The power to do more

Bootstrapping Open Source Clouds

From test bench to hyperscale: Building your own open platforms and infrastructure

A Dell technical white paper

By: Rob Hirschfeld and Greg Althaus

Updated December 2011

Bringing open APIs and best practices to cloud operations.

Tags: #DevOps, #Hyperscale, #OpenStack, #Crowbar

Table of Contents

The power to do more

Executive Summary

Bringing a cloud infrastructure online can be a daunting bootstrapping challenge. Before hanging out a shingle as a private or public cloud service provider, you must select a platform, acquire hardware, configure your network, set up operations services, and integrate it to work well together. That is a lot of moving parts before you have even installed a sellable application.

This white paper walks you through the decision process to get started with an open source cloud infrastructure based on OpenStack™ and Dell™ PowerEdge™ C servers. At the end, you'll be ready to design your own trial system that will serve as the foundation of your hyperscale cloud.

2011 Revision Notes

In the year since the the original publication of this white paper, we worked with many customers building OpenStack clouds. These clouds range in size from small six-node lab systems to larger production deployments. Based on these experiences, we updated this white paper to reflect lessons learned.

OpenStack Taxonomy

In the Diablo design, the Dell OpenStack-Powered Cloud Solution contains the core components of a typical OpenStack solution: Nova, Nova-Dashboard/Horizon Swift, Glance, and Keystone, plus components that that span the entire system such as Crowbar, Chef, Nagios, etc.

The taxonomy presented in Figure 1 reflects both included infrastructure components (shown in lime green) and OpenStack-specific components that are under active development (shown in red) by the community, Dell, and Dell partners. The taxonomy reflects a CloudOps[1] perspective that there are two sides for cloud users: standards-based API (shown in pink) interactions and site-specific infrastructure. The standards-based APIs are the same between all OpenStack deployments, and let customers and vendor ecosystems operate across multiple clouds. The site-specific infrastructure combines open and proprietary software, Dell hardware, and operational process to deliver cloud resources as a service.

Getting Started

If you want a serious leg up toward a working cloud, Dell offers an OpenStack™ - Powered Cloud Solution built using the principles discussed in this white paper.

This solution includes a base hardware specification and Crowbar, a Dell-authored open source software framework, which takes you from unboxing servers to running a usable open source cloud in hours.

Email us at OpenStack@Dell.com *if you are interested in learning more.*

[1] For more information about "CloudOps," please read the CloudOps white paper by Rob Hirschfeld.

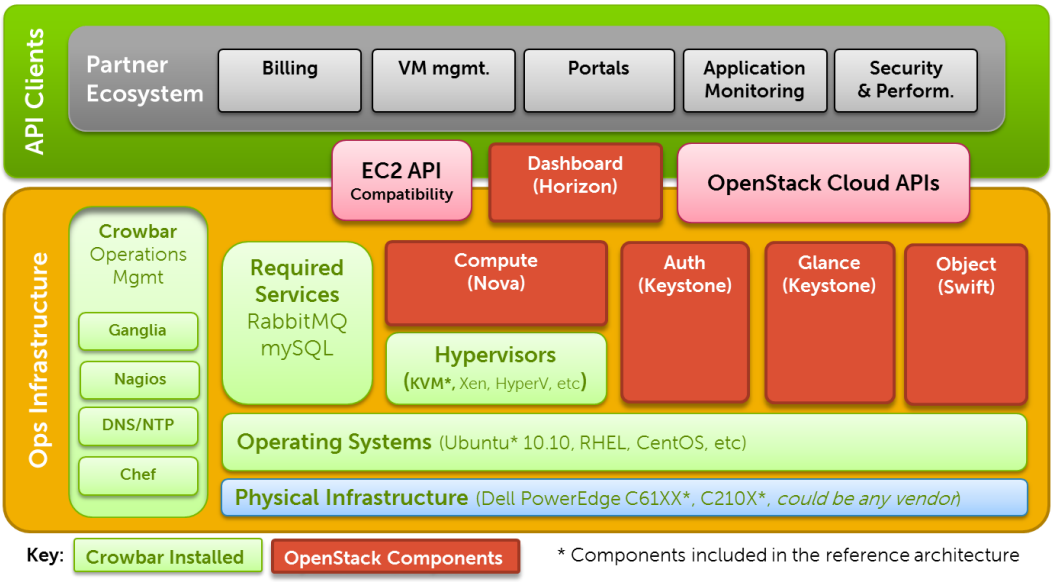

Figure 1. Cloud taxonomy using Dell, OpenStack, and community components

Taxonomy

The implementation choices for each cloud infrastructure are highly specific to the needs and requirements of each site. Many of these choices can be standardized and automated using the tools in this reference architecture (specifically Crowbar) and following the recommended CloudOps processes. Conforming to best practices can help reduce operational risk.

Selecting a Platform

This white paper assumes that you've selected OpenStack as your cloud infrastructure platform. While the concepts would hold for most cloud infrastructures, it's helpful to focus on a single platform for this reference. OpenStack is particularly interesting as an open source cloud platform because it:

- Supports the two top public compute cloud application programming interfaces, or APIs: Amazon® and OpenStack (deployed at Rackspace®, Internap®, DreamHost®, HP®, and others
- Supports the top open source hypervisors: open source KVM and Xen®, proprietary VMware® and HyperV™
- Can run guests using Windows®, Linux®, or other x86-based operating systems
- Deployed at hyperscale (>1000 nodes) at multiple sites such as NASA, Rackspace, and others, and smaller sites globally
- Is truly open and community developed allowing fixes, support, and extend features as needed
- Has a significant, growing, international community adding new features.

OpenStack represents an innovator's paradise: It offers support for existing ecosystems and opportunities to influence future direction, and it provides the foundational components for a cloud service. By building on this foundation, you can create a complete cloud solution.

The power to do more

Dell has been helping groups deploy OpenStack since the Cactus release. Dell's OpenStack getting-started kit specifically targets pilots by reducing setup time and lessening the learning curve to configure a base OpenStack cloud using the Dell Crowbar software framework.

There are five primary components of OpenStack: Authentication (Keystone), Compute (Nova), Object Storage (Swift), Dashboard (Horizon), and an Image Service (Glance). Our focus is on preparing an environment to run OpenStack.

Note: Dell has solutions targeted for customers interested in using OpenStack. Email us at OpenStack@Dell.com if you are interested in learning more.

Fundamental Hyperscale Design Patterns

Hyperscale designs are based on observations for large-scale public clouds. In practice, cloud providers make decisions based on their capabilities, customer targets, risk tolerance, finances, and scale objectives. For example, OpenStack Swift comprehends hyperscale concepts like network zones and just a bunch of disks (JBODs) storage as primary concepts. In updating this white paper, we have attempted to reflect a broader range of scale alternatives.

Fault Zones

Building a hyperscale cloud requires a different mindset – we like to call it "revolutionary" compared to a traditional enterprise virtualized infrastructure. This means driving a degree of simplicity, homogeneity, and density that is beyond most enterprise systems.

The core lesson of these large systems is that redundancy moves from the hardware into the software and applications. In fact, the expectation of failure is built into the system as a key assumption because daily failures are a fact of life when you have thousands of servers.

To achieve scale, individual components intentionally lack network, power, and disk redundancy. Servers are configured with single network paths, single power supplies, and non-RAIDed drives (a.k.a. JBOD). That means that a power distribution unit (PDU), or rack switch failure will take down a handfull of servers. To accommodate this risk, the system is divided into what we call "fault zones." Applications and data are striped across fault zones (similar to data stripping on a RAID) to isolate the impact of multiple component failures.

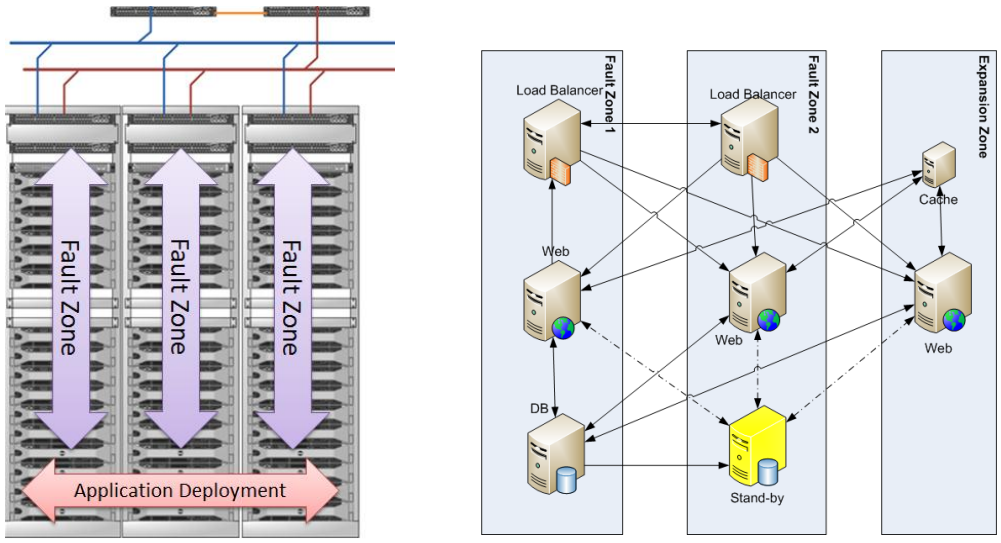

Figure 2. Striping applications and data across fault zones

What is a 'hyperscale cloud'?

Hyperscale systems are designed to operate thousands of servers under a single management infrastructure. The scope of these systems requires a different management paradigm in which hardware faults are common, manual steps are not practical, and small costs add up to large economic impacts.

An example of small costs adding to big impacts: changing a six-drive array from RAID 5 to RAID 10 would reduce total storage by 40 percent. Put another way, you'd have to buy 66 percent more disk (10 instead of 6 drives) for the same total storage!

The power to do more

The benefits of this design approach are significant:

- The ability to choose non-redundant components (disk, server and network) with a lower total cost of ownership (TCO)
- Simpler network routing and configuration
- Simpler physical data center layouts
- Higher density because capacity is not lost to redundant disk, network, and power
- Predictable and streamlined setups and deployment processes.

It is important to point out that core networking is still constructed with redundant and hardware-fault-tolerant paths.

As a consumer of this infrastructure approach, applications must take a fault-zone-tolerant deployment model. See more detail in blogs posts and presentations about application striping using redundant arrays of inexpensive nodes (RAIN).

Flatness at the Edges

"Flatness at the edges" is one of the guiding principles of hyperscale cloud designs. Flatness means that cloud infrastructure avoids creating tiers where possible. For example, having a blade in a frame aggregating networking that is connected to a SAN via a VLAN is a tiered design in which the components are vertically coupled. A single node with local disk connected directly to the switch has all the same components but in a single "flat" layer. Edges are the bottom tier (or "leaves") of the cloud. Being flat creates a lot of edges because most of the components are self-contained. To scale and reduce complexity, clouds must rely on the edges to make independent decisions, such as how to route network traffic, where to replicate data, or when to throttle virtual machines (VMs). We are effectively distributing an intelligence overhead tax on each component of the cloud rather than relying on a "centralized overcloud" to rule them all.

Concepts like "Flatness at the Edges" are based on operating hyperscale clouds. In many cases, hyperscale design requirements are contrary to traditional data center objectives because they have different core assumptions.

The Dell Data Center Solutions (DCS) group has been helping customers build clouds at this scale for years. The innovations from these hyperscale data centers have become more widely available and can now be successfully applied at a moderate scale.

Choosing Hardware

Choosing cloud hardware requires committing to a fault-tolerance strategy that matches your operations model. For hyperscale clouds, our customers had already invested in highly automated DevOps tooling that made their applications very elastic.

This elasticity allowed our customers to limit the impact of hardware failures and changed how they approached infrastructure redundancy. Just as a RAID system focuses on using interchangeable commodity disks, clouds can be built using interchangeable utility servers. The logic is that you will have sufficient scale to create redundancy at the application level.

Note: Smaller clouds cannot achieve redundancy through volume. We have seen that customers typically start with more hardware redundancy while they are building out their initial cloud.

Modularity is a critical value to help reduce complexity. When clouds are measured in the hundreds of nodes, it can be difficult to manage nodes that are linked in groups of six to a dedicated SAN and then connected with eight or more pairs of teamed network interface controllers (NICs) to different cross-connected switches. If just describing the system is difficult, then imagine trying to design, document, and maintain it.

Fundamentally, hyperscale clouds have less shared physical infrastructure by design because physical infrastructure is harder to configure, manage, and troubleshoot. It also has the unfortunate side-effect of causing broader systems outages. While the individual components

may be more likely to fail in this model, the impact of those failures is more isolated, smaller, and much easier to correct quickly.

In our experience, nodes fall into one of four performance categories:

- **Compute** solutions are not as common for virtual machine (VM)-based clouds but typical for some analytics systems (interestingly, many analytics are more disk- and network-bound). In practice, cloud applications are more likely to scale out than up.
- **Storage** solutions should be treated with caution. Use IP-network-based iSCSI storage area network (SAN) or network area storage (NAS) storage because it's much easier to centralize big data than drag all of it to your local nodes. *Note: If you have a solution that needs really big storage and lots of VMs, then it may not be a good cloud application.*
- **Network** solutions may really be compute-heavy systems in disguise. Unless you are packing a lot of RAM and CPU into your systems, it's unlikely that you will hit the wall on networking bandwidth. Remote storage is a primary driver for needing more networking capacity, so you may solve your networking constraints by using more local disk.
- **Balanced** solutions are a good compromise because even the most basic VM placement can distribute VMs to level resource use. This is likely to become even easier when live migration is a standard feature.

Comparing these four categories to available Dell PowerEdge C server models, the balanced-focus server seems to handle the broadest range of applications for compute while the storage node is the best choice for storage. We recommend the balanced node for trial systems because it can be easily repurposed anywhere else as your cloud grows.

Figure 3. Dell PowerEdge C6100 server with four server node sleds

Focus	Dell Model	Rack U	Cores	RAM	Disks/Node	Disk/Core	Net/Core
Compute	PowerEdge C6100 4 sleds (pictured above)	2	32	192	6	3:16	1:2
Balanced	*PowerEdge C6100 2 sleds*	*2*	*16*	*96*	*12*	*3:4*	*1:2*
Storage	PowerEdge C2100	2	8	48	24	3:1	1:2
Network	PowerEdge C2100 + 10-Gb NICs	2	12	48	24	2:1	~ 2:1

"To RAID or not to RAID, that is the question."

Using hardware RAID on compute nodes can provide an additional safety net for customer data. This is important when you do not expect (or force) customers to scale on multiple nodes or use network storage for critical data.

The downside of RAID is that it reduces storage capacity while adding cost and complexity. RAID may also underperform JBOD configurations if VM I/O is not uniform.

Ultimately, your Ops capability and risk tolerance determine if RAID is the right fit.

Assumptions:

- 48 gigabytes (GB) per node (actual RAM is often higher based on expected use of the server).
- 2.5-inch drives boost spindle counts. 3.5-inch drives offer more capacity and less cost but lower input/output operations per second (I/OPS).
- Disk/core assumes unRAIDed drives for comparison. Counts decrease if RAID systems are used.
- Four NICs per node, as per guidance in the "Network Configuration" section of this paper.

The key to selecting hardware is to determine your target ratios. For example, if you are planning compute to have one core per VM (a conservative estimate), then a balanced system would net nearly one spindle per VM. That effectively creates a dedicated I/O channel for each VM and gives plenty of storage. While you may target higher densities, it's useful to understand that the one core class of VMs has nearly dedicated resources. Flatness at the edges encourages this type of isolation at the VM level because it eliminates interdependencies at the maximum possible granularity.

When you look at storage hardware, it can be difficult to find a high enough disk-to-core ratio. For solutions like Swift, you may want to consider the most power-efficient CPUs and largest disks. Object stores are often fronted with a cache so that high-demand files do not hit the actual storage nodes.

So let's look at the concept of a mixed storage and compute system. In that model, the same nodes perform *both* compute and storage functions. For that configuration, the network-optimized node seems to be the best compromise; however, we consistently return to finding that a mixed-use node has too many compromises and ends up being more expensive—10-gigabit (Gb) networking has a hefty premium still—compared to a heterogeneous system. There is one exception: We recommend a mixed-use system for small-scale pilots because it provides more flexibility while learning to use cloud infrastructure.

As with any design, the challenge is to prevent exceptions from forcing suboptimal design changes. For example, the need to host some 100-Gb disk VMs should not force the entire infrastructure into a storage-heavy pattern. It is likely that a better design would be 20-Gb VMs on fast local disk with a single shared iSCSI SAN or NAS target to handle the exceptions as secondary drives. For service providers, these exceptions become premium features.

Hyperscale clouds are fundamentally multitenant. The ability to mix unrelated work together enables large-scale cloud load balancing. The expectation of dynamic demand pairs with the feature of resource elasticity.

Our multitenant assumption creates a requirement paradox: We need both isolation and aggressive intermixing. It should be no surprise that the answer is virtualization of compute, network, and storage.

Network Configuration

It is impossible to overstate the importance of networking for hyperscale clouds, but importance should not translate into complexity. The key to cloud networking is to simplify and flatten. Achieving this design objective requires making choices that are contrary to enterprise network topologies.

A Typical Topology

Best practice for hyperscale clouds calls for four logical primary networks. In practice, these networks are logically isolated on physically shared segments. Best practice is to use teaming to aggregate bandwidth between NICS and improve redundancy.

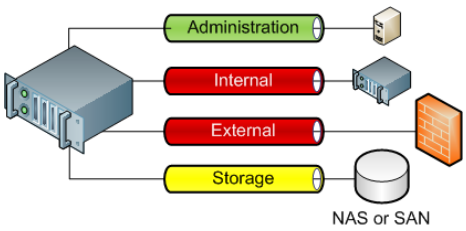

Figure 4. Four logical primary networks

1. The **administration** network connects the cloud infrastructure management to the nodes that run the cloud workloads. This network is restricted and not accessible to VMs. *See the "Operations Infrastructure" section for more information.*
2. The **internal** network provides connectivity between VMs and services (e.g. the object store) within the cloud. This network typically carries the bulk of the cloud traffic, and customers are not charged for bandwidth consumed internally.
3. The **external** network connects VMs to the Internet and is metered so use can be charged against the customer.
4. Use of a **storage** network is recommended when using centralized storage to isolate the impact of large transfers on other networks.

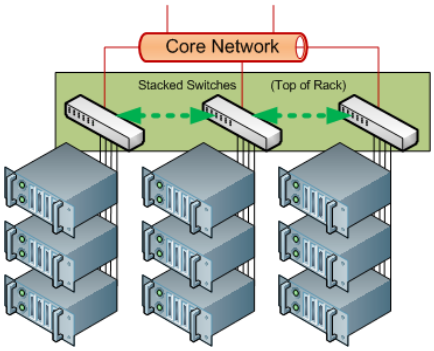

Figure 5. Top of rack switches stacked to create a high speed local fabric avoids sending traffic through the core network switches

There are several reasons for teaming the networks but the primary one is bandwidth aggregation to help maximize the available bandwidth. Teaming also improves fault tolerance: if connectivity is lost to the node, then it cannot generate revenue. Logical segmentation also allows for better IP management. Surprisingly, security is not a motivation for segmentation. In a multi-tenant cloud, we must assume that untrusted users can penetrate to VMs that have access to the internal network; consequently, we must rely on better methods to isolate intruders.

We are seeing 10-Gb networking becoming a default option for OpenStack clouds. The typical configuration is to use two teamed physical 10-Gb NICs. Typically, on-board 1-Gb NICs are also used for the admin. network and Intelligent Platform Management Interface (IPMI).

New Advice about Network Teaming

Our 2010 advice was to avoid complexity in favor of physical isolation.

This is no longer our recommendation.

We have found that teaming can be easily configured in an automated fashion and better accommodate elastic bandwidth allocation on 10-Gb links.

Design Guidelines

Since there is no one-size-fits-all topology, we will outline some basic rules for constructing cloud networks, presented in priority order. Following these rules will help ensure you have a solid cloud connectivity foundation.

Rule 1: Cost matters.

Creating unused capacity wastes money. Idle backup links and under-subscribed bandwidth can more than double costs. Logical segmentation (instead of physical) and teaming NICs allows configuration of fewer network drops and can help economically leverage your network. This is especially true with 10-Gb networks, which carry much higher costs than 1-Gb switches.

A true hyperscale cloud network should be able to use simpler switches, embracing fault zones. In the largest configurations, each server connects to just one switch. This means the need for fewer switches, fewer ports, less sophisticated paths, and shorter wires. Hyperscale clouds choose system-level redundancy plus more (low-cost) resources as a way to improve fault tolerance.

In practice, most clouds cannot tolerate the fault risk of network failures like hyperscale clouds. We see most customers investing in redundant networks with teaming and using logical segmentation (VLANs) to provide isolation. This allows the nodes to leverage all the available network connections elastically.

The power to do more

Rule 2: Keep your network flat.

Four thousand ninety six sounds like a big number. That is the maximum number of VLANs that most networks will support without forcing you to get creative. You will need some VLANs to create logical networks and manage broadcast domains; however, using VLANs to segment tenant traffic will not scale. Our current density recommendation is 36 nodes per rack. If each node supports 32 VMs (four per core) then each rack will sustain 1,152 VMs and require an allocation of nearly 2,500 IPs addresses. Managing tiered networks and VLANs for systems at that density is not practical; consequently, cloud networks tend to be as flat as possible.

Our cloud network reference designs use stacking to create a logical top-of-rack switch: Stacking uses a short-distance switch to switch 14-Gb interconnect networking that effectively merges all the switches. This allows for extremely fast and simple communication between nodes in the rack, and stacked switches can share 10-Gb uplinks to core routers per switch. This way, each switch can still be an isolated fault zone without paying the price of routing all traffic to the core.

Rule 3: Filter at the edge.

Since VLANs do not scale, we need another way to prevent unwanted cross-tenant communication. One solution is to edge filter traffic at the node level. This requires the cloud management system, OpenStack Nova, to set up network access rules for each VM that it deploys. The rules must allow VMs from the same tenant to talk to each other while blocking other traffic. Currently, Linux IPTables is the tool of choice for this filtering, but look for the OpenStack Quantum project which leverages OpenFlow and Open vSwitch.

Rule 4: Design fault zones.

Identify fault zones in your network topology. Remember that fault zones are used to both isolate the impact of failures and simplify design. The lowest paid data center tech and highly automated cloud management system must be able to understand the topology.

Rule 5: Plan for local traffic.

Cloud applications are much more likely to be chatty scale-out architectures than traditional tiered designs. While this delivers reliability by spreading work across fault zones, it creates a lot of internal network traffic. If this internal traffic has to route between switches over the core network, then you can oversubscribe your core bandwidth and impact external communications. Luckily, it is possible to predict internal communication because it is mainly between VMs for each tenant. This can be mitigated with additional outbound links, stacking top-of-rack switches (see Rule 1 above), and clustering a tenant so most of its traffic aggregates into the same core switches.

Rule 6: Offer load balancers.

Our final rule helps enable good architecture hygiene by cloud users: offer load balancers. Making load balancers inexpensive and easy to use encourages customers to scale out their applications. Cloud providers need scaled-out applications to span fault zones and mitigate a hyperscale clouds' higher risk of edge failures. Several public clouds integrate load balancing as a core service, or make pre-configured load balancer VMs easily available. If you are not encouraging customers to scale out their applications, then you should plan to scale out your help desk and operations (Ops) teams.

32nd Rule of Complexity

If you've studied computer science then you know there are algorithms that calculate "complexity." Unfortunately, these have little practical use for data center operators.

Our complexity rule does not require a PhD:

If it takes more than 30 seconds to pick out what would be impacted by a device failure then your design is too complex.

Operations Infrastructure

One of the critical lessons learned about cloud bootstrapping is that Ops capabilities are just as fundamental to success as hardware and software. You will need the same basic core Ops components whether you are planning a 1,000-node public cloud or a six-node lab. These services build upwards in layers from core network services to monitoring, to provisioning and access.

The Administration Server

Before we jump into specific services to deploy, it's important to allocate a small fraction (one for each 100 nodes) of infrastructure as an administration (Admin) service. In all of our deployment scripts, this server is the first one configured and provides the operations services that the rest of the infrastructure relies on. This server is *not* the one running your external APIs or portals; it is strictly for internal infrastructure management. During bootstrapping, it is the image server and deployment manager. Post bootstrapping, it can be your bastion host and monitoring system. Even in our smallest systems, we make sure to dedicate a server for Admin because it makes operating the cloud substantially easier. As we'll explore below, the Admin server is a real workhorse.

Core Services

Core services enable the most basic access and coordination of the infrastructure. These services are essential to cloud operations because the rest of the infrastructure anticipates a data center level of Ops capability. Unlike a single-node targeted SQL server, cloud software expects to operate in an Internet data center with all the services and network connectivity that comes with being "on the net."

Here's the list of core services:

Service	Name	Comments
Address Allocation	Static and DHCP	We've found that DHCP on the Admin network allows for central administration of node addresses and can be used to convey configuration information beyond IP address. We use static addressing on the other segments to avoid collisions with VM-focused network management services.
Domain Names	DNS	Nodes must be are able to resolve names for themselves, other nodes, the admin, and clients. Using a cloud DNS server eliminates external dependencies. Ultimately, clouds generate a lot of DNS activity and need to be able to control names within their domain.
Time Synchronization	NTP	Since the systems are generating certificates for communications, even small time drift can make it difficult to troubleshoot issues.
Network Install	PXE	PXE is required because it's impractical to install bits on large number of servers from media.
Network Access	Bastion Host	*Recommended:* To create (or resolve) network isolation, a bastion host can be configured to limit access to the admin network (production), or create access to the production networks (restricted lab).

We have gone back and forth about using Dynamic Host Configuration Protocol (DHCP) during normal operations. Initially, we were reluctant to introduce yet another dependency to set up and maintain.

Ultimately, we embraced DHCP for the Admin network because it can be used for both delivery boot-up configuration and to sustain our PXE integration. Now that we have the infrastructure in place, we use DHCP and PXE to automate BIOS updates by booting through a patch image.

Service	Name	Comments
Outbound Email	SMTP	*Recommended:* Most cloud components will send email for alerts or account creation. Not being able to send email may cause hangs or errors so it's advisable to plan for routing email.

Provisioning

The most obvious challenge for hyperscale is the degree of repetition required to bring systems online (a.k.a. provision) and then maintain their patch levels. This is especially challenging for dynamic projects like OpenStack where new features or patches may surface at any time. In the Dell cloud development labs, we plan for a weekly rebuild of the entire system.

To keep up with these installs, we invest in learning deployment tools like Puppet and Chef. Our cloud automation leverages a Chef server on the Admin and Chef clients included on the node images. After the operating system has been laid down by PXE on a node, the Chef client will retrieve the node's specific configuration from the server. The configuration scripts (recipes and cookbooks in Chef vernacular) not only install the correct packages, they also lay down the customized configuration and data files needed for that specific node. For example, a Swift data node must be given its correct ring configuration file.

To truly bootstrap a cloud, deployment automation must be understood as an interconnected system. We have been calling this description a "meta configuration." Ops must make informed decisions about which drives belong to each Swift rung and which Nova nodes belong to each scheduler. To help simplify trial systems, Crowbar makes recommendations based on your specific infrastructure. Ultimately, you must take the time to map the dependencies and fault zones of your infrastructure because each cloud is unique.

Monitoring

Once the nodes are provisioned, Ops must keep the system running. With hundreds of nodes and thousands of spindles, failures and performance collisions are normal occurrences. Of course, cloud software takes the crisis out of faults because it is designed to accommodate failures; however, Ops still needs to find and repair the faults. While edge faults should not cause panic, they can degrade available capacity. Good cloud design needs overhead to account for planned (patches) and unplanned (failures) outages.

To accommodate this need, it is essential to set up a health and performance monitoring system for your cloud infrastructure. Cloud monitoring is a well understood and a highly competitive market. If you have an existing Ops infrastructure, then you can leverage your existing systems. For automated OpenStack lab systems, we've integrated open source tools Nagios (health) and Ganglia (performance) into the automated deployment scripts. As we began testing our clouds, we found it very helpful to have these capabilities immediately available.

Want to move faster?

Dell has created the Crowbar software framework to help automate cloud bootstrapping processes and install the OpenStack components described in this white paper.

Crowbar is included as part of the Dell OpenStack-Powered Cloud Solution.

Crowbar is also online as an Apache 2 licensed open source project with a vibrant community.

The power to do more

Beyond Bootstrapping: Laying Down OpenStack

So far, we have focused on preparing the beds for our hyperscale garden: fault zones, servers, networks, and operations infrastructure that all contribute to a rich crop of cloud services.

Once you have completed these activities, you have booted up your cloud and it is time to start installing your cloud software.

This white paper focuses on design considerations before installing your OpenStack cloud software. While covering a complete OpenStack installation is beyond the scope, we want to give you a starting place as you step into the next phase of your cloud deployment.

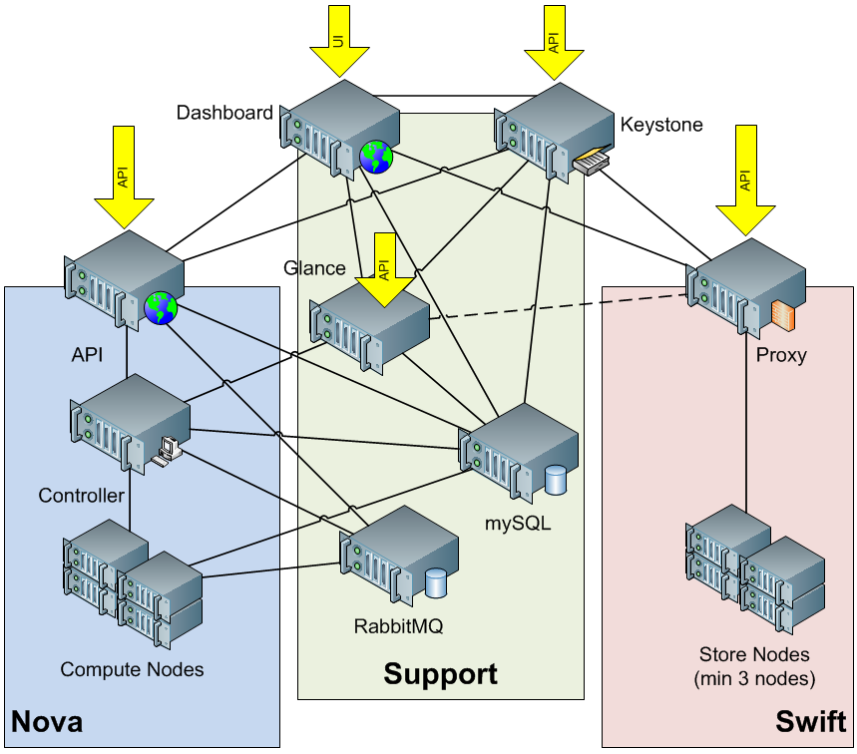

Figure 6: OpenStack architecture

Since the 2010 publication of this white paper, Crowbar has taken on a more general role.

Crowbar is also part of our Apache™ Hadoop™ solutions.

In addition, open cloud applications such as Dreamhost Ceph™ block storage and VMware® Cloud Foundry™ PaaS have created Crowbar modules known as "barclamps."

Barclamps are Crowbar modules that recommend, install and configure software to operate distributed scale-out environments.

The deployment team can choose the software components they would like to install to create the cloud/cluster that's right for their needs.

The following component descriptions are from the http://OpenStack.org site. Extensive documentation for the OpenStack components is available at http://docs.openstack.org/

Function	Code Name	URL
Authentication	Keystone	http://openstack.org/projects/ Identity Service provides unified authentication across OpenStack projects and integrates with existing authentication systems.
Dashboard / Portal	Horizon	http://openstack.org/projects/ OpenStack Dashboard enables administrators and users to access and provision cloud-based resources through a self-service portal.
Object Storage	Swift	http://openstack.org/projects/storage/ OpenStack Object Storage (code-named Swift) is open source software for creating redundant, scalable object storage using clusters of standardized servers to store petabytes of accessible data. It is not a file system, nor real-time data storage system, but rather a long-term storage system for a more permanent type of static data that can be retrieved, leveraged, and then updated if necessary. Primary examples of data that best fit this type of storage model are virtual machine images, photo storage, email storage, and backup archiving. Having no central "brain" or master point of control provides greater scalability, redundancy, and permanence. Objects are written to multiple hardware devices in the data center, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Storage clusters can scale horizontally by adding new nodes. Should a node fail, OpenStack works to replicate its content from other active nodes. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used in lieu of more expensive equipment.
Compute / IaaS	Nova	http://openstack.org/projects/compute/ OpenStack Compute is open source software designed to provision and manage large networks of virtual machines, creating a redundant and scalable cloud computing platform. It gives you the software, control panels, and APIs required to orchestrate a cloud, including running instances, managing networks, and controlling access through users and projects. OpenStack Compute strives to be both hardware and hypervisor agnostic, currently supporting a variety of standard hardware configurations and seven major hypervisors.

Virtual Images	Glance	http://openstack.org/projects/image-service OpenStack Image Service (code-named Glance) provides discovery, registration, and delivery services for virtual disk images. The Image Service API server provides a standard REST interface for querying information about virtual disk images stored in a variety of back-end stores, including OpenStack Object Storage. Clients can register new virtual disk images with the Image Service, query for information on publicly available disk images, and use the Image Service's client library for streaming virtual disk images.

Other Services

OpenStack provides an infrastructure foundation for hyperscale cloud; however, it is not a total solution. Depending on your objectives, additional components will be required to operate your cloud. These components may enable software developers, integrate with internal systems, provide prebuilt templates, and extend operations capabilities.

Some of components to consider are:

- Application support components include data storage services like structured databases (SQL), table storage (NoSQL), queuing services (AMQP), content delivery networks (CDN), and even an application programming platform (PaaS).
- Integrations such as billing, authentication, and VPN tunneling all help customers connect with their internal systems.
- Prebuilt templates and uploading images using open virtualization format (OVF) or similar technologies improves interoperability and allows customers to reuse work from other clouds.
- Operations services that take over operations challenges by offering load balancers, firewalls, security services, backups, access monitoring, or log collection can be a substantial benefit while leveraging economy of scale.

There are an overwhelming number of opportunities to expand beyond the OpenStack foundation. By investing in an open cloud infrastructure platform, you can expand the ecosystem of services and partners. Having a shared platform can reduce duplicated effort and having a large ecosystem encourages innovation and investment to solve difficult problems.

Key Takeaways

Designing an open source hyperscale data center requires thinking about operational problems differently. The large amount of resources not only creates unique complexity management challenges, but also enables solving problems by broadly distributing resources instead of relying on local redundancy.

Logical configuration is just as important as physical layout. Every step away from simplicity will cause exponential growth in complexity at scale. Find ways to automate and monitor.

To help accelerate evaluation of this powerful cloud platform, Dell has invested in creating a more effortless out-of-box experience using our open sourced Crowbar software framework. Combined with Dell's industry-leading PowerEdge C cloud optimized hardware, our cloud installation automation helps ensure that you can confidently build a cloud infrastructure solution to meet your needs over time.

Dell is an active participant in the OpenStack community because OpenStack has the potential to bring open APIs, capable practices to cloud operations, and affordable infrastructure.

To Learn More

For more information on Dell and OpenStack, visit:

www.Dell.com/OpenStack