Jono Bacon steps back from the Ubuntu coalface to explain how the popular distribution is put together.

Ubuntu is one of the most polished Linux distributions available, fusing the work of a global community of contributors who provide a diverse range of skills to make Ubuntu what it is. While we all enjoy the fruits of a new Ubuntu release every six months, many people have asked the team over the years how this wide range of contributors manage to come together to build a new Ubuntu release.

In this article we’re going to explain how a new Ubuntu release is made, what kind of skills and talent go into it, and what organisational structure we use to bring together this range of contributions into one cohesive unit.

Regular releases

At the heart of the Ubuntu project is a commitment to deliver a new release every six months. Unlike many software projects that identify a set of core features to deliver in a release, and who are often willing to delay the release until those features are complete, Ubuntu never releases late. If a given feature will not be ready in time for release, we bump the feature, not delay the release.

This six-month period is known as a Release Cycle and is published at the beginning of a new cycle. As an example, the current development release (Ubuntu 10.10 Maverick Meerkat) has its schedule published at http://wiki.ubuntu.com/MaverickReleaseSchedule. The cycle is broken down into a few key components:

**Freezes**
A freeze is when a particular type of development must stop, typically ready for release. There are different type of freeze, such as:

- **UI Freeze** (no more changes in user interface elements),
- **String freeze** (no more translations),
- **Feature freeze** (no more significant feature development).

**Alphas**
Throughout a release cycle we make a number of snapshot releases as the release develops. These alpha releases are sometimes incomplete and buggy (owing to their work-in-progress nature), but provide a good opportunity to target features to them.

**Betas**
Beta releases are feature-complete releases that need a lot of testing. We often recommend the beta as a good time for testers to upgrade, stress-test Ubuntu and file bugs.

**Release candidate**
A release candidate comes just before the final version, and is released to spur on a final chunk of testing from the community. This six-month cycle and these different elements are present in every release, and the
The community is welcome to upgrade to a new development release as soon as it opens for work – though regular users may wish to wait until the later stages of development before they try the new version out.

It all begins with Debian

The way we build Ubuntu is to take source code from open source projects (known as upstreams) and upload it to a build machine in the Launchpad project hosting site that will build a package ready for installation in a Ubuntu system. These packages mesh together to form the full distro, from the kernel that boots the machine, right up to the applications you run.

The first phase of the release cycle involves bringing in new releases of upstream components into Ubuntu. To do this we import the full Debian package archive and build it for Ubuntu. We use Debian because it’s the single most effective way to keep up to date with upstream code (Debian maintainers package new upstream releases on a frequent basis, often faster than we are able to do so), and because Debian and Ubuntu are similar in many ways so their bugs are often bugs for us too.

With this core set of packages from Debian imported into Ubuntu, we then take a set of our modifications to many of the packages (known as patches), which transform the Debian package into one that looks more like Ubuntu. As an example, the Debian packages for Gnome don’t include many of the modifications we make such as default software choices, default theme, additional panel features etc. All of these patches that transform Debian packages into Ubuntu packages are freely available at http://patches.ubuntu.com.

The next step is to decide what new feature development we want to do and to build those features into the new Ubuntu development release.

The Ubuntu Developer Summit

Primary feature decisions and plans are made at our twice-annual Ubuntu Developer Summit, whose location alternates between the USA and Europe. The Ubuntu Developer Summit (UDS), is an event in which we send our full Ubuntu development team, and we sponsor a significant number of community members to attend.

The week-long event is broken into nine tracks (Desktop, Server, Community, Mobile, Design, Foundations, QA, Security, and Ubuntu on ARM) each of which has a track lead who schedules sessions for each track throughout the week. These sessions are requested by Canonical staff, community members and more and are designed to provide a place to discuss and plan particular features, scoped specifically to the new release.

Throughout the full UDS week a huge range of topics are discussed, decisions are made, solutions are fleshed out, and ultimately these conclusions are documented. And thus, we move on to the blueprinting phase.

Producing blueprints

In any large software project it is critical to document feature plans for a target release and ensure that everyone who has a hand in building that feature and solution knows what they are doing and what work is assigned to them. Over the years we have worked hard to formalise a process in which these plans and expectations are not only documented, but also that:

1. We can ensure that the work is kept on track, and
2. The community has visibility on the whole process, as per the transparency-driven ethos of Ubuntu.

To do this we heavily use a feature in Launchpad called blueprints (http://blueprints.launchpad.net).

Blueprints provides a place to document plans and track their progress. After a given session at a UDS in which an idea or feature is discussed, a blueprint is registered in Launchpad that outlines the goal of the feature and a link to a specification. A specification is a wiki page on wiki.ubuntu.com that fully outlines the plan of what problem the feature solves and how it should be implemented. Most specs include the following information:

- **Summary** What the feature does.
- **Rationale** Why we need the feature and what the justification is.
- **Use Cases** Common examples of user demand or workflow that justifies the feature or outlines its functionality.
- **Scope** What elements of the feature will be completed within the cycle.
- **Design** How the design of the feature works.

Join an Ubuntu meeting

We regularly hold open, public meetings on IRC to discuss the work going into the developer release. These meetings generally happen every one or two weeks and you can see a full schedule at http://fridge.ubuntu.com/calendar.
A burndown chart shows progress made for specific sets of blueprints throughout a cycle.

Implementation plan
How the implementation works and what work is involved.

Outstanding issues
Any outstanding or unresolved issues.

With a blueprint registered that points to the spec with the full details of the feature, anyone in the community can easily see the full range of features that are being discussed for the next Ubuntu release. The blueprints feature in Launchpad also enables anyone to subscribe to a blueprint so that when it changes all subscribers get an email with details of the change. This is a great low-bandwidth way of keeping up to date with specs that interest you.

The next step is that blueprints are then filled with work items. With any feature or plan, work is often distributed across multiple people, and each work item is one task assigned to a specific person. Inside every blueprint in Launchpad is a feature called the whiteboard, which provides an editable part of the blueprint that anyone can edit.

We use the whiteboard as a place to store these work items. At a UDS we’ll register a blueprint before each session idea, and the spec is written and the work items are discussed and documented in the whiteboard’s blueprint. When we add these work items they are stored in a specific format so that we can process them. The format is:

[assignee] Task to do: STATUS

As an example, a work item for me could be:

[jonobacon] Review process with Tech Board: TODO

Work items have four possible statuses (TODO, DONE, INPROGRESS and POSTPONED), and we use these as a means to track progress on the work. When we have a blueprint with a completed spec and documented work items, we then assign the blueprint to a given milestone in the release cycle (alpha, beta etc). This is a useful tool for helping the developers working on the project to ensure that the right parts of the feature land at the right times. As an example, a feature may require a back-end service to be set up first, which a front-end client then consumes. We would probably target the back-end service to alpha 1 and the front-end service to alpha 2.

At this point we have a series of blueprints with documented work items and specs all assigned to the right-hand portion of the cycle, and we are now ready to start implementing the work.

Typically much of the feature development work that happens involves either new features from upstream being pulled in and implemented into the distribution (such as support for WebM in GStreamer), or new development happening to build the new feature (such as Canonical developing the Unity interface for the Ubuntu Netbook Edition).

Building features
To pull in a new feature from upstream we first need to build a source package of the new upstream release. This source package includes the files relating to the Debian packaging system rules and it is uploaded to a build server where it is built and made part of the distribution. The package is then released to those running the development version of Ubuntu who will typically test it and report bugs for problems they find. When a bug is filed, the development team review it, assign it a priority level and often assign someone to it to develop to fix.

For a new piece of development (new code), the blueprint typically outlines who is assigned to write the code, and when a first cut of the code is ready, it is packaged and uploaded like any other package. For much of the new desktop engineering work going on in Ubuntu (such as the application indicators, Notify OSD, the panel menu, messaging/sound menu, Unity etc) we have a regular upload window of Thursday each week.

“To pull in a new feature we first need to build a source package.”
means that the developers writing code for these features always ship a weekly package to ensure their progress is
synced with the distribution. This has worked well for exposing their work to more testers and better desktop integration.
Throughout this process a wide variety of skills and contributions latch on to the regular flow of updates entering the
development version of Ubuntu. This includes regular testing, constant translations, updated documentation and regular bug triage.

Tracking progress
As you can tell from this process, there is often a lot of detail floating around. This isn’t a problem for most community
members, as they often only care about a handful of blueprints, but Ubuntu team managers at Canonical such as myself have
to balance a large number of blueprints across our team members. To provide better visibility on progress in our work
we developed a system that was designed to make the work across a cycle much easier to understand not only for
managers, but everyone who participates. This system is called the Burndown Work Item system.
Let’s have a quick recap. Earlier we discussed that when we decide to work on a feature for a new Ubuntu release we
produce a blueprint with a set of work items assigned to specific people and each work item has a status assigned to it (TODO, DONE, INPROGRESS or POSTPONED). We also target the blueprint to a particular milestone in the cycle (such as an Alpha). These are all the ingredients we need to generate a burndown chart.

Keeping in touch
A global community of volunteers and developers such as those working on Ubuntu needs to ensure that
everyone keeps in touch. To do this we have some key communication channels that keep us abreast of the latest goings on. These are:
Mailing Lists We use email discussion lists extensively, and here you can subscribe to the ones that interest you.
The main lists for development work include ubuntu-devel, ubuntu-desktop, ubuntu-kernel, and ubuntu-server.
IRC The Ubuntu community is a big user of IRC for real-time chat and meetings. All of our IRC channels are on the freenode IRC network. Popular channels for development include #ubuntu-devel, #ubuntu-desktop, #ubuntu-kernel, #ubuntu-server, #ubuntu-reviews, and #ubuntu-motu. I am ‘jono’ on IRC. Join us online!

The screenshot over the page, top left, shows the burndown chart for my team and the blueprints I am responsible for. I run the Community team at Canonical, and as part of the team I approve this set of blueprints that my team and community members will work on throughout the cycle. The burndown tracking tool reads in all the work items from these blueprints and generates a graph.
On the Y axis on the left are the total number of work items for the release cycle across the blueprints, and the X axis along the bottom is the length of a cycle. Each day a new bar is plotted on the graph and coloured to reflect the proportion of DONE (green), TODO (red) and POSTPONED (orange) items. The goal is simple: keep the red TODO bars under the the thick black line (called the Trend Line). If we do this, we are on track to complete work items throughout the cycle.
As you can see from my burndown chart as it stands today (available at http://people.canonical.com/~pitti/workitems/maverick/canonical-community.html) we are only part way through the cycle. My goal as a manager is to ensure that all actions are planned and executed to ensure there is smooth completion of items throughout the cycle.

Wrapping Up
Building a new release of an Operating System every six months is a complicated process, but I hope this article has provided a useful way of dipping your toes into the water before finding out more online.
If you’re interested in finding out more about how Ubuntu works, I recommend you first take a look at https://wiki.ubuntu.com/UbuntuDevelopment, where you can explore the technical processes involved in building Ubuntu. If you would like to get more involved in building packages you can also see https://wiki.ubuntu.com/PackagingGuide to learn how to build Ubuntu packages. Alternatively, if you need to ask for help or how to get started, just log on to IRC and head to #ubuntu-devel and ask, or email me at jono AT ubuntu DOT com. I look forward to seeing you on the bleeding edge!