUP	Return new entries from a stream using a consumer group, or access the history of the pending entries for a given consumer. Can block.
XCLAIM	Changes (or acquires) ownership of a message in a consumer group, as if the message was delivered to the specified consumer.
XPENDING	Return information and entries from a stream consumer group pending entries list, that are messages fetched but never acknowledged.

XADD

```
XADD key [MAXLEN [=|~] length] [NOMKSTREAM] *|ID field value [field value ...]
```

Example

```
redis> XADD mystream * name Sara surname OConnor
"1609040574632-0"
redis> XADD mystream * field1 value1 field2 value2 field3 value3
"1609040574632-1"
redis> XLEN mystream
(integer) 2
redis> XRANGE mystream - +
1) 1) "1609040574632-0"
   2) 1) "name"
      2) "Sara"
      3) "surname"
      4) "OConnor"
2) 1) "1609040574632-1"
   2) 1) "field1"
      2) "value1"
      3) "field2"
      4) "value2"
      5) "field3"
      6) "value3"
```

Appends a new entry to a stream

XTRIM

```
XTRIM key MAXLEN [=|~] length
```

Example

```
redis> XADD mystream * field1 A field2 B field3 C field4 D
"1609040575750-0"
redis> XTRIM mystream MAXLEN 2
(integer) 0
redis> XRANGE mystream - +
1) 1) "1609040575750-0"
   2) 1) "field1"
      2) "A"
      3) "field2"
      4) "B"
      5) "field3"
      6) "C"
      7) "field4"
      8) "D"
```

Trims the stream to (approximately if '~' is passed) a certain size

XRANGE

```
XRANGE key start end [COUNT count]
```

Example

```
redis> XADD writers * name Virginia surname Woolf
"1609040578002-0"
redis> XADD writers * name Jane surname Austen
"1609040578002-1"
redis> XADD writers * name Toni surname Morrison
"1609040578003-0"
redis> XADD writers * name Agatha surname Christie
"1609040578003-1"
redis> XADD writers * name Ngozi surname Adichie
"1609040578003-2"
redis> XLEN writers
(integer) 5
redis> XRANGE writers - + COUNT 2
1) 1) "1609040578002-0"
   2) 1) "name"
      2) "Virginia"
      3) "surname"
      4) "Woolf"
2) 1) "1609040578002-1"
   2) 1) "name"
      2) "Jane"
      3) "surname"
      4) "Austen"
```

Return a range of elements in a stream, with IDs matching the specified IDs interval

XREVRANGE

```
XREVRANGE key end start [COUNT count]
```

Example

```
redis> XADD writers * name Virginia surname Woolf
"1609040579130-0"
redis> XADD writers * name Jane surname Austen
"1609040579130-1"
redis> XADD writers * name Toni surname Morrison
"1609040579130-2"
redis> XADD writers * name Agatha surname Christie
"1609040579131-0"
```

```
redis> XADD writers * name Ngozi surname Adichie
"1609040579131-1"
redis> XLEN writers
(integer) 5
redis> XREVRANGE writers + - COUNT 1
1) 1) "1609040579131-1"
   2) 1) "name"
      2) "Ngozi"
      3) "surname"
      4) "Adichie"
```

Return a range of elements in a stream, with IDs matching the specified IDs interval, in reverse order (from greater to smaller IDs) compared to XRANGE

XLEN

```
XLEN key
```

Example

```
redis> XADD mystream * item 1
"1609040580250-0"
redis> XADD mystream * item 2
"1609040580250-1"
redis> XADD mystream * item 3
"1609040580251-0"
redis> XLEN mystream
(integer) 3
```

Return the number of entries in a stream

XACK

```
XACK key group ID [ID ...]
```

Example

```
redis> XACK mystream mygroup 1526569495631-0
ERR Unknown or disabled command 'XACK'
```

Marks a pending message as correctly processed, effectively removing it from the pending entries list of the consumer group. Return value of the command is the number of messages successfully acknowledged, that is, the IDs we were actually able to resolve in the PEL.

Miscellaneous

Cluster	
CLUSTER ADDSLOTS	Assign new hash slots to receiving node
CLUSTER BUMPEPOCH	Advance the cluster config epoch
CLUSTER COUNT-FAILURE-REPORTS	Return the number of failure reports active for a given node
CLUSTER COUNTKEYSINSLOT	Return the number of local keys in the specified hash slot
CLUSTER DELSLOTS	Set hash slots as unbound in receiving node
CLUSTER FAILOVER	Forces a replica to perform a manual failover of its master.
CLUSTER FLUSHSLOTS	Delete a node's own slots information
CLUSTER FORGET	Remove a node from the nodes table
CLUSTER GETKEYSINSLOT	Return local key names in the specified hash slot
CLUSTER INFO	Provides info about Redis Cluster node state
CLUSTER KEYSLOT	Returns the hash slot of the specified key
CLUSTER MEET	Force a node cluster to handshake with another node
CLUSTER MYID	Return the node id
CLUSTER NODES	Get Cluster config for the node
CLUSTER REPLICATE	Reconfigure a node as a replica of the specified master node
CLUSTER RESET	Reset a Redis Cluster node
CLUSTER SAVECONFIG	Forces the node to save cluster state on disk
CLUSTER SET-CONFIG-EPOCH	Set the configuration epoch in a new node
CLUSTER SETSLOT	Bind a hash slot to a specific node
CLUSTER SLAVES	List replica nodes of the specified master node

CLUSTER REPLICAS	List replica nodes of the specified master node
CLUSTER SLOTS	Get array of Cluster slot to node mappings
READONLY	Enables read queries for a connection to a cluster replica node
READWRITE	Disables read queries for a connection to a cluster replica node

Transactions

DISCARD	Discard all commands issued after MULTI
EXEC	Execute all commands issued after MULTI
MULTI	Mark the start of a transaction block
UNWATCH	Forget about all watched keys
WATCH	Watch the given keys to determine execution of the MULTI/EXEC block

Scripting

EVAL	Execute a Lua script server side
EVALSHA	Execute a Lua script server side
SCRIPT DEBUG	Set the debug mode for executed scripts.
SCRIPT EXISTS	Check existence of scripts in the script cache.
SCRIPT FLUSH	Remove all the scripts from the script cache.
SCRIPT KILL	Kill the script currently in execution.
SCRIPT LOAD	Load the specified Lua script into the script cache.

Pubsub

PSUBSCRIBE	Listen for messages published to channels matching the given patterns
PUBSUB	Inspect the state of the Pub/Sub subsystem
PUBLISH	Post a message to a channel
PUNSUBSCRIBE	Stop listening for messages posted to channels matching the given patterns
SUBSCRIBE	Listen for messages published to the given channels

UNSUBSCRIBE	Stop listening for messages posted to the given channels

Top Cheatsheet

- Python Cheatsheet
  Quick Reference
- Vim Cheatsheet
  Quick Reference
- JavaScript Cheatsheet
  Quick Reference
- Bash Cheatsheet
  Quick Reference

Recent Cheatsheet

- Gnome Desktop Cheatsheet
  Quick Reference
- Arduino Programming Cheatsheet
  Quick Reference
- HTML Canvas Cheatsheet
  Quick Reference
- TypeScript Cheatsheet
  Quick Reference

Docker

This is a quick reference cheat sheet for Docker. And you can find the most common Docker commands here.

Getting Started

Getting started

Create and run a container in background

```
$ docker run -d -p 80:80 docker/getting-started
```

- `-d` - Run the container in detached mode
- `-p 80:80` - Map port 80 to port 80 in the container
- `docker/getting-started` - The image to use

Create and run a container in foreground

```
$ docker run -it -p 8001:8080 --name my-nginx nginx
```

- `-it` - Interactive bash mode
- `-p 8001:8080` - Map port 8001 to port 8080 in the container
- `--name my-nginx` - Specify a name
- `nginx` - The image to use

General commands

`docker ps`	List running containers
`docker ps -a`	List all containers

`docker ps -s`	List running containers (with CPU / memory)
`docker images`	List all images
`docker exec -it <container> bash`	Connecting to container
`docker logs <container>`	Shows container's console log
`docker stop <container>`	Stop a container
`docker restart <container>`	Restart a container
`docker rm <container>`	Remove a container
`docker port <container>`	Shows container's port mapping
`docker top <container>`	List processes
`docker kill <container>`	Kill a container

Parameter `<container>` can be container id or name

Docker Containers

Starting & Stopping

`docker start my-nginx`	Starting
`docker stop my-nginx`	Stopping
`docker restart my-nginx`	Restarting
`docker pause my-nginx`	Pausing
`docker unpause my-nginx`	Unpausing
`docker wait my-nginx`	Blocking a Container
`docker kill my-nginx`	Sending a SIGKILL
`docker attach my-nginx`	Connecting to an Existing Container

Information

`docker ps`	List running containers
`docker ps -a`	List all containers
`docker logs my-nginx`	Container Logs
`docker inspect my-nginx`	Inspecting Containers
`docker events my-nginx`	Containers Events
`docker port my-nginx`	Public Ports
`docker top my-nginx`	Running Processes
`docker stats my-nginx`	Container Resource Usage
`docker diff my-nginx`	Lists the changes made to a container.

Creating

```
docker create [options] IMAGE
  -a, --attach               # attach stdout/err
  -i, --interactive          # attach stdin (interactive)
  -t, --tty                  # pseudo-tty
      --name NAME            # name your image
  -p, --publish 5000:5000    # port map (host:container)
      --expose 5432          # expose a port to containers
  -P, --publish-all          # publish all ports
      --link container:alias # linking
  -v, --volume `pwd`:/app    # mount (absolute paths needed)
  -e, --env NAME=hello       # env vars
```

Example

```
$ docker create --name my_redis --expose 6379 redis:3.0.2
```

Manipulating

Renaming a Container

```
docker rename my-nginx my-nginx
```

Removing a Container

```
docker rm my-nginx
```

Updating a Container

```
docker update --cpu-shares 512 -m 300M my-nginx
```

Docker Images

Manipulating

`docker images`	Listing images
`docker rmi nginx`	Removing an image
`docker load < ubuntu.tar.gz`	Loading a tarred repository
`docker load --input ubuntu.tar`	Loading a tarred repository
`docker save busybox > ubuntu.tar`	Save an image to a tar archive
`docker history`	Showing the history of an image
`docker commit nginx`	Save a container as an image.
`docker tag nginx eon01/nginx`	Tagging an image
`docker push eon01/nginx`	Pushing an image

Building Images

```
$ docker build .
$ docker build github.com/creack/docker-firefox
$ docker build - < Dockerfile
$ docker build - < context.tar.gz
$ docker build -t eon/my-nginx .
$ docker build -f myOtherDockerfile .
$ curl example.com/remote/Dockerfile | docker build -f - .
```

Docker Networking

Manipulating

Removing a network

```
docker network rm MyOverlayNetwork
```

Listing networks

```
docker network ls
```

Getting information about a network

```
docker network inspect MyOverlayNetwork
```

Connecting a running container to a network

```
docker network connect MyOverlayNetwork nginx
```

Connecting a container to a network when it starts

```
docker run -it -d --network=MyOverlayNetwork nginx
```

Disconnecting a container from a network

```
docker network disconnect MyOverlayNetwork nginx
```

Creating Networks

```
docker network create -d overlay MyOverlayNetwork

docker network create -d bridge MyBridgeNetwork

docker network create -d overlay \
  --subnet=192.168.0.0/16 \
  --subnet=192.170.0.0/16 \
  --gateway=192.168.0.100 \
  --gateway=192.170.0.100 \
  --ip-range=192.168.1.0/24 \
  --aux-address="my-router=192.168.1.5" \
  --aux-address="my-switch=192.168.1.6" \
  --aux-address="my-printer=192.170.1.5" \
  --aux-address="my-nas=192.170.1.6" \
  MyOverlayNetwork
```

Clean Up

Clean All

Cleans up dangling images, containers, volumes, and networks (ie, not associated with a container)

```
docker system prune
```

Additionally, remove any stopped containers and all unused images (not just dangling images)

```
docker system prune -a
```

Containers

Stop all running containers

```
docker stop $(docker ps -a -q)
```

Delete stopped containers

```
docker container prune
```

Images

Remove all dangling (not tagged and is not associated with a container) images:

```
docker image prune
```

Remove all images which are not used by existing containers

```
docker image prune -a
```

Volumes

```
docker volume prune
```

Remove all volumes not used by at least one container

Miscellaneous

Docker Hub

`docker search search_word`	Search docker hub for images.
`docker pull user/image`	Downloads an image from docker hub.
`docker login`	Authenticate to docker hub
`docker push user/image`	Uploads an image to docker hub.

Registry commands

Login to a Registry

```
$ docker login
$ docker login localhost:8080
```

Logout from a Registry

```
$ docker logout
$ docker logout localhost:8080
```

Searching an Image

```
$ docker search nginx
$ docker search nginx --stars=3 --no-trunc busybox
```

Pulling an Image

```
$ docker pull nginx
$ docker pull eon01/nginx localhost:5000/myadmin/nginx
```

Pushing an Image

```
$ docker push eon01/nginx
$ docker push eon01/nginx localhost:5000/myadmin/nginx
```

Batch clean

`docker stop -f $(docker ps -a -q)`	Stopping all containers
`docker rm -f $(docker ps -a -q)`	Removing all containers
`docker rmi -f $(docker images -q)`	Removing all images

Volumes

Check volumes

```
$ docker volume ls
```

Cleanup unused volumes

```
$ docker volume prune
```

Related Cheatsheet

Terraform Cheatsheet
Quick Reference

Recent Cheatsheet

Gnome Desktop Cheatsheet
Quick Reference

Arduino Programming Cheatsheet
Quick Reference

HTML Canvas Cheatsheet
Quick Reference

TypeScript Cheatsheet
Quick Reference

Kubernetes

This page contains a list of commonly used kubectl commands and flags.

Viewing and finding resources

Nodes

```
kubectl get no # Display all node information
kubectl get no -o wide # Show more information about all nodes
kubectl describe no # Display node details
kubectl get no -o yaml # Display node details in yaml format
kubectl get node --selector=[label_name] # Filter the node with the specified labe
kubectl get nodes -o jsonpath='{.items[*].status.addresses[?(@.type="ExternalIP")]
# Output the field information defined by the jsonpath expression
kubectl top node [node_name] # Display node (CPU/memory/storage) usage
```

Resource name: nodes, abbreviation: no

Pods

```
kubectl get po # Display all container group information
kubectl get po -o wide
kubectl describe po
kubectl get po --show-labels # View the labels of the container group
kubectl get po -l app=nginx
kubectl get po -o yaml
kubectl get pod [pod_name] -o yaml --export
kubectl get pod [pod_name] -o yaml --export > nameoffile.yaml
# Export container group information to yaml file in yaml format
kubectl get pods --field-selector status.phase=Running
# Use the field selector to filter out container group information
```

Resource name: pods, abbreviation: po

Namespaces

```
kubectl get ns
kubectl get ns -o yaml
kubectl describe ns
```

Resource name: namespaces, abbreviation: ns

Deployments

```
kubectl get deploy
kubectl describe deploy
kubectl get deploy -o wide
kubectl get deploy -o yaml
```

Resource name: deployments, abbreviation: deploy

Services

```
kubectl get svc
kubectl describe svc
kubectl get svc -o wide
kubectl get svc -o yaml
kubectl get svc --show-labels
```

Resource name: services, abbreviation: svc

Daemon Sets

```
kubectl get ds
kubectl describe ds --all-namespaces
kubectl describe ds [daemonset_name] -n [namespace_name]
kubectl get ds [ds_name] -n [ns_name] -o yaml
```

Resource name: daemonsets, abbreviation: ds

Events

```
kubectl get events
kubectl get events -n kube-system
kubectl get events -w
```

Resource name: events, abbreviation: ev

Logs

```
kubectl logs [pod_name]
kubectl logs --since=1h [pod_name]
kubectl logs --tail=20 [pod_name]
kubectl logs -f -c [container_name] [pod_name]
kubectl logs [pod_name] > pod.log
```

Service Accounts

```
kubectl get sa
kubectl get sa -o yaml
kubectl get serviceaccounts default -o yaml >./sa.yaml
kubectl replace serviceaccount default -f ./sa.yaml
```

Resource name: serviceaccounts, abbreviation: ev

Replica Sets

```
kubectl get rs
kubectl describe rs
kubectl get rs -o wide
kubectl get rs -o yaml
```

Resource name: replicasets, abbreviation: rs

Roles

```
kubectl get roles --all-namespaces
kubectl get roles --all-namespaces -o yaml
```

Secrets

```
kubectl get secrets
kubectl get secrets --all-namespaces
kubectl get secrets -o yaml
```

Config maps

Resource name: configmaps, abbreviation: cm

```
kubectl get cm
kubectl get cm --all-namespaces
kubectl get cm --all-namespaces -o yaml
```

Ingresses

Resource name: ingresses, abbreviation: ing

```
kubectl get ing
kubectl get ing --all-namespaces
```

Persistent Volumes

Resource name: persistentvolumes, abbreviation: pv

```
kubectl get pv
kubectl describe pv
```

Persistent volume declaration

Resource name: persistentvolumeclaims, abbreviation: pvc

```
kubectl get pvc
kubectl describe pvc
```

storage class

Resource name: storageclasses, Abbreviation: sc

```
kubectl get sc
kubectl get sc -o yaml
```

Multiple resources

```
kubectl get svc, po
kubectl get deploy, no
kubectl get all
kubectl get all --all-namespaces
```

Updating resources

Taint

```
kubectl taint [node_name] [taint_name]
```

Label

```
kubectl label [node_name] disktype=ssd
kubectl label [pod_name] env=prod
```

Maintain/Schedulable

```
kubectl cordon [node_name] # node maintenance
kubectl uncordon [node_name] # node is schedulable
```

clear

```
kubectl drain [node_name] # empty the node
```

Node/Pod

```
kubectl delete node [node_name]
kubectl delete pod [pod_name]
kubectl edit node [node_name]
kubectl edit pod [pod_name]
```

Stateless/Namespaced

```
kubectl edit deploy [deploy_name]
kubectl delete deploy [deploy_name]
kubectl expose deploy [deploy_name] --port=80 --type=NodePort
kubectl scale deploy [deploy_name] --replicas=5
kubectl delete ns
kubectl edit ns [ns_name]
```

Service

```
kubectl edit svc [svc_name]
kubectl delete svc [svc_name]
```

Daemon set

```
kubectl edit ds [ds_name] -n kube-system
kubectl delete ds [ds_name]
```

Service account

```
kubectl edit sa [sa_name]
```

```
kubectl delete sa [sa_name]
```

Notes

```
kubectl annotate po [pod_name] [annotation]
kubectl annotateno [node_name]
```

Create resources

Create pod

```
kubectl create -f [name_of_file]
kubectl apply -f [name_of_file]
kubectl run [pod_name] --image=nginx --restart=Never
kubectl run [pod_name] --generator=run-pod/v1 --image=nginx
kubectl run [pod_name] --image=nginx --restart=Never
```

Create Service

```
kubectl create svc nodeport [svc_name] --tcp=8080:80
```

Create a stateless application

```
kubectl create -f [name_of_file]
kubectl apply -f [name_of_file]
kubectl create deploy [deploy_name] --image=nginx
```

interaction

```
kubectl run [pod_name] --image=busybox --rm -it --restart=Never --sh
```

Output YAML

```
kubectl create deploy [deploy_name] --image=nginx --dry-run -o yaml > deploy.yaml
kubectl get po [pod_name] -o yaml --export > pod.yaml
```

Help

```
kubectl -h
kubectl create -h
```

```
kubectl run -h
kubectl explain deploy.spec
```

Miscellaneous

APIs

```
kubectl get --raw /apis/metrics.k8s.io/
```

Information

```
kubectl config
kubectl cluster-info
kubectl get componentstatus
```

Also See

Kubernetes Official Documentation (kubernetes.io)

Related Cheatsheet

ES6 Cheatsheet
Quick Reference

Express Cheatsheet
Quick Reference

JSON Cheatsheet
Quick Reference

TOML Cheatsheet
Quick Reference

Recent Cheatsheet

Gnome Desktop Cheatsheet
Quick Reference

Arduino Programming Cheatsheet
Quick Reference

HTML Canvas Cheatsheet
Quick Reference

TypeScript Cheatsheet
Quick Reference

Terraform

This is a quick reference cheat sheet for Terraform.

HCL Syntax:

```
(Block Name)  (Resource Type)  (Resource Name)
      |           |                 |
      ▽           ▽                 ▽
  resource "aws_instance" "my_aws_server" {
                                          --
      ami           = "ami-1251351351513"    |<--(Arguments)
      instance_type = "t2.micro"             |
                                          --

}
```

Variable Types

- Simple types: number, string, bool, null.
- Complex types: Collection types: list, map, set Structural types: `object({<KEY> = <TYPE>, ...})`, `tuple([<TYPE>, ...])`

Order of Precedence: defaults < env vars < terraform.tfvars file < terraform.tfvars.json file < .auto.tfvars < command line (-var & -var-file)

```
variable "<VAR NAME>" {
    description = "<DESCRIPTION OF THE VAR>"
    type        = <VAR TYPE>
    default     = <DEFAULT VALUE>
}

# type string
```

```
variable "aws_region" {
    description = "AWS Region"
    type        = string
    default     = "us-east-1"
}

# type list(string)
variable "azs" {
    description = "AZs in the Region"
    type        = list(string)
    default     = [ "us-east-1a", "us-east-1b", "us-east-1c"]
}

# type map
variable "amis" {
    type = map(string)
    default = {
      "eu-west-1" = "ami-0fdke15151513145",
      "us-east-1" = "ami-0d17359173587519"
    }
}

# type object
variable "egress_dsg" {
    type = object({
        from_port = number
        to_port = number
        protocol = string
        cidr_blocks = list(string)
    })
    default = {
     from_port = 0,
     to_port = 65365,
     protocol = "tcp",
     cidr_blocks = ["100.0.0.0/16", "200.0.0.0/16", "0.0.0.0/0"]
    }
}
```

Meta-Arguments

Loops

count, for_each, [for(o in var.list: o.id])

```
# creating multiple EC2 instances using count
resource "aws_instance" "server" {
  ami = "ami-06ec8443c2a35b0ba"
  instance_type = "t2.micro"
  count = 3  # creating 3 resources
}

# declaring a variable
variable "users" {
  type = list(string)
  default = ["demo-user", "admin1", "john"]
}

# creating IAM users using for_each
resource "aws_iam_user" "test" {
  for_each = toset(var.users) # converts a list to a set
  name = each.key
}

# A for expression creates a complex type value by transforming another complex ty
variable "names" {
    type = list
    default = ["som", "john", "mary"]
}

output "show_names" {
    # similar to Python's list comprehension
    value = [for n in var.names : upper(n)]
}

output "short_upper_names" {
  # filter the resulting list by specifying a condition:
  value = [for name in var.names : upper(name) if length(name) > 7]
}
```

Splat

```
Splat(var.list[*].id)
```

```
# Launch an EC2 instance
resource "aws_instance" "server" {
  ami = "ami-05cafdf7c9f772ad2"
```

```
  instance_type = "t2.micro"
  count = 3
}

output "private_addresses"{
  value = aws_instance.server[*].private_ip  # splat expression
}
```

depends_on

if two resources depends on each other, depends_on specifies that dependency to enforce ordering

```
resource "aws_iam_role_policy" "example" {
    name = "example"
    role = "s3 access"
    policy = jsonencode({
      "Statement" = [{
        "Action" = "s3:*",
        "Effect" = "Allow",
      }],
    })
}

resource "aws_instance" "my_server" {
    ami  = "ami-a255235"
    instance_type = "t2.micro"

    iam_instance_profile = aws_iam_instance_profile.my_server

    depends_on = [
      aws_iam_role_policy.example,
    ]
```

lifecycle

A set of meta arguments to control behavior specific resources

```
resources "aws_instance" "server" {
  ami           = "ami-a1b3414"
  instance_type = "t2.micro"

  lifecycle {
    create_before_destroy = true
    ignore_changes = [
      # Some resources have metadata
```

```
      # modified automatically outside of TF

      tags
    ]
  }
}
```

Conditionals

```
condition ? true_val : false_val
```

Built-In functions

`max(5, 10, 9)`	12
`min(5, 10, 9)`	5
`format("There are %d servers", 4)`	There are 4 lights
`join("," ["foo", "bar", "baz"])`	foo,bar,baz
`split(",", "foo,bar,baz")`	`["foo", "bar", "baz"]`
`replace("hi world", "/w.*d/", "mom"`	hi mom
`substr("hello world", 1, 4)`	ello
`lookup({a="lol", b="sad"}, "a", "what?")`	lol
`lookup({a="lol", b="sad"}, "c", "what?")`	what?
`slice(["a", "b", "c", "d"], 1, 3)`	`["b", "c"]`
`timestamp()`	"2022-04-02T05:52:48Z"
`cidr("10.1.2.240/28", 1)`	10.1.2.241
`cidr("10.1.2.240/28", 14)`	10.1.2.254

Provider block

Details of the provider(s) being used. Includes information like access mechanisms, regional options, profile touse etc.

```
provider "aws" {
  region = "us-east-1"
}
```

```
# Additional provider config reference as `aws.west`.
provider "aws" {
  alias  = "west"
  region = "us-west-2"
}
```

Requiring Providers

```
terraform {

  # required_providers block specifies source and version
  required_providers {
    aws = {
      source  = "hashicorp/aws" # where to find the provider
      version = "5.8.0" # which version of the provider is needed for this config
    }
  }
}
```

Locals block

A local value assigns a name to an expression, so you can use the name multiple times within a module instead of repeating the expression

```
locals {
  service_name = "forum"
  owner        = "Community Team"
}

# once local declared you can reference it
locals {
  common_tags = {
    Service = local.service_name
    Owner   = local.owner
```

Output block

```
output "api_base_url" {
  value = "https://${aws_instance.example.private_dns}:8433/"

  # The EC2 instance must have an encrypted root volume.
  precondition {
    condition     = data.aws_ebs_volume.example.encrypted
    error_message = "The server's root volume is not encrypted."
  }
```

```
  # output can be marked as containing sensitive
  sensitive = true # output can be marked as containing sensitive
}
```

Data block

Data sources that can be queried (cloud provider, local list, etc.)

```
data "aws_ami" "example" {
  most_recent = true

  owners = ["self"]
  tags = {
    Name   = "app-server"
    Tested = "true"
  }
}
```

Modules

```
module "myec2" {
  source = "../modules/ec2"

  # module inputs
  ami_id = var.ami_id
  instance_type = var.instance_type
  servers = var.servers
}
```

Backends

A backend defines where Terraform stores its state data files.

Available backends: local(default), remote, azurerm, consul, cos, gcs, http, Kubernetes, oss, pg, s3

```
terraform {
  backend "remote" {
    organization = "example_corp"

    workspaces {
      name = "my-app-prod"
    }
  }
}
```

Resource Addressing

`[module path][resource info`	Resource path syntax
`module.<MODULE_PATH>[optional] module index`	Module path syntax
`resource_type.user_defined_name[optional index]`	Resource spec syntax
`<RESOURCE_TYPE>.<NAME>`	List all images
`var.<NAME>`	Input Variable
`local.<NAME>`	Locals
`module.<MODULE_NAME>`	Child module outputs
`data.<DATA TYPE>.<NAME>`	Data blocks
`path.module`	Location of expresssion
`path.root`	Root Module location
`terraform.workspace`	Current workspace

Terraform CLI

Initialization

```
terraform init [options]

    -upgrade            # Install latest module & provider versions
    -reconfigure        # reconfigure backend, ignoring any saved config
    -backend=false      # Disable backend & use previous Initialization
    -migrate-state      # reconfigure backend & attempt to migrate any existing st
```

Planning

Generates & review an execution plan

```
terraform plan [options]

    -var 'user=john'    # set value for input vars in the root module of config
```

```
    -var-file=filename  # load var values from the given file
    -input=true         # ask for input for vars if not directly set
    -out=path           # write a plan file to given path. can be used as input fc
    -refresh-only       # verifies remote object consistency without proposing act
    -destroy            # create plan to destroy all objects currenly managed
    -target=resource    # target planning to given resouce & its dependencies only

terraform plan -refresh-only # updates state to match changes made outside of TF.
```

Validation

```
terraform validate      # Validates the config files for errors
```

Apply

Executes changes to infra

```
terraform apply [options]

    -auto-approve       # skip interactive approval of plan before applying
    -replace            # force replacement of a particular resource instance
    -var 'foo=bar'      # set a value for input vars in the root module of config
    -var-file=filename  # load var values from the given file
    -parallelism=n      # limit the no of concurrent operations. Default=10

terraform apply -auto-approve var-file=web-dev.tfvars
terraform apply -replace="aws_instance.server"
terraform plan -refresh-only # Updates statefile to accept changes made manually.
```

Destroy

Destroy (deletes) Terraform managed infra. Same as `terraform apply -destroy`

```
terraform destroy [options]

    -auto-approve        # skip interactive approval before destroying
    -target              # limits destroy to only given resource & its dependencie


terraform destroy -target aws_vpc.my_vpc -auto-approve
```

Miscellaneous

```
terraform state show aws_instance.my_vm
terraform state pull > my_terraform.tfstate
terraform state mv aws_iam_role.my_ssm_role
terraform state list
terraform state rm aws_instance.my_server

terraform import aws_instance.new_server i-243abc
sudo apt install graphviz
terraform graph | dot -Tpng > graph.png
```

Logging

log levels = TRACE > DEBUG > INFO > WARN > ERROR

```
export TF_LOG_CORE=TRACE     # enable core logging
export TF_LOG_PROVIDER=TRACE # enable provider logging
export TF_LOG_PATH=logs.txt  # to persist logs
```

Also See

Docs
Good FCC Article

Related Cheatsheet

Docker Cheatsheet
Quick Reference

Recent Cheatsheet

Gnome Desktop Cheatsheet
Quick Reference

Arduino Programming Cheatsheet
Quick Reference

HTML Canvas Cheatsheet
Quick Reference

TypeScript Cheatsheet
Quick Reference