```
#!/bin/bash
# shellcheck source=kz_common.sh
##############################################################################
# Distribute kz package.
#
# Written by Karel Zimmer <info@karelzimmer.nl>, CC0 1.0 Universal
# <https://creativecommons.org/publicdomain/zero/1.0>, 2009-2023.
##############################################################################

##############################################################################
# Import
##############################################################################

program_path=$(cd "$(dirname "$(realpath "$0")")" && pwd)
source "$program_path"/kz_common.sh 2> >(systemd-cat) || exit 1


##############################################################################
# Variables
##############################################################################

program_name='kz-deploy'
program_desc=$(gettext 'Distribute kz package')
display_name=${program_name/kz-/kz }

usage=$(eval_gettext "Usage: \$display_name \$options_usage")
help="$(eval_gettext "Usage: \$display_name [OPTION...]")

$program_desc.

$(gettext 'Options:')
$options_help"

deb_repo=$HOME/kz-deb
docs_repo=$HOME/kz-docs
scripts_repo=$HOME/kz-scripts
uploads_repo=$HOME/kz-uploads
uploads_repo_distdir=$uploads_repo/dist


##############################################################################
# Functions
##############################################################################

function check_input {
    local -i rc=0
    local parsed=''

    parsed=$(
        getopt  --alternative                       \
                --options       "$options_short"    \
                --longoptions   "$options_long"     \
                --name          "$display_name"     \
                --              "$@"
        ) || rc=$?
    if [[ $rc -ne $ok ]]; then
        printf '%s\n' "$usage_line"
        exit $error
    fi
    eval set -- "$parsed"
    process_option "$@"

    while true; do
        case $1 in
```

```
            --)
                shift
                break
                ;;
            *)
                shift
                ;;
        esac
    done

    if [[ -n "$*" ]]; then
        printf  "$display_name: $*: %s\n$usage_line\n" \
                "$(gettext 'arguments are not allowed')"
        exit $error
    fi
}


function process_input {
    if groups "$USER" | grep --quiet --regexp='sudo'; then
        sudo true
    else
        printf  '%s\n' "$(gettext 'Already performed by the administrator.')"
        term_script
    fi
    printf "${bold}%s${normal} ($display_name)\n" "$(gettext 'Check')"
    check_repos
    printf  "\n${bold}%s${normal} ($display_name)\n" \
            "$(gettext 'Build - final')"
    build_package
    printf "\n${bold}%s${normal} ($display_name)\n" "$(gettext 'Deploy')"
    upload_website
    printf "\n${bold}%s${normal} ($display_name)\n" "$(gettext 'Install')"
    install_package
}


function check_repos {
    info "$(gettext 'Check that all repos are on branch main...')"
    for repo in  $deb_repo $docs_repo $scripts_repo $uploads_repo; do
        cd "$repo"
        if [[ $(git branch --show-current) != 'main' ]]; then
            error "$(eval_gettext "Repo \$repo not on branch main.")"
            git status
            maxrc=1
        fi
    done
    info "$(gettext 'Check that all repos are clean...')"
    for repo in  $deb_repo $docs_repo $scripts_repo $uploads_repo; do
        cd "$repo"
        if ! git diff-index --quiet HEAD; then
            error "$(eval_gettext "Repo \$repo is not clean.")"
            git status
            maxrc=1
        fi
    done
    if [[ $maxrc -gt $ok ]]; then
        exit $maxrc
    fi
}


function build_package {
    local -i rc=0
```

```
    info "$(gettext 'Build package and website (kz build)...')
"
    # Call kz-build.
    "$scripts_repo"/usr/bin/kz build || rc=$?
    if [[ $rc -ne $ok ]]; then
        exit $rc
    fi
}


function upload_website {
    local ftp_set='set ssl:verify-certificate no'
    local ftp_from=$uploads_repo_distdir
    local ftp_to=/httpdocs
    local ftp_opts='--reverse --delete --verbose'
    local ftp_exclude='--exclude icaclient_20.04.0.21_amd64.deb'
    local ftp_cmd="mirror $ftp_exclude $ftp_opts $ftp_from $ftp_to; exit"
    local ftp_host=server106.hosting2go.nl
    local ftp_user=kzimmer
    local ftp_login=$HOME/.kz-$ftp_host

    info "$(gettext 'Upload website (lftp)...')"
    if ! [[ -f $ftp_login ]]; then
        read -rsp "$(gettext 'Password for') ftp://$ftp_host': " < /dev/tty
        printf '%s' "$REPLY" > "$ftp_login"
        printf '\n'
        chmod 'u=rw,g=,o=' "$ftp_login" |& $logcmd
    fi
    if ! lftp   --user      "$ftp_user"             \
                --password  "$(cat "$ftp_login")"   \
                -e          "$ftp_set; $ftp_cmd"    \
                "$ftp_host"; then
        rm "$ftp_login"
        error "$(gettext 'Website upload failed.')"
        exit $error
    fi
    sleep 5
}


function install_package {
    # Call kz-getdeb via wget as a regular user would do.
    info "$(gettext 'Install package kz (kz getdeb)...')
"
    if ! wget --output-document=- karelzimmer.nl/kz 2> >($logcmd) | bash; then
        error "$(gettext 'Install package kz failed.')"
        exit $error
    fi
}


function term_script {
    exit $ok
}


################################################################################
# Main
################################################################################

function main {
    init_script "$@"
    check_input "$@"
```

```
    process_input
    term_script
}

main "$@"
```