

# CONFIGSHELL

**Christian ENGEL (2023-02 V1.1.0)**

RELEASE 1.1

# NEW COMMANDS

- `urlencode <<str>>`, `urldecode <<str>>`  
Conversion of URL-encoded parameters
- `base64-to-base64url`  
Pipe-tool to convert base64 format

# INTRODUCTION

# CONFIGSHELL

ConfiShell is an open-source (MIT-based) project to support Linux/UNIX® interactive working environments. This work is not paid by an employer. It offers:

- Shell interactive support for bash, fish, zsh (partly)
- Linux support for proxied environments
- goLang development support
- TLS & SSH key and certificate creation and analysis
- bash scripting, markdown & LaTeX support
- git encryption and general git support
- k8s support

# FEATURES

- can be installed (git clone) without password
- script included to automatically upgrade when logging in
- continuously be updated
- experience from 40 years of UNIX experience

# THE INSTALLATION

# 1ST STEPS - PUT YOUR SHOES ON...



1. Use the default path if possible

```
mkdir /opt/ConfigShell
```

1. Install ConfigShell

```
git clone https://github.com/engelch/ConfigShell /opt/ConfigShell
```

2. Activate it for you (home directory of the current user)

```
/opt/ConfigShell/installDotFiles2home
```

3. Restart your shell



# GIT-SUBMODULES CONFIG LINUX CONFIG DARWIN

Additions for Linux, OS-X alias Darwin, e.g.

- Linux gnome-terminal colours; signfile, gosha256
- OSX iTerm and Terminal colour profiles; divvy profile; signfile, gosha256

```
cd /opt/ConfigShell  
git submodule update -init
```

# YOU ARE USING CONFIGSHELL, IF

with bash, your prompt looks like # EXTRA by ConfigShell

```
[0] 14:13:46|engelch@mac160|D? feature/vcs...bs/feature/vcs|pki-zz|~/x/cias-pki-svs-k8s
```

exit-code prompt-creation user@host git-repo aws-profile CWD

Here, a possible fish prompt # EXTRA by ConfigShell

```
engelch@mac160 ~/x/cias-pki-svs-k8s (D? feature/vcs...bs/feature/vcs) <AWS:pki-zz> >
```

user@host	PWD/CWD	if in a git-repo	if AWS_PROFILE
-----------	---------	------------------	----------------

# SHELL ENHANCEMENT

# ABOUT

- abbreviations for often used programmes
- let's live the AKÜFI

# ABOUT

- abbreviations for often used programmes
- let's live the AKÜFI (Abkürzfimmel /abbreviation madness)
- standard on all systems (@work, @home, @IoT, @srv)

# BASH & FISH COMMANDS 1 - DIRECTORIES # EXTRA by ConfigShell

ls	multiple columns, colourised, showing file type
la	like ls above with -a option (showing dot files)
ll	like ls above with -l option (long listing, one file per line)
lla	like ls above with -la options
lld	like ls above with -d (show current and parent directory file)
llad	like ls above with -adl
cd..	same as cd .. (typoo saver)
.2 ... .7	cd ../.. ... cd ../../../../../../../..
mkcd <dir>	create directory and make it the CWD (current working directory)
brmd	go to the parent directory and try to delete the child directory (must be mt)

# BASH & FISH COMMANDS 1 - DIRECTORIES

# EXTRA by ConfigShell

ls	multiple columns, colourised, showing file type
la	like ls above with -a option (showing dot files)
ll	like ls above with -l option (long listing, one file per line)
lla	like ls above with -la options
lld	like ls above with -d (show current and parent directories)
llad	like ls above with -adl
cd..	same as cd .. (typoo saver)
.2 ... .7	cd ../.. ... cd ../../../../..
mkcd <dir>	create directory and make it the CWD (current working directory)
brmd	go to the parent directory and try to delete the child directory (must be mt)

**cd -**

change to previous CWD

# BASH & FISH COMMANDS 2 # EXTRA by ConfigShell

cp / rm / mv	all mapped to include the -i option (confirm deletion, protect overwrite)
l	less ( $\Delta$ to some Ubuntu default installations)
j	jobs
wh	which
rm~ rmtex	delete backup files or TeX intermediate files
a	alias (show aliases - depending on the shell there might be more)
pu	pushd .
po	popd
8601	output date/time in UTC and in ISO 8601 format w/out spaces



# BASH & FISH COMMANDS 3 # EXTRA by ConfigShell

tm	tmux new -s
tw ⇔ tn	tmux new-window -n
tj	tmux join-pane -s
tmux-prd, tmux-prd2	red-coloured tmux, tmux select-pane -P "fg=white,bg=color052
tmux-qu1	yellow-coloured tmux
tmux-dv1	blue-coloured tmux
tmux-loc	white-grey coloured tmux
tmux-blwh	black-whitish coloured tmux

# BASH LOADING FILES # EXTRA by ConfigShell

bash uses `~/.bashrc.d`

`*.rc` files are sourced into the current shell

`*.sh` files are executed (sub-shell)

When?

1. Done when starting the shell
2. Re-Done by executing `rl` or `rlFull` (bash)

# BASH LOADING FILES - EXAMPLE # EXTRA by ConfigShell

bash uses `~/.bashrc.d`

`*.rc` files are sou

`*.sh` files are exe



`~/.bashrc.d/aws.rc`

`export AWS_PROFILE=pki-zz`

When?

1. Done when starting the shell
2. Re-Done by executing `rl` or `rlFull` (bash)

# FISH LOADING FILES

fish uses the default directory `~/.config/fish`

Most files here are s-linked to `/opt/ConfigShell`, but:

fish sources 2 files at start/end of loading if existing:

`~/.config/fish/pre.fish` # EXTRA by ConfigShell

`~/.config/fish/conf.d/*.fish`

`~/.config/fish/post.fish` # EXTRA by ConfigShell

EXTRA: `~/.config.fish/conf.d/*.sh` are executed with bash

# FISH VS BASH VS ZSH

[comparison article on medium.com](#)

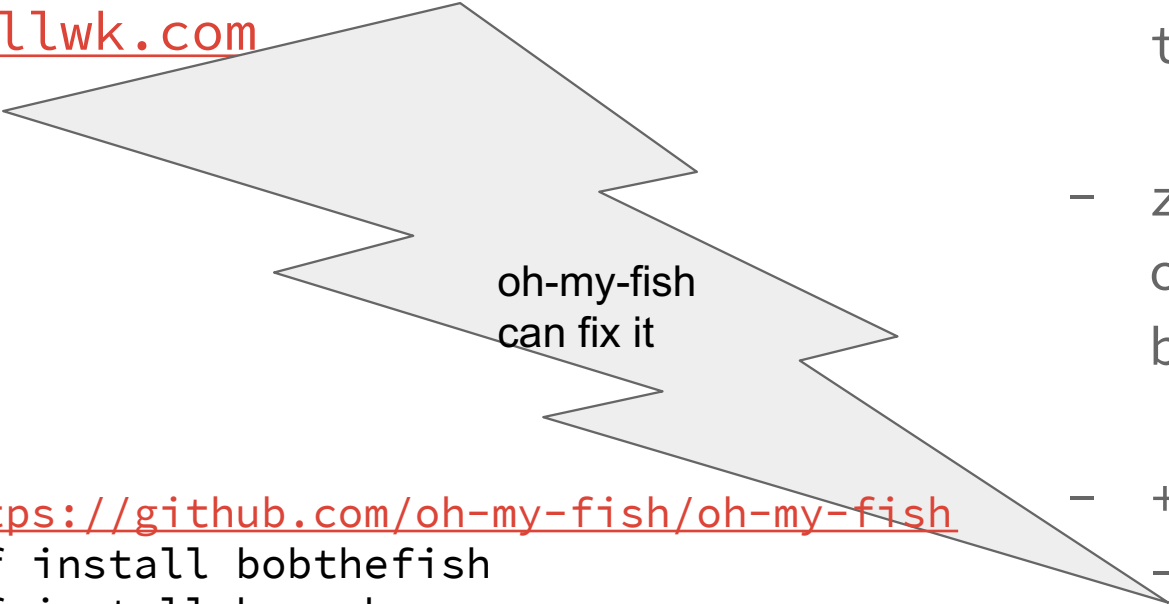
[zellwk.com](#)

- completion in bash can be a pain and was better before (good old tcsh times,...)
- zsh scripts and add-ons (oh-my-zsh) can become complex
- +good defaults: fish  
-missing in fish: !! !\$

# FISH VS BASH VS ZSH

[comparison article on medium.com](#)

[zellwk.com](#)



oh-my-fish  
can fix it

<https://github.com/oh-my-fish/oh-my-fish>

omf install bobthefish

omf install bang-bang

- completion in bash can be a pain and was better before (good old tcsh times,...)
- zsh scripts and add-ons (oh-my-zsh) can become complex
- +good defaults: fish  
-missing in fish: !! !\$

# OH-MY-FISH INSTALLATION

## Installation

```
curl https://raw.githubusercontent.com/oh-my-fish/oh-my-fish/master/bin/install | fish
```

## Overview about themes

<https://github.com/oh-my-fish/oh-my-fish/blob/master/docs/Themes.md>

## Recommended Themes

```
omf install bobthefish
```

```
omf install bang-bang          # !! and !$ now working
```

# OH-MY-FISH INSTALLATION 2

Terminal requires Powerline font

1. Install powerline

```
sudo apt install powerline && source /usr/share/powerline/bindings/bash/powerline.sh
```

2. Download and install fonts

```
git clone https://github.com/powerline/fonts.git && cd fonts && sh ./install.sh
```

3. Add this lines to .bashrc file if bash is what you use. You could add to other files like .zshrc etc. I use bash so there. This ensures that this switch is automatic when you start bash

```
if [ -f /usr/share/powerline/bindings/bash/powerline.sh ]; then
    source /usr/share/powerline/bindings/bash/powerline.sh
fi
```

Ubuntu 22.04 - as CE did it

1. sudo apt install -y fonts-powerline
2. ln -s usr/share/fonts/opentype/PowerlineSymbols.otf ~/.fonts
3. gnome-terminal to use Ubuntu Mono Font  
[ ConfigLinux gnome-terminal settings use Ubuntu Mono]

OR cd /opt/ConfigShell/PowerlineFonts; ./install.sh



# ENCRYPTION & TLS & SSH

# BASH & FISH COMMANDS 4 # EXTRA by ConfigShell

encryptFile [-d] [-k] [-f] [file]...	encrypt/decrypt file
	-k :: keep original file
	-f :: force, overwrite existing files
signFile	part of ConfigLinux and ConfigDarwin
	also for verification of signatures
gosha256	fast sha256 digest
hex2	convert into hex or base64 or raw

# TLS KEY MATERIAL & CSR

Create key-pair

```
tls-csr-key-creator.sh -k <<baseFilename, here:test13>>
```

⇒ creates 2 files test13.key (prv key) and test13.pub (pub key)

Create key-pair & CSR

```
tls-csr-key-creator.sh -g -c cn-value -o ou-value cert1
```

Create CSR for existing key-pair

```
tls-csr-key-creator.sh -c cn-value -o ou-value cert1
```

# TLS KEY MATERIAL & CSR

tls-csr-key-creator.sh supports

- RSA 2K/4k keys, ECC keys
- CSR with CN, O, OU, S, SAN fields
- `-h` for help

Create key-pair

```
tls-csr-key-creator.sh -g <<baseFilename, here:test13>>
```

⇒ creates 2 files test13.key (prv key) and test13.pub (pub key)

Create key-pair & CSR

```
tls-csr-key-creator.sh -g -c cn-value -o ou-value cert1
```

Create CSR for existing key-pair

```
tls-csr-key-creator.sh -c cn-value -o ou-value cert1
```

# CSR SHOW

`tls[-cert.sh] cert1.csr ...` OR `tls-csr.sh cert1.sh`

cert1.csr:Certificate Request:

cert1.csr: Data:

cert1.csr: Version: 1 (0x0)

cert1.csr: Subject: 0 = demoou, CN = democn

cert1.csr: Subject Public Key Info:

cert1.csr: Public Key Algorithm: rsaEncryption

cert1.csr: RSA Public-Key: (4096 bit)

cert1.csr: Modulus:

cert1.csr: Exponent: 65537 (0x10001)

cert1.csr: Attributes:

cert1.csr: a0:00

cert1.csr: Requested Extensions:

cert1.csr: Signature Algorithm: sha256WithRSAEncryption

# TLS TOOLS - FILE COMPARISON - DO YOU BELONG TOGETHER

check / compare keys of different files

```
tls-cert.sh -f cert1.*
```

```
2a5076acd8f0efb279de0e9d116417507e8bf08702bb627bf5756f009048faba cert1.crt  
2a5076acd8f0efb279de0e9d116417507e8bf08702bb627bf5756f009048faba cert1.csr  
2a5076acd8f0efb279de0e9d116417507e8bf08702bb627bf5756f009048faba cert1.key  
2a5076acd8f0efb279de0e9d116417507e8bf08702bb627bf5756f009048faba cert1.pub
```

# CONVERT P7B TO PEM

```
tls-p7b-to-pem.sh [ file ]
```

- can also be used in a pipe

# GET PUBLIC KEY FROM PRIVATE KEY

```
tls-rsa-prv-2-pub-key.sh [ file ]
```

- can also be used in a pipe



# CA - CREATION FOR SIMPLE TESTS

Simple CA creation does not replace a fully fledged PKI!

```
tls-ca-create-key-cert.sh <<options>> <<base-filename>>
```

```
-c cn-value -o o-value -u ou-value -s c-value
```

```
-y expiryDays4caCert
```

```
-g # generate-key-material
```

# SIGN CSRS

For simple CA cases

```
tls-sign-csr.sh [-D] [-d days] [-s] -c ca-basefilename [<<csr-file>> ...]
```

```
-s ::= server-certificate
```

# OLD

## Commands

```
tls-rsa-prv-fingerprint.sh [file]
```

```
tls-rsa-pub-fingerprint.sh [file]
```

can be replaced by

```
tls[-cert] -f [file]
```

# FINAL SIMPLIFICATION

Command

`tls-cert.sh`

can be called as `tlsCert` or just as

`tls`

SSH

# SSH KEY AND CERTIFICATE SUPPORT

`ssh-certificate`

SSH cert information

`ssh-fingerprint`

accepts pub, prv keys, and certs

`ssh-prv-2-pub-key`

extracts the public key

# SSH EXTENDED COMMANDS

`ssh-createCompletionList`

creates/overwrites ssh completion list for hostnames from \*.config files under ~/.ssh/ as file ~/.ssh/completion.lst  
(ok for bash & fish: completion with → char)

`ssh-grep`

`ssf`

Searches ssh .config files for a hostname pattern (case-insensitive)

ssf is an alias for ssh-grep

# GIT & GEE ENHANCEMENT



# GIT INITIALISATION

`gitInit`

initialise git with reasonable default settings.  
This commands changes `~/.gitconfig`

# GIT STATUS

`gitStatus`

helper command for shell prompts to show the status of the current git repository.  
(current: from the CWD-perspective)

`git-status-all`

Show status of git directories starting from your home directory

# GIT HELPERS

`git-delete-last-remote-commit`

Delete the last remote commit, keeping local commits untouched. The executed command will be shown first before the execution starts.

`git-delete-remote-branch`

Deletes the specified remote branch. Also shows the command first and asks for execution.

# GIT GEE - INTRODUCTION / ABOUT

git gee

extension for git to

- store confidential or secret data in a git repository
- uses a shared, symmetric encryption key model (key to be shared by a password mgmt solution)
- other solutions (git crypt, ...) did not work as expected
- asymmetric keys solutions face the problem to share commits that were created before a person was added to the team
- fitting our demands, not trying to fit for all requirements
- prevents committing confidential files (bullet proofed)

# GIT GEE - START

`git gee init`

1. You have an existing git repository, let's call it gee-test
2. Create and save a secret key (use a key-creator tool)  
Skeletal-Absently-Linguini-Return-Politely2-Legacy-Cape
3. Put this into a file besides the gee-test/ directory called gee-test.gee.pw
4. Can't remember the filename, just call `git gee init`  
ERROR:Password file /Users/engelch/tmp/gee-test.gee.pw not found
5. Also check the permissions of the pw file and the git repo  
`ll -d gee*`  
drwx-----@ 5 engelch staff 160B Feb 2 18:46 gee-test/  
-rw-----@ 1 engelch staff 56B Feb 2 18:50 gee-test.gee.pw
6. call `git gee init` from inside the git repo  
engelch@mac160 ~/t/gee-test (master) > git gee init  
engelch@mac160 ~/t/gee-test (master) >

# GIT GEE - LIST FILES PROTECTED BY GEE

`git gee help`

help, please read it once

`git gee -h`

`git version`

show version

# GIT GEE - LIST FILES PROTECTED BY GEE

```
git gee l
```

1. list files under git gee, now: no files

```
git gee li
```

```
git gee list
```

```
git gee lst
```

```
git gee a a.txt
```

2. put file a.txt under git gee protection

```
engelch@mac160 ~/t/gee-test (? master) > git gee a a.txt
```

```
Processing file a.txt Encryption successful
```

- a. encrypted file a.txt.gee was created
- b. a.txt is in .gitignore
- c. git hook prevents committing if a.txt is newer than a.txt.gee

# GIT GEE - LIST FILES, ENCRYPT FILES, REMOVE UNENCRYPTED

`git gee s`

show status, git status does not say everything. Why?

*git status does not show the status of ignored files. We ignored a.txt, but not a.txt.gee. So, if a.txt is change, the pre-commit would prevent a commit, but git status might say, everything is ok.*

`git gee e (git gee encrypt)`

encrypt files again, which are newer than the .gee counterpart. (File modification times are important.)

`git gee clean`

delete the unencrypted versions of .gee encrypted files



# GIT GEE - UNENCRYPT

`git gee u`

unencrypt files protected by git gee

# GIT GEE WRONG PASSKEY

If the wrong passkey is installed, git gee d might produce output like

```
engelch@air0 ~/x/cias-pki-svs-test (master...origin/master) <AWS:pki-zz> > git gee u
Procsesing file /Users/engelch/x/cias-pki-svs-test/tests/fda/d/pki-d-zz-fda.key.gee ERROR! Decryption failed (no vault secrets were found that could decrypt) on /private/var/fold
ers/ns/shqtfbhx4fz5g2ll6rttpwsc0000gn/T/git-gee.GBHxJkDE for /private/var/folders/ns/shqtfbhx4fz5g2ll6rttpwsc0000gn/T/git-gee.GBHxJkDE
ERROR:encrypting file /var/folders/ns/shqtfbhx4fz5g2ll6rttpwsc0000gn/T/git-gee.GBHxJkDE
engelch@air0 ~/x/cias-pki-svs-test (master...origin/master) <AWS:pki-zz> [66]> cat ../cias-pki-svs-test.gee.pw
```

# GIT ALIASES & FUNCTIONS

gia	git add -A
gibr	git branch -avv
gidi	git diff
gidic	git diff -cached
gife	git fetch -all -p
gilo	git log --branches --remotes --tags --graph --oneline --decorate
gist	git status -u --show-stash --ignore-submodules
gipl	git pull --all; git fetch --tags
girm	git status   sed '1,/not staged/d'   grep deleted   awk '{print \"\$2\"}'   xargs git rm
gicm / gicma	git commit -m ... / git commit -a -m ...
gipu / gipua	git push --all \$argv; and git push --tags \$argv / for all remote repos: gipu

# KUBERNETES (K8S) ENHANCEMENT

# KUBECTL HELPERS

k / k8 / k8s	kubectl
k8af / k8df	kubectl apply -f ... / kubectl delete -f ...
k8c / k8cv	kubectl config / kubectl config view
k8cg / k8cs / k8cu	kubectl config get/set/use-context
k8gd / k8gdA ⇔ k8gda	kubectl get deploy -o wide / k8gd -A
k8gn	kubectl get nodes -o wide
k8gp / k8gpa ⇔ k8gpA	kubectl get pods -o wide / k8gp -A
k8ns	kubectl get services
k8ga / k8gaa ⇔ k8gaA	kubectl get all / k8ga -A
k8ev ⇔ k8events	kubectl get events --sort-by=.metadata.creationTimestamp

# KUBECTL POD HELPERS

These commands work as soon as the specification of the pod uniquely identifies one pod.

<code>k8logs [ -n &lt;&lt;ns&gt;&gt; ] [ -f ] &lt;&lt;unique-pod-id&gt;&gt;</code>	get logs from a cube
<code>k8exec [ -n &lt;&lt;ns&gt;&gt; ] &lt;&lt;unique-pod-id&gt;&gt; [cmd]</code>	exec in pod, def: bash
<code>k8cp [-c &lt;cont&gt;] [-n &lt;ns&gt;] &lt;uniquePodSpec&gt;:file file</code>	copy file from container
<code>k8cp [-c &lt;cont&gt;] [-n &lt;ns&gt;] file &lt;uniquePodSpec&gt;:file</code>	copy file to container

DEVELOPMENT

# SCRIPT DEVELOPMENT

1. Usually, stay with bash as this is the most common shell on all systems these days.
2. Start a script with shebang  
`#!/usr/bin/env bash`
3. Use shellcheck to check your shell
4. Easy start: copy  
`/opt/ConfigShell/Template/bash.mini.skeleton.sh`



# GOLANG DEVELOPMENT

<code>gode</code>	<code>goexec-debug</code> - compile debug version
<code>gore</code>	<code>goexec-release</code> - compile release version
<code>godue</code>	<code>goexec-upx</code> - compile compressed version
	upx is not working on all platforms
<code>godebug [-f]</code>	compile debug version
<code>gorelease [-f]</code>	
<code>go-status</code> ↔ <code>gost</code>	check if a compilation makes sense
<code>godistclean</code>	delete build directories,...

# VERSION INCREASES - SEMANTIC VERSIONING CONCEPT

version.sh [-v]

1. if ./versionFilePattern (filename pattern), then it uses the pattern as a regex for the filename to determine the version.
2. elif if version.txt exists, then get the version# from this file (empty lines are removed)
3. else it greps all \*.go files for `app.?version[[:space:]]*= x.y.z` and uses x.y.z as the version#
4. -v returns the format in the form file-name:version#. Otherwise, just the version# is returned.

bump\* (below) are based on version.sh and bumpversion (not part of ConfigShell)

<code>version.sh</code>	used to determine version of actual sw
<code>bumpmajor</code> ⇔ <code>bma</code>	increase the major version
<code>bumpminor</code> ⇔ <code>bmi</code>	increase the minor version
<code>bumppatch</code> ⇔ <code>bpa</code>	increase the patch version

# CONNECT TO POSTGRESQL, MARIADB, MYSQL

`db-connect.sh [ <<role>> ]`

`db-connect-<<role>>.sh`

being s-link to `db-connect.sh`

Requires `db-connect.pw` or `db-connect.pws` (must be s-link)

Example `db-connect.pw` for role postgres, file can contain many roles

```
postgres_DB_TYPE=psql
postgres_HOST=127.0.0.1
postgres_PORT=5432
postgres_USER=postgres
postgres_PW=secret%9127/Pe
postgres_DB=postgres
```

# TEMPLATES

See the Template directory for

- bash skeletons
- markdown skeletons
- LaTeX skeletons

VIRTUALISATION

# VIRTUALISATION SUPPORT

`container-container-ls.sh`

- list all matching containers (instances of images)

`container-container-rm.sh`

- delete all matching containers

# BUILDING CONTAINER IMAGES

`container-image-build.sh [-n] [-t <<arch>>]...`

- use podman if available, else docker
- use Containerfile, if not existing check for Dockerfile
- use version.sh to determine the version to be build, and build for tag latest
- name of container from `_name_<<name>>` file or from current directory
- build a container for default architecture amd64, if not `-t arm64,...` is specified
- `-n` dry-run

`10_simpleBuild.sh`

- roughly identical to above `container-image-build.sh`

# BUILDING CONTAINER IMAGES 2

`10_simpleAwsBuild.sh [ -n ] ...`

- log-in to AWS to be able to build images based on images in ECR
- options are passed to container-image-build.sh
- -n dry-run

`10_goCompileBuild.sh`

`10_goCompileAwsBuild.sh`

- copy ../\*.go and ../../packages to this directory
- compile application using bootstrap container
- based on container-image-build.sh
- Aws version allows for using AWS ECR images for the build process



# BUILDING CONTAINER IMAGES 3

`20_aws_tag.sh`

- tag container for upload to AWS ECR

`30_aws_push.sh`

- push previously tagged image to AWS ECR

Based on

`container-image-aws-push.sh`

`container-image-aws-tag_push.sh`

# MANAGE CONTAINER IMAGES

`container-image-ls.sh [ pattern ]`

- list container images in inspect form

`container-image-rm.sh [ pattern ]`

- delete matching container images

# EKSCTL AWS EKS CLUSTER VISIBILITY

Change `clustervisibility`

- requirements: creation with eksctl

`eksctlClusterVisibility <<clustername>> (true|false)`

- true: publicly visible

DOCUMENTATION

# LATEX

`\latexMoveSections ( down | up ) [ file ... ]`

- move all sections one level up/down.
- if `\section` is existing, then up will create an error (`\part` not supported)
- if `\subparagraph` is existing, then down will create an error

# MARKDOWN

`mdMoveSections ( down | up ) [ file ... ]`

- still to be implemented

# LOREM3

lorem3 - create senseless text for testing

by Per Erik Strandberg

-h, --help	show this help message and exit
-v, --version	show program's version number & exit
--words N, -n N	number of words
--sentences S, -s S	number of sentences
--lines L, -l L	number of lines
--chars C, -c C	number of chars (excl. final \n)
--lorem, --cicero	
--faust, --goethe ...	

OS-SPECIFIC  
ENHANCEMENT  
—  
SUBMODULES



# OS UPGRADE

pkgUpgrade

- update packages on OS
- support
  - OSX
  - Linux DEB (Debian, Ubuntu,...)
  - Linux DNF (Fedora,...)

# CONFIGSHELL UPGRADE

`upgradeConfigShell.sh`

- to be linked to `~/.bashrc.d/` or to `~/.config/fish/conf.d`
- upgrades (git pull) ConfigShell if the last call to the script was more than 4 hours before
- duration (4 hours; negative numbers) can be changed using the environment-variable

```
${UPDATE_CONFIGSHELL_FREQUENCY:-4}
```

# PROXIES

proxy, sudoProxy

- Start a command with proxy settings
- Start a command with sudo and proxy settings, e.g.  
`sudoProxy apt update`

Commands require environment variables to be set (`~/.bashrc.d/<<xx.rc>>`):

PROXYUSER

PROXYHOST

`https_proxy`, `http_proxy`, and `ftp_proxy` will be set

# ROUTE53

route53 – change AWS hosted DNS records

domains

list domains hosted by

current AWS account

domainlist|listdomain <domain>

list A and CNAME records of a domain

listcname <domain>

list CNAME records of a domain

search <domain> <regExpr>

search the domain for CNAME and A

records

create <FQDN> <IPAddr>|<destFQDN>

add the A|CNAME record to the domain if not

existing

upsert <FQDN> <IPAddr>|<destFQDN>

update/insert the A|CNAME record to the domain if

not existing

delete <FQDN> <IPAddr>|<destFQDN>

delete an A|CNAME record

zoneid <domain>

list hostedzoneID

JOIN THE TEAM

# THE FUTURE OF CONFIGSHELL

## Help us

Configshell is a work of enthusiastic persons, please join

## Propose Ideas

Tell us the opinion about it, please help to improve it

## Code with us

Even better, add functionality, streamline elements



**Christian ENGEL**

<mailto:engel-ch@outlook.com>