Bash Regex Cheat Sheet

Edit Cheat Sheet

See also Bash

Regexp Matching

Use conditions with doubled [] and the =~ operator. Ensure not to quote the regular expression. Only BRE are allowed. If the regexp has whitespaces put it in a variable first.

```
if [[ $string =~ ^[0-9]+$ ]]; then
    echo "Is a number"
fi
```

Regexp Match Extraction

Variant #1: You can do this with grouping in bash. Despite only BRE being supported grouping works also. Note how you need to set the regexp into a variable because you must not quote it in the if condition!

```
REGEXP="2013:06:23 ([0-9]+):([0-9]+)"
if [[ $string =~ $REGEXP ]]; then
    echo "Hour ${BASH_REMATCH[1]} Minute ${BASH_REMATCH[2]}"
fi
```

Variant #2: Actually using "expr" can much simpler especially when only on value is to be extracted:

```
hour=$(expr match "$string" '2013:06:23 \([0-9]\+\)')
```

Validate IPs

If you need to validate an IP try the following function

```
function validate_ip {
        local net=$1
        [[ $net =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}/[0-9]{1,2}$ ]] || return 1
        [[ ${net#*/} -le 32 ]] || return 1
        local ip=${net%/*}

        local -a oc=(${ip//\./ })
```

CHEAT SHEETS

Agile
Containers
DevOps Automation
DevOps Linux
DevOps Services
Languages
Network
Scripting
Security
Solutions
Virtualization
Windows
Glib Examples
Go Examples
Javascript Examples
PHP Examples

APIs
Bash Associative Array
Bash Functions
Bash Regex
Bash *
Color Distance
Shell Problems
Shell-Scripting
awk
packages.json
sed

```
        [[ ${oc[0]} -le 255 && ${oc[1]} -le 255 && ${oc[2]} -le 255 && ${oc[3]} -le 255 ]] || return 1
        return 0
}
```