Using jq or alternative command line tools to diff JSON files

Are there any command line utilities that can be used to find if two JSON files are identical with invariance to within-dictionary-key and within-list-element ordering?

Could this be done with `jq` or some other equivalent tool?

Examples:

These two JSON files are identical

```
A:
{
  "People": ["John", "Bryan"],
  "City": "Boston",
  "State": "MA"
}

B:
{
  "People": ["Bryan", "John"],
  "State": "MA",
  "City": "Boston"
}
```

but these two JSON files are different:

```
A:
{
  "People": ["John", "Bryan", "Carla"],
  "City": "Boston",
  "State": "MA"
}

C:
{
  "People": ["Bryan", "John"],
  "State": "MA",
  "City": "Boston"
}
```

That would be:

```
$ some_diff_command A.json B.json

$ some_diff_command A.json C.json
The files are not structurally identical
```

json diff jq

edited Dec 28 '16 at 10:47
peak
8,940 4 20 33

asked Aug 10 '15 at 22:11
Amelio Vazquez-Reina
18.2k 39 176 355

5 Answers

Since jq's comparison already compares objects without taking into account key ordering, all that's left is to sort all lists inside the object before comparing them. Assuming your two files are named `a.json` and `b.json`, on the latest jq nightly:

```
jq --argfile a a.json --argfile b b.json -n '($a | (.. | arrays) |= sort) as $a | ($b |
(.. | arrays) |= sort) as $b | $a == $b'
```

This program should return "true" or "false" depending on whether or not the objects are equal using the definition of equality you ask for.

EDIT: The `(.. | arrays) |= sort` construct doesn't actually work as expected on some edge cases. This GitHub issue explains why and provides some alternatives, such as:

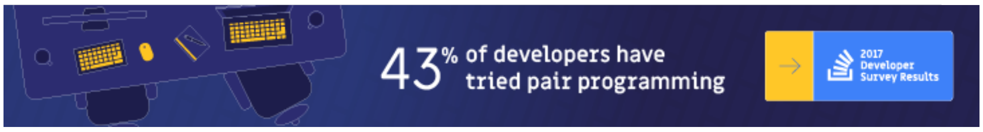
```
def post_recurse(f): def r: (f | select(. != null) | r), .; r; def post_recurse:
post_recurse(.[]?); (post_recurse | arrays) |= sort
```

Applied to the jq invocation above:

```
jq --argfile a a.json --argfile b b.json -n 'def post_recurse(f): def r: (f | select(. !=
null) | r), .; r; def post_recurse: post_recurse(.[]?); ($a | (post_recurse | arrays) |=
sort) as $a | ($b | (post_recurse | arrays) |= sort) as $b | $a == $b'
```

edited Jan 19 at 16:39
UrsinusTheStrong
837 1 9 29

answered Aug 11 '15 at 4:28
Santiago Lapresta
2,824 5 19


43% of developers have tried pair programming
2017 Developer Survey Results

In principle, if you have access to bash or some other advanced shell, you could do something like

```
cmp <(jq -cS . A.json) <(jq -cS . B.json)
```

using subprocesses. This will format the json with sorted keys, and consistent representation of floating points. Those are the only two reasons I can think of for why json with the same content would be printed differently. Therefore doing a simple string comparison afterwards will results in a proper test. It's probably also worth noting that if you can't use bash you can get the same results with temporary files, it's just not as clean.

This doesn't quite answer your question, because in the way you stated the question you wanted `["John", "Bryan"]` and `["Bryan", "John"]` to compare identically. Since json doesn't have the concept of a set, only a list, those should be considered distinct. Order is important for lists. You would have to write some custom comparison if you wanted them to compare equally, and to that would need to define what you mean by equality. Does order matter for all lists or only some? What about duplicate elements? Alternatively if you want them to be represented as a set, and the elements are strings, you could put them in objects like `{"John": null, "Bryan": null}`. Order will not matter when comparing those for equality.

Update

From the comment discussion: If you want to get a better idea of why the the json isn't the same, then

```
diff <(jq -S . A.json) <(jq -S . B.json)
```

will produce more interpretable output. `vimdiff` might be preferable to diff depending on tastes.

edited Jan 26 at 22:10

answered May 12 '16 at 0:52
Erik
1,219 1 16 25

Note that this seems to require version 1.5 or later of `jq` – Adam Baxter Aug 18 '16 at 5:31

1 @voltagex From looking at the online manual (stedolan.github.io/jq/manual/v1.4/#Invokingjq) It seems that it was actually added in 1.4, although I don't know if `jq` does posix style arguments so you may have to invoke `jq -c -S ...` – Erik Aug 18 '16 at 14:46

1 A cleaner, visual form IMO is `vimdiff <(jq -S . a.json) <(jq -S . b.json)` – Ashwin Jayaprakash Dec 8 '16 at 1:25

Yeah, you should remove the `-c` (which makes output compact), style preferences isn't relevant to your answer. – odinho - Velmont Jan 26 at 9:06

@odinho-Velmont @Ashwin Jayaprakash It's true that the `c` isn't strictly necessary, but to me there's no reason for cmp to compare identical whitespace, and no reason for jq to bother emitting it. `diff`, `vimdiff`, or any tool that does file comparison will work, but `cmp` is all that's necessary. – Erik Jan 26 at 17:45

|

Here is a solution using the generic function *walk/1*:

```
# Apply f to composite entities recursively, and to atoms
def walk(f):
  . as $in
  | if type == "object" then
      reduce keys[] as $key
        ( {}; . + { ($key):  ($in[$key] | walk(f)) } ) | f
  elif type == "array" then map( walk(f) ) | f
  else f
  end;

def normalize: walk(if type == "array" then sort else . end);

# Test whether the input and argument are equivalent
# in the sense that ordering within lists is immaterial:
def equiv(x): normalize == (x | normalize);
```

Example:

```
{"a":[1,2,[3,4]]} | equiv( {"a": [[4,3], 2,1]} )
```

produces:

```
true
```

And wrapped up as a bash script:

```
#!/bin/bash

JQ=/usr/local/bin/jq
BN=$(basename $0)

function help {
  cat <<EOF

Syntax: $0 file1 file2

The two files are assumed each to contain one JSON entity.  This
script reports whether the two entities are equivalent in the sense
that their normalized values are equal, where normalization of all
component arrays is achieved by recursively sorting them, innermost first.

This script assumes that the jq of interest is $JQ if it exists and
otherwise that it is on the PATH.

EOF
  exit
}

if [ ! -x "$JQ" ] ; then JQ=jq ; fi

function die     { echo "$BN: $@" >&2 ; exit 1 ; }

if [ $# != 2 -o "$1" = -h  -o "$1" = --help ] ; then help ; exit ; fi

test -f "$1" || die "unable to find $1"
test -f "$2" || die "unable to find $2"

$JQ -r -n --argfile A "$1" --argfile B "$2" -f <(cat<<"EOF"
# Apply f to composite entities recursively, and to atoms
def walk(f):
  . as $in
  | if type == "object" then
      reduce keys[] as $key
        ( {}; . + { ($key):  ($in[$key] | walk(f)) } ) | f
  elif type == "array" then map( walk(f) ) | f
  else f
  end;

def normalize: walk(if type == "array" then sort else . end);

# Test whether the input and argument are equivalent
# in the sense that ordering within lists is immaterial:
def equiv(x): normalize == (x | normalize);

if $A | equiv($B) then empty else "\($A) is not equivalent to \($B)" end

EOF
)
```

POSTSCRIPT: walk/1 is a built-in in versions of jq > 1.5, and can therefore be omitted if your jq includes it, but there is no harm in including it redundantly in a jq script.

edited Nov 6 '15 at 15:42   answered Aug 11 '15 at 7:09
peak
8,940 4 20 33

Use `jd` with the `-set` option:

No output means no difference.

```
$ jd -set A.json B.json
```

Differences are shown as an @ path and + or -.

```
$ jd -set A.json C.json

@ ["People",{}]
+ "Carla"
```

The output diffs can also be used as patch files with the `-p` option.

```
$ jd -set -o patch A.json C.json; jd -set -p patch B.json

{"City":"Boston","People":["John","Carla","Bryan"],"State":"MA"}
```

https://github.com/josephburnett/jd#command-line-usage

edited Dec 6 '16 at 3:53

answered Dec 5 '16 at 21:18
Joe Burnett
31 2

If you also want to see the differences, using @Erik's answer as inspiration and js-beautify:

```
$ echo '[{"name": "John", "age": 56}, {"name": "Mary", "age": 67}]' > file1.json
$ echo '[{"age": 56, "name": "John"}, {"name": "Mary", "age": 61}]' > file2.json

$ diff -u --color \
        <(jq -cS . file1.json | js-beautify -f -) \
        <(jq -cS . file2.json | js-beautify -f -)
--- /dev/fd/63  2016-10-18 13:03:59.397451598 +0200
+++ /dev/fd/62  2016-10-18 13:03:59.397451598 +0200
@@ -2,6 +2,6 @@
     "age": 56,
     "name": "John Smith"
 }, {
-    "age": 67,
+    "age": 61,
     "name": "Mary Stuart"
 }]
```

edited Oct 18 '16 at 11:04

answered Oct 18 '16 at 10:56
tokland
44k 7 93 120

1 ... or y'know just remove the `-c` from the `jq` command line. I dunno, prefer not introducing extra unnecessary tools ;) – odinho - Velmont Jan 26 at 9:05