KSConf Documentation

Release 0.5.4

Lowell Alleman

Jan 24, 2019

Contents:

1 Intro **1**

2 Install **3**

2.1 Installation Guide . . . 3
2.2 Command line reference . . . 10
2.3 Developer setup . . . 20
2.4 Contributing back . . . 21
2.5 Changelog . . . 22
2.6 License . . . 26

3 Indices and tables **31**

CHAPTER 1

Intro

This utility handles a number of common Splunk app maintenance tasks in an installable python package. Specifically, this tools deals with many of the nuances with storing Splunk apps in a version control system like git and pointing live Splunk apps to a working tree, merging changes from the live system's (local) folder to the version controlled (default) folder, and dealing with more than one layer of "default" (which splunk can't handle natively).

CHAPTER 2

Install

```
pip install kintyre-splunk-conf
```

2.1 Installation Guide

The following doc describes installation options for Kintyre's Splunk Configuration tool. This tool is available as a normal Python package that should require very minimal effort to install and upgrade. However, sometimes Python packaging gets ugly and the one-liner may not work.

A portion of this document is targeted at those who can't install packages as Admin or are forced to use Splunk's embedded Python. For everyone else, please start with the one-liner!

2.1.1 Quick install

Using pip:

```
pip install kintyre-splunk-conf
```

System-level install: (For Mac/Linux)

```
curl https://bootstrap.pypa.io/get-pip.py | sudo python - kintyre-splunk-conf
```

Note: This will also install/update `pip` and work around some known TLS/SSL issues

Enable Bash completion

If you're on a Mac or Linux, and would like to enable bash completion, run these commands:

```
pip install argcomplete
echo 'eval "$(register-python-argcomplete ksconf)"' >> ~/.bashrc
```

2.1.2 Requirements

- Python Supports Python 2.7, 3.4+
- PIP (strongly recommended)
- Tested on Mac, Linux, and Windows

Check Python version

Check your installed python version by running:

```
python --version
```

Note that Linux distributions and Mac OS X that ship with multiple version of Python may have renamed this to `python2`, `python2.7` or similar.

Check PIP Version

```
pip --version
```

If you are running a different python interpreter version, you can instead run this as:

```
python2.7 -m pip --version
```

2.1.3 Installation

There are several ways to install ksconf. Technically all standard python packaging approaches should work just fine, there's no compiled code or external runt-time dependencies so installation is fairly easy, but for non-python developers there are some gotchas. Installation options are listed from the most easy and recommended to more obscure and difficult:

Install from PyPI with PIP

The preferred installation method is to install via the standard Python package tool 'pip'. Ksconf can be installed via the registered `kintyre-splunk-conf` package using the standard python process.

There are 2 popular variations, depending on whether or not you would like to install for all users or just play around with it locally.

Install ksconf into a virtual environment

Use this option if you don't have admin access

Installing `ksconf` with virtualenv is a great way to test the tool without requiring admin privileges and has many advantages for a production install too. Here are the basic steps to get started.

Please change `venv` to a suitable path for your environment.

```
# Install Python virtualenv package (if not already installed)
pip install virtualenv

# Create and activte new 'venv' virtual environment
virtualenv venv
source venv/bin/activate

pip install kintyre-splunk-conf
```

Windows users: The above virtual environment activation should be run as `venv\Scripts\activate.bat`.

Install ksconf system-wide

Note: This requires admin access.

This is the absolute easiest install method where 'ksconf' is available to all users on the system but it requires root access and `pip` must be installed and up-to-date.

On Mac or Linux, run:

```
sudo pip install kintyre-splunk-conf
```

On Windows, run this commands from an Administrator console.

```
pip install kintyre-splunk-conf
```

CentOS (RedHat derived) distros

```
# Enable the EPEL repo so that `pip` can be installed.
sudo yum install -y epel-release

# Install pip
sudo yum install -y python-pip

# Install ksconf (globally, for all users)
sudo pip install kintyre-splunk-conf
```

Install from GIT

If you'd like to contribute to ksconf, or just build the latest and greatest, then install from the git repository is a good choice. (Technically this is still installing with `pip`, so it's easy to switch between a PyPI install, and a local install.)

```
git clone https://github.com/Kintyre/ksconf.git
cd ksconf
pip install .
```

See developer docs for additional details about contributing to ksconf.

2.1.4 Use the standalone executable

Ksconf can be installed as a standalone executable zip app. This approach still requires a python interpreter to be present either from the OS or the one embedded with Splunk Enterprise. This works well for testing or when all other options fail.

From the GitHub releases page, grab the file name `ksconf-*.pyz`, download it, copy it to a `bin` folder in your PATH and rename it `ksconf`. The default shebang looks for 'python' in the PATH, but this can be adjusted as needed. Since installing with Splunk is a common use case, a second file named `ksconf-*-splunk.pyz` already has the shebang set for the standard `/opt/splunk` install path.

Typical embedded Splunk install example:

```
VER=0.5.0
curl https://github.com/Kintyre/ksconf/releases/download/v${VER}/ksconf-${VER}-splunk.
↪pyz
mv ksconf-${VER}-splunk.pyz /opt/splunk/bin/
cd /opt/splunk/bin
ln -sf ksconf-${VER}-splunk.pyz ksconf
chmod +x ksconf
ksconf --version
```

Reasons why this is a non-ideal install approach:

- Lower performance since all python files live in a zip file, and precompiled version's can be cached.
- No standard install pathway (doesn't use pip); user must manually copy the executable into place.
- Uses a non-standard build process. (May not be a big deal, but could cause things to break in the future.)

Install the Wheel manually (offline mode)

Download the latest "Wheel" file file from PyPI, copy it to the destination server and install with pip.

Offline pip install:

```
pip install ~/Downloads/kintyre-splunk-conf-0.4.2-py2.py3-none-any.whl
```

Install with Splunk's Python

Splunk Enterprise 6.x and later installs an embedded Python 2.7 environment. However, Splunk does not provide packing tools (such as `pip` or the `distutils` standard library which is required to bootstrap install `pip`). For these reasons, it's typically easier and cleaner to install `ksconf` with the system provided Python. However, sometimes the system-provided Python environment is the wrong version, is missing (like on Windows), or security restrictions prevent the installation of additional packages. In such cases, Splunk's embedded Python becomes a beacon of hope.

On Linux or Mac

Download the latest "Wheel" file file from PyPI. The path to this download will be set in the `pkg` variable as shown below.

Setup the shell:

```
export SPLUNK_HOME=/opt/splunk
export pkg=~/Downloads/kintyre_splunk_conf-0.4.9-py2.py3-none-any.whl
```

Run the following:

```
cd $SPLUNK_HOME
mkdir Kintyre
cd Kintyre
```

(continues on next page)

(continued from previous page)

```
# Unzip the 'kconf' folder into SPLUNK_HOME/Kintyre
unzip "$pkg"

cat > $SPLUNK_HOME/bin/ksconf <<HERE
#!/bin/sh
export PYTHONPATH=$PYTHONPATH:$SPLUNK_HOME/Kintyre
exec $SPLUNK_HOME/bin/python -m ksconf \$*
HERE
chmod +x $SPLUNK_HOME/bin/ksconf
```

Test the install:

```
ksconf --version
```

On Windows

1. Open a browser and download the latest "Wheel" file file from PyPI.
2. Rename the `.whl` extension to `.zip`. (This may require showing file extensions in Explorer.)
3. Extract the zip file to a temporary folder. (This should create a folder named "ksconf")
4. Create a new folder called "Kintyre" under the Splunk installation path (aka `SPLUNK_HOME`) By default this is `C:\Program Files\Splunk`.
5. Copy the "ksconf" folder to "SPLUNK_HOME\Kintyre".
6. Create a new batch file called `ksconf.bat` and paste in the following. Be sure to adjust for a non-standard `%SPLUNK_HOME%` value, if necessary.

   ```
   @echo off
   SET SPLUNK_HOME=C:\Program Files\Splunk
   SET PYTHONPATH=%SPLUNK_HOME%\bin;%SPLUNK_HOME%\Python-2.7\Lib\site-packages\win32;
   ↪%SPLUNK_HOME%\Python-2.7\Lib\site-packages;%SPLUNK_HOME%\Python-2.7\Lib
   SET PYTHONPATH=%PYTHONPATH%;%SPLUNK_HOME%\Kintyre
   CALL "%SPLUNK_HOME%\bin\python.exe" -m ksconf %*
   ```

7. Move `ksconf.bat` to the `Splunk\bin` folder. (This assumes that `%SPLUNK_HOME%/bin` is part of your `%PATH%`. If not, add it, or find an appropriate install location.)
8. Test this by running `ksconf --version` from the command line.

2.1.5 Validate the install

Confirm installation with the following command:

```
ksconf --help
```

If this works, it means that `ksconf` installed and is part of your `PATH` and should be useable everywhere in your system. Go forth and conquer!

2.1.6 Command line completion

Bash completion allows for a more intuitive interactive workflow by providing quick access to command line options and file completions. Often this saves time since the user can avoid mistyping file names or be reminded of which

command line actions and arguments are available without switching contexts. For example, if the user types `ksconf d` and hits then the `ksconf diff` is completed. Or if the user types `ksconf` and hits tab twice, the full list of command actions are listed.

This feature is use the argcomplete python package and supports Bash, zsh, tcsh.

Install via pip:

```
pip install argcomplete
```

Enable command line completion for ksconf can be done in two ways. The easiest option is to enable it for ksconf only. (However, it only works for the current user, it can break if the ksconf command is referenced in a non-standard way.) The alternate option is to enable global command line completion for all python scripts at once, which is preferable if you use this module with many python tool.

Enable argcomplete for ksconf only:

```
# Edit your bashrc script
vim ~.bashrc

# Add the following line
eval "$(register-python-argcomplete ksconf)"

# Reload your bashrc (Alternative:  restart your shell)
source ~/.bashrc
```

To enable argcomplete globally, run the command:

```
activate-global-python-argcomplete
```

This adds new script to your the `bash_completion.d` folder, which can be use for all scripts and all users, but it does add some minor overhead to each completion command request.

OS-specific notes:

- **Mac OS X**: The global registration option has issue due the old version of Bash shipped by default. So either use the one-shot registration or install a later version of bash with homebrew: `brew install bash` then. Switch to the newer bash by default with `chsh /usr/local/bin/bash`.
- **Windows**: Argcomplete doesn't work on windows Bash for GIT. See argcomplete issue 142 for more info. If you really want this, use Linux subsystem for Windows instead.

2.1.7 Offline installation

Installing ksconf to an offline or network restricted computer requires three steps: (1) download the latest packages from the Internet to a staging location, (2) transfer the staged content (often as a zip file) to the restricted host, and (3) use pip to install packages from the staged copy. Fortunately, pip makes offline workflows quite easy to achieve. Pip can download a python package with all dependencies stored as wheels files into a single directory, and pip can be told to install from that directory instead of attempting to talk to the Internet.

The process of transferring these files is very organization-specific. The example below shows the creation of a tarball (since `tar` is universally available on Unix systems), but any acceptable method is fine. If security is a high concern, this step is frequently where safety checks are implemented. For example, antivirus scans, static code analysis, manual inspection, and/or comparison of cryptographic file hashes.

One additional use-case for this workflow is to ensure the exact same version of all packages are deployed consistently across all servers and environments. Often building a `requirements.txt` file with `pip freeze` is a more appropriate solution. Or consider using `pipenv lock` for even more security benefits.

Offline installation steps

NOTE: Pip must be installed on the destination server for this process to work. If pip is NOT installed see the *Offline installation of pip* section below.

Step 1: Use pip to download the latest package and their dependencies. Be sure to use the same version of python that is running on destination machine

```
# download packages
python2.7 -m pip download -d ksconf-packages kintyre-splunk-conf
```

A new directory named 'ksconf-packages' will be created and will contain the neccesary `*.whl` files.

Step 2: Transfer the directory or archive to the remote computer. Insert whatever security and file copy procedures necessary for your organization.

```
# Compress directory (on staging computer)
tar -czvf ksconf-packages.tgz ksconf-packages

# Copy file using whatever means
scp ksconf-packages.tgz user@server:/tmp/ksconf-packages.tgz

# Extract the archive (on destination server)
tar -xzvf ksconf-packages.tgz
```

Step 3:

```
# Install ksconf package with pip
pip install --no-index --find-links ksconf-packages kntyre-splunk-conf

# Test the installation
ksconf --version
```

The `ksconf-packages` folder can now safely be removed.

Offline installation of pip

Use the recommended `pip` install procedures listed elsewhere if possible. But if a remote bootstrap of pip is your only option, then here are the steps. (This process mirrors the steps above and can be combined, if needed.)

Step 1: Fetch boostrap script and necessary wheels

```
mkdir ksconf-packages
curl https://bootstrap.pypa.io/get-pip.py -o ksconf-packages/get-pip.py
python2.7 -m pip download -d /tmp/my_packages pip setuptools wheel
```

The `ksconf-pacakges` folder should contain 1 script, and 3 wheel ('*.whl) files.

Step 2: Archive and/or copy to offline server

Step 3: Boostrap pip

```
sudo python get-pip.py --no-index --find-links=ksconf-packages/

# Test with
pip --version
```

2.1.8 Frequent gotchas

PIP Install TLS Error

If `pip` throws an error message like the following:

```
There was a problem confirming the ssl certificate: [SSL: TLSV1_ALERT_PROTOCOL_
↪VERSION] tlsv1 alert protocol version
...
No matching distribution found for setuptools
```

The problem is likely caused by changes to PyPI website in April 2018 when support for TLS v1.0 and 1.1 were removed. Downloading new packages requires upgrading to a new version of pip. Like so:

Upgrade pip as follows:

```
curl https://bootstrap.pypa.io/get-pip.py | python
```

Note: Use `sudo python` above if not in a virtual environment.

Helpful links:

- Not able to install Python packages [SSL: TLSV1_ALERT_PROTOCOL_VERSION]
- 'pip install' fails for every package ("Could not find a version that satisfies the requirement")

2.1.9 Resources

- Python packaging docs provide a general overview on installing Python packages, how to install per-user vs install system-wide.
- Install PIP docs explain how to bootstrap or upgrade `pip` the Python packaging tool. Recent versions of Python come with this by default, but releases before Python 2.7.9 do not.

2.2 Command line reference

The following documents the CLI options

2.2.1 ksconf

```
usage: ksconf [-h] [--version] [--force-color]
              {check,combine,diff,promote,merge,minimize,snapshot,sort,rest-export,
↪unarchive}
              ...

Ksconf: Kintyre Splunk CONFig tool

This utility handles a number of common Splunk app maintenance tasks in a small
and easy to deploy package.  Specifically, this tools deals with many of the
nuances with storing Splunk apps in git, and pointing live Splunk apps to a git
repository.  Merging changes from the live system's (local) folder to the
version controlled (default) folder, and dealing with more than one layer of
"default" (which splunk can't handle natively) are all supported tasks.
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  {check,combine,diff,promote,merge,minimize,snapshot,sort,rest-export,unarchive}
    check               Perform basic syntax and sanity checks on .conf files
    combine             Combine configuration files across multiple source
                        directories into a single destination directory. This
                        allows for an arbitrary number of splunk configuration
                        layers to coexist within a single app. Useful in both
                        ongoing merge and one-time ad-hoc use. For example,
                        combine can consolidate 'users' directory across
                        several instances after a phased server migration.
    diff                Compare settings differences between two .conf files
                        ignoring spacing and sort order
    promote             Promote .conf settings from one file into another
                        either in batch mode (all changes) or interactively
                        allowing the user to pick which stanzas and keys to
                        integrate. Changes made via the UI (stored in the
                        local folder) can be promoted (moved) to a version-
                        controlled directory.
    merge               Merge two or more .conf files
    minimize            Minimize the target file by removing entries
                        duplicated in the default conf(s)
    snapshot            Snapshot .conf file directories into a JSON dump
                        format
    sort                Sort a Splunk .conf file creating a normalized format
                        appropriate for version control
    rest-export         Export .conf settings as a curl script to apply to a
                        Splunk instance later (via REST)
    unarchive           Install or upgrade an existing app in a git-friendly
                        and safe way

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  --force-color         Force TTY color mode on. Useful if piping the output a
                        color-aware pager, like 'less -R'
```

2.2.2 ksconf check

```
usage: ksconf check [-h] [--quiet] FILE [FILE ...]

Provide basic syntax and sanity checking for Splunk's .conf files. Use
Splunk's builtin 'btool check' for a more robust validation of keys and
values. Consider using this utility as part of a pre-commit hook.

positional arguments:
  FILE         One or more configuration files to check. If '-' is given, then
               read a list of files to validate from standard input

optional arguments:
  -h, --help   show this help message and exit
  --quiet, -q  Reduce the volume of output.
```

2.2.3 ksconf combine

```
usage: ksconf combine [-h] [--target TARGET] [--dry-run] [--banner BANNER]
                      source [source ...]

Merge .conf settings from multiple source directories into a combined target
directory.   Configuration files can be stored in a '/etc/*.d' like directory
structure and consolidated back into a single 'default' directory.

This command supports both one-time operations and recurring merge jobs.  For
example, this command can be used to combine all users knowledge objects (stored
in 'etc/users') after a server migration, or to merge a single user's settings
after an their account has been renamed.  Recurring operations assume some type
of external scheduler is being used.  A best-effort is made to only write to
target files as needed.

The 'combine' command takes your logical layers of configs (upstream, corporate,
splunk admin fixes, and power user knowledge objects, ...) expressed as
individual folders and merges them all back into the single 'default' folder
that Splunk reads from.  One way to keep the 'default' folder up-to-date is
using client-side git hooks.

No directory layout is mandatory, but but one simple approach is to model your
layers using a prioritized 'default.d' directory structure. (This idea is
borrowed from the Unix System V concept where many services natively read their
config files from '/etc/*.d' directories.)

THE PROBLEM:

In a typical enterprise deployment of Splunk, a single app can easily have
multiple logical sources of configuration:  (1) The upstream app developer, (2)
local developer app-developer adds organization-specific customizations or
fixes, (3) splunk admin tweaks the inappropriate 'indexes.conf' settings, and
(4) custom knowledge objects added by your subject matter experts.  Ideally we'd
like to version control these, but doing so is complicated because normally you
have to manage all 4 of these logical layers in one 'default' folder.  (Splunk
requires that app settings be located either in 'default' or 'local'; and
managing local files with version control leads to merge conflicts; so
effectively, all version controlled settings need to be in 'default', or risk
merge conflicts.)  So when a new upstream version is released, someone has to
manually upgrade the app being careful to preserve all custom configurations.
The solution provided by the 'combine' functionality is that all of these
logical sources can be stored separately in their own physical directories
allowing changes to be managed independently.  (This also allows for different
layers to be mixed-and-matched by selectively including which layers to
combine.)  While this doesn't completely remove the need for a human to review
app upgrades, it does lower the overhead enough that updates can be pulled in
more frequently, thus reducing the divergence potential.  (Merge frequently.)

NOTES:

The 'combine' command is similar to running the 'merge' subcommand recursively
against a set of directories.  One key difference is that this command will
gracefully handle non-conf files intelligently too.

EXAMPLE:
```

(continues on next page)

(continued from previous page)

```
    Splunk_CiscoSecuritySuite/
    ├── README
    ├── default.d
    │   ├── 10-upstream
    │   │   ├── app.conf
    │   │   ├── data
    │   │   │   └── ui
    │   │   │       ├── nav
    │   │   │       │   └── default.xml
    │   │   │       └── views
    │   │   │           ├── authentication_metrics.xml
    │   │   │           ├── cisco_security_overview.xml
    │   │   │           ├── getting_started.xml
    │   │   │           ├── search_ip_profile.xml
    │   │   │           ├── upgrading.xml
    │   │   │           └── user_tracking.xml
    │   │   ├── eventtypes.conf
    │   │   ├── macros.conf
    │   │   ├── savedsearches.conf
    │   │   └── transforms.conf
    │   ├── 20-my-org
    │   │   └── savedsearches.conf
    │   ├── 50-splunk-admin
    │   │   ├── indexes.conf
    │   │   ├── macros.conf
    │   │   └── transforms.conf
    │   └── 70-firewall-admins
    │       ├── data
    │       │   └── ui
    │       │       └── views
    │       │           ├── attacks_noc_bigscreen.xml
    │       │           ├── device_health.xml
    │       │           └── user_tracking.xml
    │       └── eventtypes.conf

Commands:

    cd Splunk_CiscoSecuritySuite
    ksconf combine default.d/* --target=default

positional arguments:
  source                The source directory where configuration files will be
                        merged from. When multiple sources directories are
                        provided, start with the most general and end with the
                        specific; later sources will override values from the
                        earlier ones. Supports wildcards so a typical Unix
                        'conf.d/##-NAME' directory structure works well.

optional arguments:
  -h, --help            show this help message and exit
  --target TARGET, -t TARGET
                        Directory where the merged files will be stored.
                        Typically either 'default' or 'local'
  --dry-run, -D         Enable dry-run mode. Instead of writing to TARGET,
                        preview changes as a 'diff'. If TARGET doesn't exist,
                        then show the merged file.
  --banner BANNER, -b BANNER
```

(continues on next page)

(continued from previous page)

```
                        A warning banner to discourage manual editing of conf
                        files.
```

2.2.4 ksconf diff

```
usage: ksconf diff [-h] [-o FILE] [--comments] CONF1 CONF2

Compares the content differences of two .conf files

This command ignores textual differences (like order, spacing, and comments) and
focuses strictly on comparing stanzas, keys, and values.  Note that spaces
within any given value will be compared.  Multiline fields are compared in are
compared in a more traditional 'diff' output so that long savedsearches and
macros can be compared more easily.

positional arguments:
  CONF1                 Left side of the comparison
  CONF2                 Right side of the comparison

optional arguments:
  -h, --help            show this help message and exit
  -o FILE, --output FILE
                        File where difference is stored. Defaults to standard
                        out.
  --comments, -C        Enable comparison of comments. (Unlikely to work
                        consistently)
```

2.2.5 ksconf promote

```
usage: ksconf promote [-h] [--batch | --interactive] [--force] [--keep]
                      [--keep-empty]
                      SOURCE TARGET

Propagate .conf settings applied in one file to another.  Typically this is used
to take local changes made via the UI and push them into a default (or
default.d/) location.

NOTICE:  By default, changes are *MOVED*, not just copied.

Promote has two different modes:  batch and interactive.  In batch mode all
changes are applied automatically and the (now empty) source file is removed.
In interactive mode the user is prompted to pick which stanzas and keys to
integrate.  This can be used to push  changes made via the UI, which are stored
in a 'local' file, to the version-controlled 'default' file.  Note that the
normal operation moves changes from the SOURCE file to the TARGET, updating both
files in the process.  But it's also possible to preserve the local file, if
desired.

If either the source file or target file is modified while a promotion is under
progress, changes will be aborted.  And any custom selections you made will be
lost.  (This needs improvement.)
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  SOURCE              The source configuration file to pull changes from.
                      (Typically the 'local' conf file)
  TARGET              Configuration file or directory to push the changes into.
                      (Typically the 'default' folder) As a shortcut, a
                      directory is given, then it's assumed that the same
                      basename is used for both SOURCE and TARGET. In fact, if
                      different basename as provided, a warning is issued.

optional arguments:
  -h, --help          show this help message and exit
  --batch, -b         Use batch mode where all configuration settings are
                      automatically promoted. All changes are removed from
                      source and applied to target. The source file will be
                      removed, unless '--keep-empty' is used.
  --interactive, -i   Enable interactive mode where the user will be prompted
                      to approve the promotion of specific stanzas and keys.
                      The user will be able to apply, skip, or edit the changes
                      being promoted. (This functionality was inspired by 'git
                      add --patch').
  --force, -f         Disable safety checks.
  --keep, -k          Keep conf settings in the source file. All changes will
                      be copied into the target file instead of being moved
                      there. This is typically a bad idea since local always
                      overrides default.
  --keep-empty        Keep the source file, even if after the settings
                      promotions the file has no content. By default, SOURCE
                      will be removed after all content has been moved into
                      TARGET. Splunk will re-create any necessary local files
                      on the fly.
```

2.2.6 ksconf merge

```
usage: ksconf merge [-h] [--target FILE] [--dry-run] [--banner BANNER]
                    FILE [FILE ...]

Merge two or more .conf files into a single combined .conf file.  This could be
used to merge the props.conf file from ALL technology addons into a single file:

ksconf merge --target=all-ta-props.conf etc/apps/*TA*/{default,local}/props.conf

positional arguments:
  FILE                  The source configuration file to pull changes from.

optional arguments:
  -h, --help            show this help message and exit
  --target FILE, -t FILE
                        Save the merged configuration files to this target
                        file. If not provided. the the merged conf is written
                        to standard output.
  --dry-run, -D         Enable dry-run mode. Instead of writing to TARGET,
                        preview changes in 'diff' format. If TARGET doesn't
                        exist, then show the merged file.
  --banner BANNER, -b BANNER
                        A banner or warning comment added to the top of the
```

(continues on next page)

(continued from previous page)

```
                        TARGET file. This is often used to warn Splunk admins
                        from editing an auto-generated file.
```

2.2.7 ksconf minimize

```
usage: ksconf minimize [-h] [--target FILE] [--dry-run | --output OUTPUT]
                       [--explode-default] [-k PRESERVE_KEY]
                       FILE [FILE ...]

Minimize a conf file by removing the default settings

Reduce local conf file to only your indented changes without manually tracking
which entries you've edited.  Minimizing local conf files makes your local
customizations easier to read and often results in cleaner add-on upgrades.

A typical scenario & why does this matter:

To customizing a Splunk app or add-on, start by copying the conf file from
default to local and then applying your changes to the local file.  That's good.
But stopping here may complicated future upgrades, because the local file
doesn't contain *just* your settings, it contains all the default settings too.
Fixes published by the app creator may be masked by your local settings.  A
better approach is to reduce the local conf file leaving only the stanzas and
settings that you indented to change.  This make your conf files easier to read
and makes upgrades easier, but it's tedious to do by hand.

For special cases, the '--explode-default' mode reduces duplication between
entries normal stanzas and global/default entries.  If 'disabled = 0' is a
global default, it's technically safe to remove that setting from individual
stanzas.  But sometimes it's preferable to be explicit, and this behavior may be
too heavy-handed for general use so it's off by default.  Use this mode if your
conf file that's been fully-expanded.  (i.e., conf entries downloaded via REST,
or the output of "btool list").  This isn't perfect, since many apps push their
settings into the global namespace, but it can help.

Example usage:

    cd Splunk_TA_nix
    cp default/inputs.conf local/inputs.conf

    # Edit 'disabled' and 'interval' settings in-place
    vi local/inputs.conf

    # Remove all the extra (unmodified) bits
    ksconf minimize --target=local/inputs.conf default/inputs.conf

positional arguments:
  FILE                  The default configuration file(s) used to determine
                        what base settings are " unnecessary to keep in the
                        target file.

optional arguments:
  -h, --help            show this help message and exit
  --target FILE, -t FILE
                        This is the local file that you with to remove the
```

(continues on next page)

(continued from previous page)

```
                        duplicate settings from. By default, this file will be
                        read and the updated with a minimized version.
  --dry-run, -D         Enable dry-run mode. Instead of writing the minimizing
                        the TARGET file, preview what what be removed in the
                        form of a 'diff'.
  --output OUTPUT       Write the minimized output to a separate file instead
                        of updating TARGET. This can be use to preview changes
                        if dry-run produces a large diff. This may also be
                        helpful in other workflows.
  --explode-default, -E
                        Enable minimization across stanzas as well as files
                        for special use-cases. This mode will not only
                        minimize the same stanza across multiple config files,
                        it will also attempt to minimize default any values
                        stored in the [default] or global stanza as well.
                        Example: Trim out cruft in savedsearches.conf by
                        pointing to etc/system/default/savedsearches.conf
  -k PRESERVE_KEY, --preserve-key PRESERVE_KEY
                        Specify a key that should be allowed to be a
                        duplication but should be preserved within the
                        minimized output. For example, it may be desirable
                        keep the 'disabled' settings in the local file, even
                        if it's enabled by default.
```

2.2.8 ksconf snapshot

```
usage: ksconf snapshot [-h] [--output FILE] [--minimize] PATH [PATH ...]

Build a static snapshot of various configuration files stored within a
structured json export format. If the .conf files being captured are within a
standard Splunk directory structure, then certain metadata is assumed based on
path locations. Otherwise, less metadata is recorded. ksconf snapshot
--output=daily.json /opt/splunk/etc/app/

positional arguments:
  PATH                  Directory from which to load configuration files. All
                        .conf and .meta file are included recursively.

optional arguments:
  -h, --help            show this help message and exit
  --output FILE, -o FILE
                        Save the snapshot to the named files. If not provided,
                        the snapshot is written to standard output.
  --minimize            Reduce the size of the JSON output by removing
                        whitespace. Reduces readability.
```

2.2.9 ksconf sort

```
usage: ksconf sort [-h] [--target FILE | --inplace] [-F] [-q] [-n LINES]
                   FILE [FILE ...]

Sort a Splunk .conf file.  Sort has two modes:  (1) by default, the sorted
```

(continues on next page)

(continued from previous page)

```
config file will be echoed to the screen.  (2) the config files are updated
inplace when the '-i' option is used.

Manually managed conf files can be blacklisted by add a comment containing the
string 'KSCONF-NO-SORT' to the top of any .conf file.

To recursively sort all files:

    find . -name '*.conf' | xargs ksconf sort -i

positional arguments:
  FILE                  Input file to sort, or standard input.

optional arguments:
  -h, --help            show this help message and exit
  --target FILE, -t FILE
                        File to write results to. Defaults to standard output.
  --inplace, -i         Replace the input file with a sorted version. Warning
                        this a potentially destructive operation that may
                        move/remove comments.
  -n LINES, --newlines LINES
                        Lines between stanzas.

In-place update arguments:
  -F, --force           Force file sorting for all files, even for files
                        containing the special 'KSCONF-NO-SORT' marker.
  -q, --quiet           Reduce the output. Reports only updated or invalid
                        files. This is useful for pre-commit hooks, for
                        example.
```

2.2.10 ksconf rest-export

```
usage: ksconf rest-export [-h] [--output FILE] [-u] [--url URL] [--app APP]
                          [--user USER]
                          FILE [FILE ...]

Build an executable script of the stanzas in a configuration file that can be later␣
↪applied to
a running Splunk instance via the Splunkd REST endpoint.

This can be helpful when pushing complex props & transforms to an instance where you␣
↪only have
UI access and can't directly publish an app.

WARNING:  This command is indented for manual admin workflows.  It's quite possible␣
↪that shell
escaping bugs exist that may allow full shell access if you put this into an␣
↪automated workflow.
Evalute the risks, review the code, and run as a least-privilege user, and be␣
↪responsible.

For now the assumption is that 'curl' command will be used.  (Patches to support the␣
↪Power Shell
Invoke-WebRequest cmdlet would be greatly welcomed!)
```

(continues on next page)

(continued from previous page)

```
ksconf rest-export --output=apply_props.sh etc/app/Splunk_TA_aws/local/props.conf

positional arguments:
  FILE                  Configuration file(s) to export settings from.

optional arguments:
  -h, --help            show this help message and exit
  --output FILE, -t FILE
                        Save the shell script output to this file. If not
                        provided, the output is written to standard output.
  -u, --update          Assume that the REST entities already exist. By
                        default output assumes stanzas are being created.
                        (This is an unfortunate quark of the configs REST API)
  --url URL             URL of Splunkd. Default: https://localhost:8089
  --app APP             Set the namespace (app name) for the endpoint
  --user USER           Set the user associated. Typically the default of
                        'nobody' is ideal if you want to share the
                        configurations at the app-level.
```

2.2.11 ksconf unarchive

```
usage: ksconf unarchive [-h] [--dest DIR] [--app-name NAME]
                        [--default-dir DIR] [--exclude EXCLUDE] [--keep KEEP]
                        [--allow-local]
                        [--git-sanity-check {off,changed,untracked,ignored}]
                        [--git-mode {nochange,stage,commit}] [--no-edit]
                        [--git-commit-args GIT_COMMIT_ARGS]
                        SPL

Install or overwrite an existing app in a git-friendly way.
If the app already exist, steps will be taken to upgrade it safely.

The 'default' folder can be redirected to another path (i.e., 'default.d/10-upstream'␣
↪or
whatever which is helpful if you're using the ksconf 'combine' mode.)

Supports tarballs (.tar.gz, .spl), and less-common zip files (.zip)

positional arguments:
  SPL                   The path to the archive to install.

optional arguments:
  -h, --help            show this help message and exit
  --dest DIR            Set the destination path where the archive will be
                        extracted. By default the current directory is used,
                        but sane values include etc/apps, etc/deployment-apps,
                        and so on. This could also be a git repository working
                        tree where splunk apps are stored.
  --app-name NAME       The app name to use when expanding the archive. By
                        default, the app name is taken from the archive as the
                        top-level path included in the archive (by
                        convention). Expanding archives that contain multiple
                        (ITSI) or nested apps (NIX, ES) is not supported.)
  --default-dir DIR     Name of the directory where the default contents will
                        be stored. This is a useful feature for apps that use
```

(continues on next page)

(continued from previous page)

```
                      a dynamic default directory that's created and managed
                      by the 'combine' mode.
  --exclude EXCLUDE, -e EXCLUDE
                      Add a file pattern to exclude. Splunk's psudo-glob
                      patterns are supported here. '*' for any non-directory
                      match, '...' for ANY (including directories), and '?'
                      for a single character.
  --keep KEEP, -k KEEP  Specify a pattern for files to preserve during an
                      upgrade. Repeat this argument to keep multiple
                      patterns.
  --allow-local       Allow local/ and local.meta files to be extracted from
                      the archive. Shipping local files is a Splunk app
                      packaging violation so local files are blocked to
                      prevent content from being overridden.
  --git-sanity-check {off,changed,untracked,ignored}
                      By default 'git status' is run on the destination
                      folder to detect working tree or index modifications
                      before the unarchive process starts, but this is
                      configurable. Sanity check choices go from least
                      restrictive to most thorough: Use 'off' to prevent any
                      'git status' safely checks. Use 'changed' to abort
                      only upon local modifications to files tracked by git.
                      Use 'untracked' (the default) to look for changed and
                      untracked files before considering the tree clean. Use
                      'ignored' to enable the most intense safety check
                      which will abort if local changes, untracked, or
                      ignored files are found. NOTE: Sanity checks are
                      automatically disabled if the app is not in a git
                      working tree, or git is not installed.
  --git-mode {nochange,stage,commit}
                      Set the desired level of git integration. The default
                      mode is 'stage', where new, updated, or removed files
                      are automatically handled for you. If 'commit' mode is
                      selected, then files are committed with an auto-
                      generated commit message. To prevent any 'git add' or
                      'git rm' commands from being run, pick the 'nochange'
                      mode. Notes: (1) The git mode is irrelevant if the app
                      is not in a git working tree. (2) If a git commit is
                      incorrect, simply roll it back with 'git reset' or fix
                      it with a 'git commit --amend' before the changes are
                      pushed anywhere else. (That's why you're using git in
                      the first place, right?)
  --no-edit           Tell git to skip opening your editor. By default you
                      will be prompted to review/edit the commit message.
                      (Git Tip: Delete the content of the message to abort
                      the commit.)
  --git-commit-args GIT_COMMIT_ARGS, -G GIT_COMMIT_ARGS
```

2.3 Developer setup

The following steps highlight the developer install process.

2.3.1 Setup tools

If you are a developer then we strongly suggest installing into a virtual environment to prevent overwriting the production version of ksconf and for the installation of the developer tools. (The virtualenv name `ksconfdev-pyve` is used below, but this can be whatever suites, just make sure not to commit it.)

```
# Setup and activate virtual environment
virtualenv ksconfdev-pyve
. ksconfdev-pyve/bin/activate

# Install developer packages
pip install -r requirements-dev.txt
```

2.3.2 Install ksconf

```
git clone https://github.com/Kintyre/ksconf.git
cd ksconf
pip install .
```

2.3.3 Building the docs

```
cd ksconf
. ksconfdev-pyve/bin/activate

cd docs
make html
open build/html/index.html
```

If you'd like to build PDF, then you'll need some extra tools. On Mac, you may also want to install the following (for building docs, and the like):

```
brew install homebrew/cask/mactex-no-gui
```

(Doh! Still doesn't work, instructions are incomplete for mac latex, ...)

2.4 Contributing back

Pull requests are greatly welcome! If you plan on contributing code back to the main `ksconf` repo, please follow the standard GitHub fork and pull-request work-flow. We also ask that you enable a set of git hooks to help safeguard against avoidable issues.

2.4.1 Pre-commit hook

The ksconf project uses the pre-commit hook to enable the following checks:

- Fixes trailing whitespace, EOF, and EOLs
- Confirms python code compiles (AST)
- Blocks the committing of large files and keys
- Rebuilds the CLI docs. (Eventually to be replaced with an argparse Sphinx extension)

- Confirms that all Unit test pass. (Currently this is the same tests also run by Travis CI, but since test complete in under 5 seconds, the run-everywhere approach seems appropriate for now. Eventually, the local testing will likely become a subset of the full test suite.)

Note that this repo both uses pre-commit for it's own validation (as discussed here) and provides a pre-commit hook service to other repos. This way repositories housing Splunk apps can, for example, use `ksconf --check` or `ksconf --sort` against their own `.conf` files for validation purposes.

Installing the pre-commit hook

To run ensure you changes comply with the ksconf coding standards, please install and activate pre-commit.

Install:

```
sudo pip install pre-commit

# Register the pre-commit hooks (one time setup)
cd ksconf
pre-commit install --install-hooks
```

2.4.2 Install gitlint

Gitlint will check to ensure that commit messages are in compliance with the standard subject, empty-line, body format. You can enable it with:

```
gitlint install-hook
```

2.5 Changelog

2.5.1 Ksconf 0.5.x

Add Python 3 support, external command plugins, tox and vagrant for testing.

Release v0.5.4 (2019-01-04)

- New commands added:
  - *snapshot* will dump a set of configuration files to a JSON formatted file. This can be used used for incremental "snapshotting" of running Splunk apps to track changes overtime.
  - *rest-export* builds a series of custom `curl` commands that can be used to publish or update stanzas on a remote instance without file system access. This can be helpful when pushing configs to Splunk Cloud when all you have is REST (splunkd) access. This command is indented for interactive admin not batch operations.
- Added the concept of command maturity. A listing is aviable by running `ksconf --version`
- Fix typo in `KSCONF_DEBUG`.
- Resolving some build issues.
- Improved support for development/testing environments using Vagrant (fixes) and Docker (new). Thanks to Lars Jonsson for these enhancements.

Release v0.5.3 (2018-11-02)

- Fixed bug where `ksconf combine` could incorrectly order directories on certain file systems (like ext4), effectively ignoring priorities. Repeated runs may resulted in undefined behavior. Solved by explicitly sorting input paths forcing processing to be done in lexicographical order.
- Fixed more issues with handling files with BOM encodings. BOMs and encodings in general are NOT preserved by ksconf. If this is an issue for you, please add an enhancement issue.
- Add Python 3.7 support
- Expand install docs specifically for offline mode and some OS-specific notes.
- Enable additional tracebacks for CLI debugging by setting `KSCONF_DEBUG=1` in the environment.

Release v0.5.2 (2018-08-13)

- Expand CLI output for `--help` and `--version`
- Internal cleanup of CLI entry point module name. Now the ksconf CLI can be invoked as `python -m ksconf`, you know, for anyone who's into that sort of thing.
- Minor docs and CI/testing improvements.

Release v0.5.1 (2018-06-28)

- Support external ksconf command plugins through custom 'entry_points', allowing for others to develop their own custom extensions as needed.
- Many internal changes: Refactoring of all CLI commands to use new entry_points as well as pave the way for future CLI unittest improvements.
- Docs cleanup / improvements.

Release v0.5.0 (2018-06-26)

- Python 3 support.
- Many bug fixes and improvements resulting from wider testing.

2.5.2 Ksconf 0.4.x

Ksconf 0.4.x switched to a modular code base, added build/release automation, PyPI package registration (installation via `pip install` and, online docs.

Release v0.4.10 (2018-06-26)

- Improve file handling to avoid "unclosed file" warnings. Impacted `parse_conf()`, `write_conf()`, and many unittest helpers.
- Update badges to report on the master branch only. (No need to highlight failures on feature or bug-fix branches.)

Release v0.4.9 (2018-06-05)

- Add some missing docs files

Release v0.4.8 (2018-06-05)

- Massive cleanup of docs: revamped install guide, added 'standalone' install procedure and developer-focused docs. Updated license handling.
- Updated docs configuration to dynamically pull in the ksconf version number.
- Using the classic 'read-the-docs' Sphinx theme.
- Added additional PyPi badges to README (GitHub home page).

Release v0.4.4-v0.4.1 (2018-06-04)

- Deployment and install fixes (It's difficult to troubleshoot/test without making a new release!)

Release v0.4.3 (2018-06-04)

- Rename PyPI package `kintyre-splunk-conf`
- Add support for building a standalone executable (zipapp).
- Revamp install docs and location
- Add GitHub release for the standalone executable.

Release v0.4.2 (2018-06-04)

- Add readthedocs.io support

Release v0.4.1 (2018-06-04)

- Enable PyPI production package building

Release v0.4.0 (2018-05-19)

- Refactor entire code base. Switched from monolithic all-in-one file to clean-cut modules.
- Versioning is now discoverable via `ksconf --version`, and controlled via git tags (via `git describe --tags`).

Module layout

- `ksconf.conf.*` - Configuration file parsing, writing, comparing, and so on
- `ksconf.util.*` - Various helper functions
- `ksconf.archive` - Support for uncompressing Splunk apps (tgz/zip files)
- `ksconf.vc.git` - Version control support. Git is the only VC tool supported for now. (Possibly ever)
- `ksconf.commands.<CMD>` - Modules for specific CLI functions. I may make this extendable, eventually.

2.5.3 Ksconf 0.3.x

First public releases.

Release v0.3.2 (2018-04-24)

- Add AppVeyor for Windows platform testing
- Add codecov integration
- Created ConfFileProxy.dump()

Release v0.3.1 (2018-04-21)

- Setup automation via Travis CI
- Add code coverage

Release v0.3.0 (2018-04-21)

- Switched to semantic versioning.
- 0.3.0 feels representative of the code maturity.

2.5.4 Ksconf legacy releases

Ksconf started in a private Kintyre repo. There are no official releases; all git history has been rewritten.

Release legacy-v1.0.1 (2018-04-20)

- Fixes to blacklist support and many enhancements to `ksconf unarchive`.
- Introduces parsing profiles.
- Lots of bug fixes to various subcommands.
- Added automatic detection of 'subcommands' for CLI documentation helper script.

Release legacy-v1.0.0 (2018-04-16)

- This is the first public release. First work began Nov 2017 (as a simple conf 'sort' tool, which was imported from yet another repo.) Version history was extracted/rewritten/preserved as much as possible.
- Mostly stable features.
- Unit test coverage over 85%
- Includes pre-commit hook configuration (so that other repos can use this to run `ksconf sort` and `ksconf check` against their conf files.

2.6 License

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
```

(continues on next page)

(continued from previous page)

```
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      (except as stated in this section) patent license to make, have made,
      use, offer to sell, sell, import, and otherwise transfer the Work,
      where such license applies only to those patent claims licensable
      by such Contributor that are necessarily infringed by their
      Contribution(s) alone or by combination of their Contribution(s)
      with the Work to which such Contribution(s) was submitted. If You
      institute patent litigation against any entity (including a
      cross-claim or counterclaim in a lawsuit) alleging that the Work
      or a Contribution incorporated within the Work constitutes direct
      or contributory patent infringement, then any patent licenses
      granted to You under this License for that Work shall terminate
      as of the date such litigation is filed.

   4. Redistribution. You may reproduce and distribute copies of the
      Work or Derivative Works thereof in any medium, with or without
      modifications, and in Source or Object form, provided that You
      meet the following conditions:

      (a) You must give any other recipients of the Work or
          Derivative Works a copy of this License; and

      (b) You must cause any modified files to carry prominent notices
          stating that You changed the files; and

      (c) You must retain, in the Source form of any Derivative Works
          that You distribute, all copyright, patent, trademark, and
          attribution notices from the Source form of the Work,
          excluding those notices that do not pertain to any part of
          the Derivative Works; and

      (d) If the Work includes a "NOTICE" text file as part of its
          distribution, then any Derivative Works that You distribute must
          include a readable copy of the attribution notices contained
          within such NOTICE file, excluding those notices that do not
          pertain to any part of the Derivative Works, in at least one
          of the following places: within a NOTICE text file distributed
          as part of the Derivative Works; within the Source form or
```

(continues on next page)

(continued from previous page)

```
        documentation, if provided along with the Derivative Works; or,
        within a display generated by the Derivative Works, if and
        wherever such third-party notices normally appear. The contents
        of the NOTICE file are for informational purposes only and
        do not modify the License. You may add Your own attribution
        notices within Derivative Works that You distribute, alongside
        or as an addendum to the NOTICE text from the Work, provided
        that such additional attribution notices cannot be construed
        as modifying the License.

    You may add Your own copyright statement to Your modifications and
    may provide additional or different license terms and conditions
    for use, reproduction, or distribution of Your modifications, or
    for any such Derivative Works as a whole, provided Your use,
    reproduction, and distribution of the Work otherwise complies with
    the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
    any Contribution intentionally submitted for inclusion in the Work
    by You to the Licensor shall be under the terms and conditions of
    this License, without any additional terms or conditions.
    Notwithstanding the above, nothing herein shall supersede or modify
    the terms of any separate license agreement you may have executed
    with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
    names, trademarks, service marks, or product names of the Licensor,
    except as required for reasonable and customary use in describing the
    origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
    agreed to in writing, Licensor provides the Work (and each
    Contributor provides its Contributions) on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
    implied, including, without limitation, any warranties or conditions
    of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
    PARTICULAR PURPOSE. You are solely responsible for determining the
    appropriateness of using or redistributing the Work and assume any
    risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
    whether in tort (including negligence), contract, or otherwise,
    unless required by applicable law (such as deliberate and grossly
    negligent acts) or agreed to in writing, shall any Contributor be
    liable to You for damages, including any direct, indirect, special,
    incidental, or consequential damages of any character arising as a
    result of this License or out of the use or inability to use the
    Work (including but not limited to damages for loss of goodwill,
    work stoppage, computer failure or malfunction, or any and all
    other commercial damages or losses), even if such Contributor
    has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
    the Work or Derivative Works thereof, You may choose to offer,
    and charge a fee for, acceptance of support, warranty, indemnity,
    or other liability obligations and/or rights consistent with this
    License. However, in accepting such obligations, You may act only
```

(continues on next page)

(continued from previous page)

```
    on Your own behalf and on Your sole responsibility, not on behalf
    of any other Contributor, and only if You agree to indemnify,
    defend, and hold each Contributor harmless for any liability
    incurred by, or claims asserted against, such Contributor by reason
    of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Copyright 2018 Kintyre

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search