Bash for Loop Over JSON Array Using jq (/blog/bash-for-loop-over-json-array-using-jq/)

Sometimes you just want to read a JSON config file from Bash and iterate over an array. For example, when seeding some credentials to a credential store (https://github.com/cloudfoundry-community/credhub-seeder-boshrelease/blob/797228c3bf13c0c4f5aefaafa9e38f393bd78b1f/jobs/credhub-seeder/templates/bin/credhub-seeder). This sometimes can be tricky especially when the JSON contains multi-line strings (for example certificates).

In this blog post I will explain how this can be done with jq (https://stedolan.github.io/jq/download/) and a Bash for loop.

First we will start with some data:

```
sample='[{"name":"foo"},{"name":"bar"}]'
echo "${sample}" | jq '.'
```

```
[
  {
    "name": "foo"
  },
  {
    "name": "bar"
  }
]
```

By using `jq --compact-output` (or `-c`) we can get each object on a newline.

```
sample='[{"name":"foo"},{"name":"bar"}]'
echo "${sample}" | jq -c '.[]'
```

```
{"name":"foo"}
{"name":"bar"}
```

We could start iterating of the above with a Bash for loop if our data does not contain spaces or newlines. But since certificates contain newlines we better base64 encode each line. Also, instead of `-c`, we now use `-r` to get rid of the extra quotes.

```
sample='[{"name":"foo"},{"name":"bar"}]'
echo "${sample}" | jq -r '.[] | @base64'
```

```
eyJuYW1lIjoiZm9vIn0=
eyJuYW1lIjoiYmFyIn0=
```

Now let's build our for loop.

```
sample='[{"name":"foo"},{"name":"bar"}]'
for row in $(echo "${sample}" | jq -r '.[] | @base64'); do
    _jq() {
     echo ${row} | base64 --decode | jq -r ${1}
    }

   echo $(_jq '.name')
done
```

```
foo
bar
```