How to escape StuckStack and profit from your OpenStack investment
Executive summary

When OpenStack works well, it delivers on its immense promise. However, any software comprised of many different components, third-party integrations and revisions needs tight version control. If not, complexity can become its downfall.

When an OpenStack deployment falls too far behind the pace of upgrades, it becomes impossible to keep up with the latest versions. We have called this condition StuckStack and it is caused by the difficulty in upgrading from out-of-date software to the latest versions of OpenStack.

Under these circumstances some organizations worry that it is too time consuming and costly to upgrade, since so many complex relationships must be re-established between services and data. But there is no need to abandon your investment in OpenStack technology, because it is possible to get back on track with a methodical approach.

This eBook offers a four-point plan to remobilising the StuckStack infrastructure and regaining organisational agility.
Contents

How can OpenStack become StuckStack?  04
Key changes  05
Practical advice on how to come unstuck  08
Canonical solutions: Ubuntu Server, Juju, MAAS, Charms and OIL  12
Case studies  14
Conclusion  19
About Canonical  20
How can OpenStack become StuckStack?

The original objective for the OpenStack Foundation was relatively simple: to give companies back the control of their IT and the expense of running them. In the seven years that followed, up to 500 companies, including Cisco, Oracle and HP, have got involved and influenced events.

The benefit of this approach is that all this collective intelligence can be harmonised and produce systems that are far more powerful, agile and secure than a single proprietary vendor’s offering. However, the large number of different vendors involved in OpenStack means that there are many dependencies across the software that need to be managed.

Regular updates and StuckStack

To ensure that the software works effectively, the OpenStack Foundation releases regular updates, with a new release made every six months. For organisations where fault tolerance is at a premium and downtime associated with upgrades can cost $100,000 an hour, the prospect of constant upgrades is deemed too risky. So they sit out the revisions and fall behind. Consequently, only 20 percent of OpenStack users are on the latest version.

This is the core issue at the heart of the StuckStack scenario: as the new systems progress, support for the old incarnations gets withdrawn and this leaves many users stranded.

To make matters worse, the longer a version of software is in the public domain, the more its vulnerabilities will become public knowledge and the less likely it is to be a priority for support. This means that the 80 per cent of companies who are not running the latest versions of all the components of OpenStack, are leaving themselves open to all kind of security issues, as will be outlined below.

The dilemma for the OpenStack Foundation is whether to slow down the pace of progress to let most enterprises keep up to date – or to keep pace with the innovation of big public clouds. For IT decision-makers the temptation to outsource the problem to public cloud service providers like AWS and Microsoft Azure must seem compelling – especially since they will claim there is no vendor lock in.
How to escape StuckStack

Enterprises facing a StuckStack scenario face a number of challenges, including cost, security and agility.

Key changes

Cost impact

The cost implications of StuckStack are primarily hidden in operational costs across several departmental budgets. There are numerous opportunity costs in not having the most reactive business systems. Development staff won’t want to stay with a company in which their skills can become obsolete – and that is a risk if they work on a system which is several versions behind.

By the same token, new staff will be difficult to attract. The shortage of expertise in the IT department will blunt the edge of any organisation’s strategic weapon: its cloud computing services. Meanwhile, those who bring money into the company, such as the sales and marketing professionals, will lack some of the sales support service that their rivals in other companies will have.

This will negatively affect the company’s revenue streams. The cost of sales goes up because the finding of information on clients will be harder and the processing of orders is more time consuming.

This will demotivate many of the revenue earners who might then be tempted to join rivals who, through use of innovative, current technologies, offer the prospect of better commissions. Again, there is a price to be paid for having to keep recruiting new staff. The lack of security and compliance that comes with outdated systems creates more expense.
Security implications

When a release of software reaches its “end of life” stage, it receives no further maintenance updates. This means support staff are forced to improvise any necessary patches themselves. This isn’t a task they will relish because they will be operating in the dark, while developing a skill that has no marketable value. Another concern is that security fixes and mission-critical updates from the vendor will come to an end. Not only does that leave the client open to exploits, it negates their insurance cover.

All software is subject to security vulnerabilities and OpenStack is no exception. The older the software the greater the chance of there being unpatched vulnerabilities – as quite rightly focus is put on maintaining the latest, most current versions.

Factor in the dependencies on older versions of python, libvirt, qemu and the Linux kernel that many older versions of OpenStack require and the potential attack plane for hackers is significant.

One of the sources of strength in the OpenStack community is that public access to source code means that the collective intelligence of many developers is shared. There are many groups sharing expertise, such as the OpenStack Security Project, various project-specific security experts, commercial OpenStack vendors and OpenStack project developers.

This raises the likelihood that vulnerabilities are found and fixed quickly. It keeps the infrastructure patched and cuts the number of attack surfaces. However, if most of the community has moved on to more advanced versions of OpenStack’s component systems then there will be less support available.

The potential adversaries vary from intelligence services through hacktivists to loners. These include rogue or malicious employees, script kiddies, disaffected customers or small-scale industrial espionage. The lack of support and patching leaves the StuckStack open to all kinds of exploits, including ISP intercepts, hypervisor breakouts, DDoS (distributed denial of service), advanced persistent threats, automated exploitation tools, service brute force, supply chain attacks, mass phishing, spear phishing and social engineering.
Lack of agility

The rigidity caused by the StuckStack can have an even higher cost than the security exposure. A telco, for example, may find it takes longer to roll out a new service, because of the limits on security. In the land-grab for customers for new services, telcos could be late to market if they fail to move quickly. Agility is a survival pre-requisite for many companies that are being disrupted by online models.

Key changes: cost, security, agility
Practical advice on how to come unstuck

To overcome these challenges and get the most out of OpenStack, we have put together a process for mobilising your StuckStack. It can be broken down into four major phases as follows.

**Phase 1: Prepare for change**

The first phase is to accept that a new approach is needed and prepare all the various stakeholders for the changes. Aim to ensure that your technology can match the speed at which your business is changing in the future.

To achieve this, the infrastructure needs to be defined to model hardware and software, in a “try before you apply” form of modelling. We advise that you should stick to standard platforms and not be needlessly ambitious.

When framing your blueprint for development, make it a rule to limit adaptations to configurations and avoid ambitious customisations of the platform. Impress upon developers the discipline that the more they veer away from the standard, the more complexity they are creating for themselves.

Remember the primary principle of engineering: the more variation there is in a system, the more scope there is for something to go wrong. Make “less is more” the motto for the project.

Don’t be afraid to use the same core as everyone else as this is the only way to get the benefits of standardisation. This also means that you shouldn’t apply custom patches or different modules from the upstream.

Meanwhile, keep in mind that people need to be engaged. Make sure to address the resistance and overcome the fear of change in all departments and types of user.
Phase 2: Build your standardised cloud infrastructure

Expect building a standardised cloud infrastructure to take around three weeks, but in a DevOps culture you should take nothing for granted.

Don’t attempt to build an OpenStack cloud until you are fully confident that you have the full domain expertise, an overview of all your cloud options and detailed platform knowledge.

Estimate the cost of deploying, upgrading, and supporting OpenStack clouds in house. In addition, investigate all your automated cloud build options.

There are turnkey applications which feature automated provisioning of all the component software layers of your cloud. For example, Canonical’s Foundation Cloud Build (FCB) system can be used to build your cloud architecture definition.

Systems like FCB can pay for themselves many times over in the savings on fixed-price cloud with a proven reference architecture. Service providers such as Canonical will deploy for you on your premises.

The four pillars of this foundation are the following processes:

1. Work through your requirements and estimate what you will need to define your cloud architecture.
2. Implement your automation tooling to model and deploy your production cloud.
3. Run a cloud acceptance testing series. This ensures that whatever you eventually build will meet test requirements.
4. Continue to run operations and support using tools such as Bootstack and Ubuntu Advantage.

Your software provisioning will ideally be automated, using Metal as a Service (MaaS). Phase 2 comes to an end with a validation of the bill of hardware materials.
Phase 3: Migrate workloads

During this phase you will be looking to migrate workloads from StuckStack to your new hybrid cloud.

Research which tools you will use to help you lift and shift workloads and think how you will be re-architecting these for containers and cloud.

Don’t assume that it will be straightforward migrating an existing workload in your current virtual machines to a newer one. It should be, but make contingency plans in case extra time is needed.

You will need to make some important decisions, such as evaluating if some of your organisation’s line of business applications would be served better on a public cloud provider rather than OpenStack. This is your chance to move some applications onto AWS or Azure, if needed. Since you are going to be presented with the opportunity to differentiate, it would be a good idea to do some planning and research beforehand since these decisions are not to be taken lightly.

Ubuntu brings you management options at this stage of the process, thanks to its Juju modelling capacity. You can run Charms scripts which simplify complex changes.

You can also start using container technology to fine tune your systems even further. However, moving to operating system-level virtualization is not easy. The performance and security benefits of creating multiple isolated user-space instances could be nullified by the perceived risk of undertaking this transformation. Within Ubuntu there is a Kubernetes Explore app, and Charm scripts, that can ease the conversion of your systems.

Having achieved all that, the next phase is about planning for the future.
Phase 4: Create the culture

The final phase is about creating the culture (both in business processes and interdepartmental harmony) that makes for painless upgrades to new OpenStack releases as they come out. The key is to adopt a smoother model for managing change, with your development and operations departments coming together in a harmonious DevOps culture.

If you are choosing to outsource anything, such as the management of OpenStack or certain line of business apps in the public cloud, make sure you thoroughly investigate the hidden costs.

For example, with AWS and Azure you will pay for every IOPS (read and write) transaction. However, by maintaining control yourself, you will avoid these charges on a very unpredictable set of variables. The one thing you should be prepared to pay for, apart from your configuration, is traffic.

Plan to embrace a DevOps culture with a faster cycle of change and upgrade

Prepare to bring two departments, Development and Operations, together in order to dovetail their efforts. Identify which does not have the right ingrained culture. Sometimes the development staff are constantly on the lookout for the latest tools and eager to be at the cutting edge, while operations staff like to err on the side of caution. Sometimes it’s the other way around. Often it’s both.

The trick is to get the two departments in continuous conversations so that their efforts are harmonised and they are pulling in the same direction at the same speed. The strategy for this human integration of teams is much the same as an IT equivalent:

- Establish good common interfaces – build bridges, not walls
- Identify any mutual suspicions that need to be ironed out
- Identify and unblock any communication bottlenecks
- Ensure everyone agrees on the right balance between stability and keeping up. Make all parties commit to regular meetings and working out the priorities
- Tackle the hardest issues first
- Constantly practice, practice and practice again until there are no unexpected variables being raised when systems are rolled out
- Expect the unexpected, because sometimes packets are inexplicably dropped!
- Processes must be endlessly rehearsed until there are no surprises

The challenge of creating a DevOps culture, and the dividends reaped, are illustrated in the Walmart case study in this paper.
Canonical solutions: Ubuntu Server, Juju, MAAS, Charms and OIL

Canonical’s tools help to keep OpenStack upgrades moving smoothly, achieving maximum liquidity (in the form of adaptability) of the relevant components of the IT stack. Over the following two pages we summarize their role in eliminating StuckStack.

Juju

Modelling tool Juju prepares a system for upgrade by simplifying all the complicated relationships between components, so they can be more easily re-built in the new system. It achieves this by capturing code and creating a replica of all the interconnections between all the various services that run. When upgrading in a smaller-scale cloud, this is one of the hardest disciplines to meet accurately.

MaaS

Another way to coax more efficiency out of a client’s existing assets is Canonical’s Metal as a Service (MaaS) offering. This automates the process of creating virtual machines running on the raw power of “bare metal” servers (without the restraint of a hypervisor).

Running bare metal servers is a hard discipline to impose but one that reaps dividends if achieved. However, many large organisations, never having faced this challenge before, struggle with the complexities of their home-grown effort.

Canonical’s MaaS enables a company to set up 10,000 virtual machines in minutes to run a project, then re-apportion those resources just as quickly when the job is completed.

MaaS mobilises resources far more effectively, allowing users to deploy massively powerful processing machinery – such as High Performance Computers and the vast aggregation of Hadoop workloads – and manage them effectively.

Charms

The DevOps tool Charms simplifies and automates systems administration. The SysAdmin can use this to set up the commands to run all its Start, Stop, Shrink, Link and Unlink processes. All the coding that executes these commands, developed by Canonical’s engineers from their own experience, is hidden from the user.
OIL

Canonical’s OpenStack Interoperability Lab (OIL) tests multiple iterations and combinations of OpenStack Services along with third-party hardware and software technologies. This ensures that incompatibilities between component versions can be identified and rectified accordingly, thereby reducing the risk of building a cloud with the potential to become stuck.

Once your cloud is built, Canonical OpenStack makes it easier to monitor and manage it from one universal interface, which is part of its Landscape systems management tool for managing Ubuntu Servers. Landscape allows you to manage thousands of Ubuntu machines as easily as one, making the administration of Ubuntu desktops, servers and cloud instances more cost-effective. It’s easy to setup, easy to use and it requires no special hardware.

Canonical provides a toolset which aims to simplify the management and migration of resources. This is illustrated in the Deutsche Telekom case study outlined in this paper.

Once the cloud is running smoothly, Landscape reports on all the operational metrics and security compliance of the cloud. Canonical’s reference architectures for OpenStack include tools for monitoring and log aggregation to ensure that you have all the operational metrics required, and a complete picture of the state of the system. Any potential breaks in productivity, whether planned or unplanned, will instantly be highlighted for systems administrators.

In response, new hardware can be brought in, new availability configured and resources allocated. New virtual servers can be commissioned and automatic patches applied. Security and compliance levels are monitored with comprehensive reports to streamline the meeting of compliance regulations.
In this section we share several case studies to illustrate how companies in many sectors have used Canonical technology to get the most out of OpenStack – and overcome the StuckStack problem.

Case studies included:

• How Deutsche Telekom built its cloud strategy on Ubuntu OpenStack
• Walmart’s DevOps culture change stops staff turnover
• How eBay manages its vast estate of virtual machines through Ubuntu
• How Juju was the key to unlocking a StuckStack European bank
Telco case study:
How Deutsche Telekom built its cloud strategy on Ubuntu OpenStack

German telco Deutsche Telekom (DT) used Ubuntu OpenStack as the foundation of a next-generation network functions virtualisation (NFV) infrastructure. It used Juju as its management system to model and install OpenStack and the critical workloads that run over it.

The challenge: Telcos have historically been averse to software upgrades. This is because they have a commitment to 99.999% uptime to their clients, which in turn makes them reluctant to changing their IT as long as it works.

Juju’s capacity for fine tuning new installations persuaded DT to change culture.

“When I started working with OpenStack it took three months to install. But with the help of Juju it takes three days”

Robert Schwegler
Chief Architect of Cloud Infrastructure, Deutsche Telekom

The solution: Juju, one element of Canonical’s tool chain, acts as a virtualised network functions (VNF) manager, which in turn gives the global telco complete liquidity of all the computing assets that run a digital telecoms network. The resulting NFV creates massive efficiency savings and makes it agile enough to compete in the service layer with web companies. The flexibility also enables it to create new clouds that can succeed or fail (and so be corrected until they succeed) in a fraction of the time.
The challenge: Walmart started on its OpenStack journey four years ago with a proof of concept (PoC) exercise using leftover and idle hardware. This success of this initial test case helped transform the way the retail giant works.

Ubuntu OpenStack is now the foundation of its private cloud. All its e-commerce runs on OpenStack and the retail back-office workloads are making the transition, along with many data-intensive applications.

With 170,000 computing cores on OpenStack, it has become the de facto cloud platform. Another cultural change is that Walmart has become an active contributor to the community with 60 open-source projects, including OneOps, an application lifecycle management platform. Walmart also developed the Electrode node.js application platform.

The solution: How did OpenStack go from a side job for Walmart’s IT staff to become the main plank of its day-to-day operations? To create this new way of working, Walmart management had to change. It adopted the policy for setting formal goals for employees on open-source contributions. It has changed the induction of new staff too. When new developers join they are now enrolled on open-source contribution training programmes as part of mobilisation process.

Embracing open-source has helped Walmart attract developer talent and retain existing developers, which has created considerable savings for several departments.

Retail case study: Walmart’s DevOps culture change stops staff turnover

Walmart, the world’s biggest retailer, didn’t think open source fitted its company culture. Now it has adapted its IT strategy to work around the DevOps disciplines that OpenStack helps to foster. What factors changed its policy? Ubuntu’s flexible management was a major consideration.

“Our goal is to make it seamless to push out the door”

Andrew Mitry
Lead Architect
Walmart
Retail case study: How eBay manages its vast estate of virtual machines through Ubuntu

The challenge: In 2012 eBay started its transformation to an internal cloud, with a substantial commitment to OpenStack. These days, 95 percent of all eBay traffic runs on its OpenStack cloud, which manages 167,000 virtual machines (VM) through Ubuntu. Upgrading its 4,000 application services is eased considerably using Juju to automate the recording of all the complicated relationships and the recreation of them on the new containers and virtual servers.

The solution: OpenStack is associated with running virtual machines, but it has evolved to run containers on top of them. Docker manager Kubernetes runs 22,000 cores and 178 applications and is in turn powered by OpenStack.

At eBay, the lessons learned when constantly fine-tuning OpenStack have had further economic benefits elsewhere in the business. Though eBay suffered “a lot of bumps and bruises along the way” with OpenStack, its acquired knowledge has proven valuable when applied to the use of cutting edge technology such as containers.
Financial services case study: How Juju was the key to unlocking a StuckStack European bank

One large European bank had been unable to upgrade its OpenStack for two years. This became a problem in the highly-interconnected ecosystem of the financial services industry, which is arguably the fastest, most eclectic trading environment in the world. The intense competition already demands sub-millisecond response times to trading systems and this sector is subject to massive disruption from aggressive new startups, whose speed and agility is their greatest assets.

The challenge: Only the fittest survive in this environment, and that fitness is dependent on the ability to keep pace with the pack. However, when developers for other trading partners complained that their new services could not be supported by the bank’s outdated versions of OpenStack components, this incompatibility became a serious issue.

The solution: The problem facing the bank was that it hadn’t modelled its relationships. This is where Canonical’s modelling tool Juju came into its own. Manual modelling is time consuming, and consequently systems migration by trial and error becomes a lengthy process. Juju, which automates the process of modelling, enabled thousands of new clouds to be built, tested and tweaked to perfection in a fraction of the time.
Conclusion: unlock the benefits of OpenStack with Canonical

OpenStack is the product of millions of hours of effort from thousands of the finest engineering brains in the industry. It works brilliantly on so many levels because so much enthusiasm and attention is invested in it by developers who are passionate about sharing their expertise. OpenStack is still the best way to run workloads wherever they are most cost effective.

The combined value of all this intellectual capacity can sometimes become locked up in a StuckStack situation. Under these circumstances, the user must decide whether to ditch the machinery for a public cloud which, analysts warn, may involve hidden costs and, worse still, vendor lock in.

The alternative is to unlock the considerable potential of the StuckStack using management tools, such as those outlined in this paper. This would have the added benefit of providing business continuity in areas such as staffing, systems management and security. It also puts the owner of the StuckStack back in control of their company.

By unlocking StuckStack, an organisation can achieve multiple costs saving, reap productivity dividends and beef up security.

Canonical has the keys to achieve this.
About Canonical

At CANONICAL, we are passionate about the potential of open source software to transform business. For over a decade, we have supported the development of Ubuntu and promoted its adoption in the enterprise.

By providing custom engineering, support contracts and training, we help clients in the telecoms and IT services industries to cut costs, improve efficiency and tighten security with Ubuntu and OpenStack.

We work with hardware manufacturers like HP, Dell and Intel, to ensure the software we create can be delivered on the world’s most popular devices. And we contribute thousands of man-hours every year to projects like OpenStack, to ensure that the world’s best open source software continues to fulfil its potential.