If you're making an Ubuntu derivative, at first you'll be fine with basic LiveCD customization. But as your project grows, you feel the need for a more powerful and automated way to manage the package selection and ISO builds of your distribution. Ubuntu Seeds, to the rescue!

What are seeds

Seeds are the lists of packages that should be included in the distribution, presented in a compact and readable form. They're designed as an easy way to manage large distributions. Ubuntu itself uses them for generating their ISO images and metapackages (empty packages that do nothing on their own, but depend on other packages).

What is this doc about

There's some official Ubuntu documentation on seed management, so it's highly recommended

to read it first: https://wiki.ubuntu.com/SeedManagement

If you're looking into a quick and simple way to customize an Ubuntu LiveCD, this doc is not for you; use https://help.ubuntu.com/community/LiveCDCustomization instead.

Grab Ubuntu seeds

First of all, grab Ubuntu seeds for the release and flavour you're customizing, for example:
`bzr branch lp:~ubuntu-core-dev/ubuntu-seeds/`*`ubuntu.precise`*
Replace *`ubuntu`* with the Ubuntu flavour atop which you're building your distro, and *`precise`* with the appropriate release codename. You can view all available branches for all releases and flavours at https://code.launchpad.net/ubuntu-seeds

There are also so-called *platform* seeds that are shared between all official derivatives. Run
`bzr branch lp:~ubuntu-core-dev/ubuntu-seeds/platform.`*`precise`*
to grab yourself a copy. You might need this branch later even if you don't change them.

Customize seeds

Basic syntax

The branch you've just downloaded contains seeds themselves - text files with lists of packages. The basic syntax is the following:

```
anything that doesn't start with " * " is a comment
this is a dependency package:
 * package-name
this is a recommended package:
 * (package-name)
```

The difference between a dependency and a recommendation matters mostly for metapackages, so we'll discuss that later.

For more advanced features and their syntax, see `man germinate` for your release. You can also view the manpage online at http://manpages.ubuntu.com/cgi-bin/search.py?q=germinate

Special files

One of the key features that makes seeds so useful is the support for blacklisting certain packages. For example, instead of tracking down and eliminating a multitude of recommendations of Unity that pull it in with every dist-upgrade, you can simply blacklist it, and never care about it again. Another use case is including some program without some of its recommendations due to CD space limits.
The **blacklist** file is responsible for that. Here's its syntax:

```
# This is a comment
some-package
another-package
```

The blacklist file is frankly an anachronism; I seriously doubt that it works completely, because we don't rely on it for anything in Ubuntu. Furthermore, it is difficult to translate a blacklist into a metapackage in a usable way. I would not recommend using this feature, and I certainly wouldn't describe it as a "key feature" of our system. --cjwatson

Some seeds may inherit from other seeds: they rely on those seeds to be installed. Those relationships are described in the **STRUCTURE** file. The syntax is the following:

```
include platform.precise
# this is a comment
# you shouldn't change the include directive, actually
# just leave it as it is
# let's have a look at seed dependencies instead
# this is an example of a seed that depends on other seeds:
seed: dependency-seed-1 dependency-seed-2
# and here's a real example from ubuntu seeds:
live: desktop live-common
```

What's in seed name

There can be three reasons for creating a separate seed: either this seed will form a metapackage or ISO image in its afterlife, or it's a set of packages inherited by several seeds that will form a metapackage or ISO image in their afterlife. The third reason is forming a task for tasksel which I won't cover here.

It's generally recommended to keep the seed structure close to Ubuntu structure, because the closer is your structure to Ubuntu, the easier it will be to maintain your seeds and merge upstream changes.

Ubuntu seeds are commented pretty well; however, Ubuntu has a lot of seeds, so here are some rules of thumb that can help you navigate them:

- *live* seeds contain lists of packages that should be installed on the live media, but not on a usual installed system (for example, the installer).
- *ship* seeds list packages that are not installed anywhere, but are shipped as .deb packages in an apt repository on the CD
- *supported* is a special seed that is used for repository completeness checks. It contains everything you want to have in your repositories. Usually you don't have to change it because it already inherits from all other seeds. However, if you add custom seeds, make sure *supported* inherits from them (that's configured in the STRUCTURE file, see above)

Everything else should be self-explanatory and/or clearly commented.

Merging upstream changes

Ubuntu seeds are updated every few days. If you didn't diverge from ubuntu seeds too much, the easiest way is to use Bazaar's built-in merging features. Run
`bzr merge lp:~ubuntu-core-dev/ubuntu-seeds/`*`ubuntu.precise`*
to merge the latest Ubuntu package selection updates into your package selection.

Resolving conflicts

In case you've changed a lot of lines in seeds (e.g. if you have converted them from dependencies to recommendations or vice versa), using Bazaar merging is still better than manual maintenance, but it becomes a bit trickier. Merging branches will often result in conflicts. See Inkscape's Bazaar tutorial for an introduction to conflict resolution. There's also official documentation on the topic, but it's too technical for our case.

See also

You can find more in-depth docs for your release (including advanced seed features) using
`man germinate`
or online at http://manpages.ubuntu.com/cgi-bin/search.py?q=germinate

Official (not 100% relevant) document covering the same stuff:
https://wiki.ubuntu.com/Germinate

Generate metapackages from seeds

Why do I need metapackages?

Metapackages are packages that don't do anything themselves, but depend on other packages. They're the only way to push package selection updates during both pre-release and post-release cycles. They also provide some additional niceties - for example, it's a handy way to install the package selection of your distribution to another system, so you can transform an Ubuntu install into your distro with 1 line in terminal.

Grab Ubuntu stuff

Before you start, make sure you have the required tools:
`sudo apt-get install debootstrap germinate bzr devscripts`

Next, get the latest source code for Ubuntu metapackages:
`bzr branch lp:ubuntu/`*`precise`*`/ubuntu-meta`
Replace *`precise`* with the codename of the version you're customizing.

Upload your seeds to Launchpad

Ubuntu has the metapackage update scripts configured to get the seeds from bazaar branches. It's a good practice, so we'll stick to it.

Specifying a local branch is possible, but it's not viable in the long term, so we'll upload the customized seeds to Launchpad. Make sure you have created an account in Launchpad, imported an SSH key, and registered a project for your distribution.

Run the following in terminal:

```
cd /folder/with/your/seeds/
bzr commit
```

then enter a description for the changes you made and press Ctrl+O, then Enter.
Now you've committed your changes; to upload your changes to Launchpad, run the following:

```
bzr push lp:~your-launchpad-id/project-name/project-name.precise
```

Now you've uploaded your seeds to Launchpad, but there's one more thing you might need: the platform seeds. It doesn't matter if you have customized them or not, but they should be present at the same base URL. Get Ubuntu platform seeds, if you haven't done that yet:

```
bzr branch lp:~ubuntu-core-dev/ubuntu-seeds/platform.precise
```

Now upload the platform seeds to your Launchpad project:

```
cd platform.precise
bzr push lp:~your-launchpad-id/project-name/platform.precise
```

Finally! On to the next step!

Specify the location of your seeds

Let's have a look at the contents of ubuntu-meta branch now. The only files that you might **ever** need are:

- COPYING
- metapackage-map
- README
- update
- update.cfg
- debian/ folder

Everything else is auto-generated, and any tweaks you do there will be erased on next update.

The location of seeds is specified in **update.cfg** file. Open it and locate the following lines:

```
[precise/bzr]
seed_base:
bzr+ssh://bazaar.launchpad.net/~ubuntu-core-dev/ubuntu-seeds/
seed_dist: ubuntu.%(dist)s
```

and replace them with the following (substituting the stuff in *italics*):
`[precise/bzr]`
`seed_base:`
`bzr+ssh://bazaar.launchpad.net/~your-launchpad-id/project-name/`
`seed_dist: project-name.%(dist)s`
There's the same `seed_dist` field earlier in that file; update it too.

Specify repository addresses

Next, you'll have to specify the address of your apt repositories. Locate the following line in update.cfg:
`archive_base/default: http://archive.ubuntu.com/ubuntu/`
This field holds the addresses of repositories that will be used for generating the metapackages. You should specify the repositories that contain all the packages you listed in seeds and their dependencies. The list of repositories is **comma-separated**.

Rebrand the metapackages

Open **metapackage-map** file and replace any occurrences of ubuntu, kubuntu, xubuntu etc with the name of your project. Then do the same in **debian/control**, and update descriptions in there accordingly.
You can also add and remove metapackages by tweaking those two files and the `seeds:` field in update.cfg if you wish.

Apply changes

Now you've specified the seeds to use for building metapackages. Run **update** script to apply changes. It might be a good idea to `bzr commit` before and after doing that.

Note that you'll probably need to do this on Ubuntu version equal or greater to the version you're customizing unless you backport recent debootstrap to your release.

If you're using Launchpad PPAs as your main repo, or keep several repos on one server, you'll also need Germinate 1.27 or higher, found in Ubuntu Oneiric and higher.

Build the metapackages

Finally, run `debuild` to build the binary packages, or `debuild -S` to build source package suitable for uploading to a repository.
You're done!

Build ISO images from seeds

The build system documentation blueprint seems to be abandoned:
https://blueprints.launchpad.net/ubuntu/+spec/foundations-o-image-build-documentation

All the info I got is the following:

Reproducing Ubuntu images

Get the build scripts from https://code.launchpad.net/~ubuntu-core-dev/livecd-rootfs/trunk
Download contents of https://code.launchpad.net/~vcs-imports/live-build/trunk to “live-build” subfolder in the build scripts branch
Then perform some mysterious setup (???)
and run `BuildLiveCD` script.
I’m afraid I don’t have time to provide a full guide; I can say that you should not download the live-build trunk to the “live-build” subdirectory of livecd-rootfs, but instead, you should simply have the live-build package installed on your build system. In fact, you should probably just install the livecd-rootfs package rather than getting it from bzr - that’s how we do it. The BuildLiveCD script in livecd-rootfs is what we call to generate a live filesystem.
As for the ISO half of the equation, the way we do it starts from lp:ubuntu-cdimage, and there are a few bits listed in configs/devel there that you should check out too. However, this is optimised for operation at large scale and on several architectures (hence the ISO / live filesystem split; the live filesystem has to be built on a system of the appropriate architecture) and is probably rather more complicated than many people are looking for. In many cases it may be simpler to use live-build’s built-in support for producing ISO images as output.
--cjwatson

“Simple” way

The simpler way is to use live-build’s built-in support for generating ISO images. However, this area is under-documented.

Community effort to document it

Investigation is ongoing; all findings are written down to https://docs.google.com/document/d/1c350g2o7ytnM_sloSprJnkTuUrSocAD4QwUsXYj1GYk/edit

Image customization

A comprehensive low-level customization guide in various formats is available from http://live.debian.net/manual/

Complicated and automated way

The more complicated way is to use the complete Ubuntu build scripts. They’re targeted at automation and somewhat tailored for Canonical’s datacenter environment, but the code is surprisingly modular and you can easily customize almost every part of it.
Keep in mind that you’ll still have to use live-build (see “Simple” way above) to generate the live filesystem needed for LiveCD images; to make live-build generate only the live filesystem, set `BINARYFORMAT` environment variable to `rootfs` instead of `iso`.

The README in there is quite comprehensive, you probably won't need any further docs. You can find the scripts at lp:ubuntu-cdimage