



The power to do more

Bootstrapping OpenStack Clouds

Platforms and Infrastructure for Hyperscale Environments

A Dell Technical White Paper

Authored by Rob Hirschfeld and Greg Althaus

Contributions from Bret Piatt, Director of Product Management, Rackspace Cloud Builders

Bring open APIs and best practices to cloud operations.



THIS WHITE PAPER IS FOR INFORMATIONAL PURPOSES ONLY, AND MAY CONTAIN TYPOGRAPHICAL ERRORS AND TECHNICAL INACCURACIES. THE CONTENT IS PROVIDED AS IS, WITHOUT EXPRESS OR IMPLIED WARRANTIES OF ANY KIND.

Table of Contents

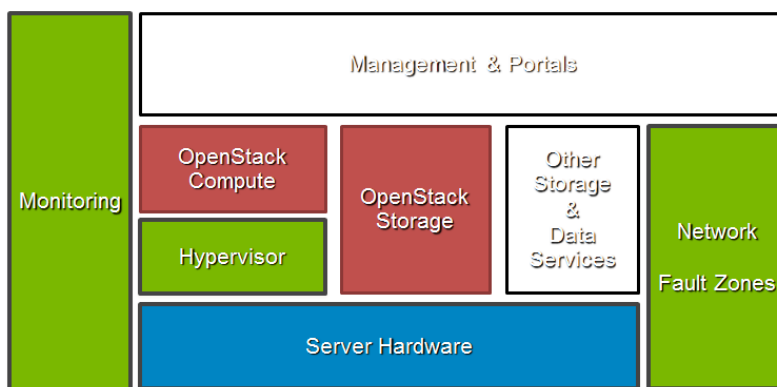
| | |
|--|----|
| Executive Summary..... | 3 |
| Selecting a Platform..... | 3 |
| Fundamental Hyperscale Design Patterns..... | 4 |
| Fault Zones | 4 |
| Flatness at the Edges | 5 |
| Choosing Hardware | 5 |
| Network Configuration..... | 7 |
| Design Guidelines..... | 8 |
| Operations Infrastructure..... | 10 |
| The Administration Server..... | 10 |
| Core Services..... | 10 |
| Provisioning | 12 |
| Monitoring | 12 |
| Beyond Bootstrapping: Laying Down OpenStack..... | 13 |
| Deploying Storage (Swift)..... | 13 |
| Deploying Compute (Nova)..... | 13 |
| Other Services..... | 14 |
| Key Takeaways | 15 |
| To Learn More..... | 15 |



Executive Summary

Bringing a cloud infrastructure online can be a daunting bootstrapping challenge. Before hanging out a shingle as a private or public service provider, you must select a platform, acquire hardware, configure your network, set up operations services, and integrate it all 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 Power Edge C class hardware. The figure below serves as a roadmap for components that we'll cover: red for OpenStack, blue for hardware, green for operations and configuration, and white for topics reserved for future white papers from Dell. At the end, you'll be ready to design your own trial system that will serve as the foundation of your hyperscale cloud.



Selecting a Platform

This white paper assumes that you've selected OpenStack Bexar Release on Ubuntu 10.10 as your infrastructure platform. While the concepts would hold for any hyperscale cloud infrastructure, it's helpful to focus on a single platform for this reference. OpenStack is particularly interesting as an open source cloud because it:

- Supports the two top public compute cloud application programming interfaces, or APIs (Amazon and Rackspace)
- Supports the two top open source hypervisors (KVM and Xen)
- Can run guests using Windows, Linux, or other x86-based operating systems
- Will be deployed at hyperscale (>1000 nodes) at multiple sites (NASA, Rackspace, and others)
- Is truly open and community developed allowing fixes, support, and extend features as needed
- Has a significant, 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. We will discuss added services and extensions at the end of this paper.

Getting Started

If you want a serious leg up toward a working cloud, Dell is building a getting-started kit around the principles discussed in this white paper.

This kit includes a base hardware specification and tools that take you from unboxing servers to running a usable OpenStack cloud in hours.

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



Interested users are now reaching out to Dell for help in test driving OpenStack as an open source cloud. Dell's OpenStack getting-started kit specifically targets trials by reducing setup time and lessening the learning curve to configure a base OpenStack cloud.

There are three primary components of OpenStack: Compute (Nova), Object Storage (Swift), and an Image Service (Glance). Our focus is on preparing an environment to run OpenStack. You will need additional references to learn everything you need to manually complete an OpenStack install.

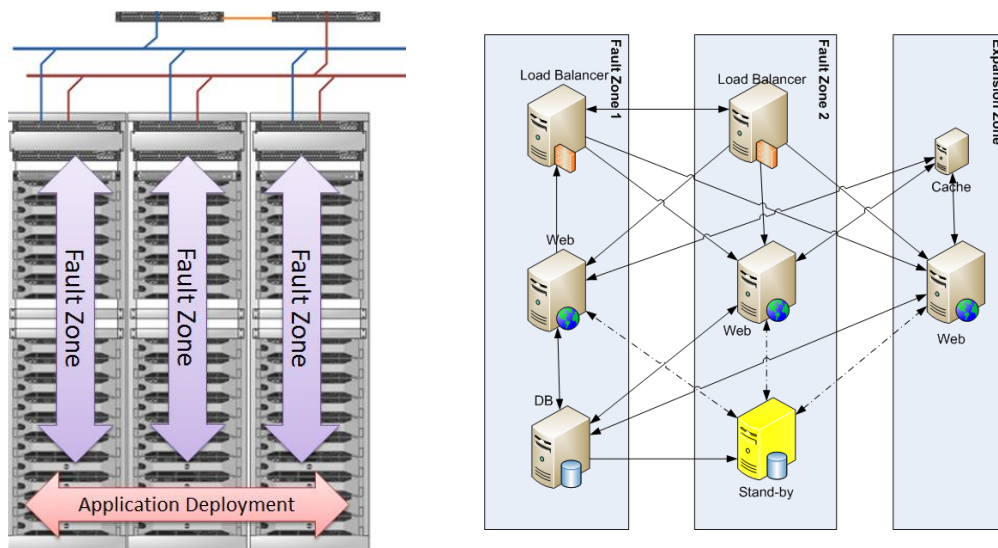
Note: Dell is actively seeking customers interested in conducting a proof-of-concept (PoC) using OpenStack. A PoC engagement with Dell will involve many of the configurations and services discussed in this white paper. You may also wish to take advantage of services available from Rackspace CloudOps. Email us at OpenStack@Dell.com if you are interested in learning more.

Fundamental Hyperscale Design Patterns

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 (aka JBOD, or just a bunch of disks). That means that a power distribution unit (PDU) or rack switch failure will take down a handful 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.

The benefits of this design approach are significant:

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



- 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. We have discussed this in 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.

Note: An anti-example of edge design is using VLANs to segment tenants because VLANs (a limited resource) require configuration at the switching tier to manage traffic generated by an edge component.

Choosing Hardware

Choosing cloud hardware requires committing to a fault-tolerance strategy that matches your operations model. For hyperscale clouds, our customers demand highly modular and dense solutions. Just as a RAID system focuses on using interchangeable commodity disks, clouds are built using interchangeable utility servers. The logic is that you will have sufficient scale to create redundancy and, more importantly, sufficient modularity to grow incrementally.

Modularity is a critical value to help reduce complexity. When clouds are measured in the hundreds of nodes, it is 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 shared 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:

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.

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 begun to trickle down and can now be successfully applied at a moderate



The power to do more

- **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 SAN or NAS storage to address these cases 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 (more about this later). 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 (expected before the OpenStack Cactus release)



"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.

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.

| Focus | Dell Model | Rack U | Cores | RAM | Disks/Node | Disk/Core | Net/Core |
|----------|-----------------------------------|--------|-------|-----|------------|-----------|----------|
| Compute | PEC 6100 4 sleds (pictured above) | 2 | 32 | 192 | 6 | 3:16 | 1:2 |
| Balanced | PEC 6100 2 sleds | 2 | 16 | 96 | 12 | 3:4 | 1:2 |
| Storage | PEC 2100 | 2 | 8 | 48 | 24 | 3:1 | 1:2 |
| Network | PEC 2100 +10 Gb NICs | 2 | 12 | 48 | 24 | 2:1 | ~ 2:1 |

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.

Assumptions:

48 gigabytes (GB) per node (actual RAM can be higher or lower)

- 2.5-inch drives boost spindle counts. 3.5-inch drives offer more capacity and less cost but lower IOPS (input/output operations per second).
- Disk/Core assumes unRAIDed drives for comparison. Counts decrease if RAID systems are used.

Ultimately, your Ops capability and risk



- 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 your 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 gives you the most flexibility while you are learning to use your cloud infrastructure.

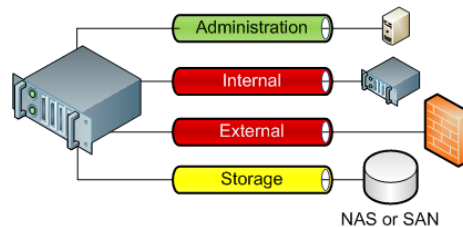
As with any design, the challenge is to prevent exceptions from forcing suboptimal design changes. For example, the need to host some 100-Gb disk VMs should not force the entire infrastructure into a storage-heavy pattern. It is likely a better design to assume 20-Gb VMs on fast local disk and set up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

Network Configuration

It is virtually 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 three logical primary networks with a possible fourth. In practice, these networks are usually mapped directly to physical NICs; however, that mapping is not required.



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.

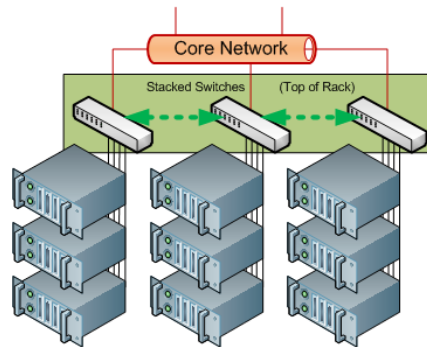
Hyperscale clouds are fundamentally multitenant. The ability to mix unrelated work together enables large-scale cloud load balancing. The expectation of dynamic demand pairs with the feature of resource elasticity.

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,



The power to do more

- Use of a **storage** network is recommended when using centralized storage to isolate the impact of large transfers on other networks. If storage traffic is isolated from the VMs then it may be possible to combine storage with the administration network.



There are several reasons for segmenting the networks but the primary one is bandwidth distribution. We want to ensure that traffic on our money network (external) is not disrupted by activity on the other networks. Segmentation also allows for better IP management. Surprisingly, security is not a motivation for segmentation. In a multitenant 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.

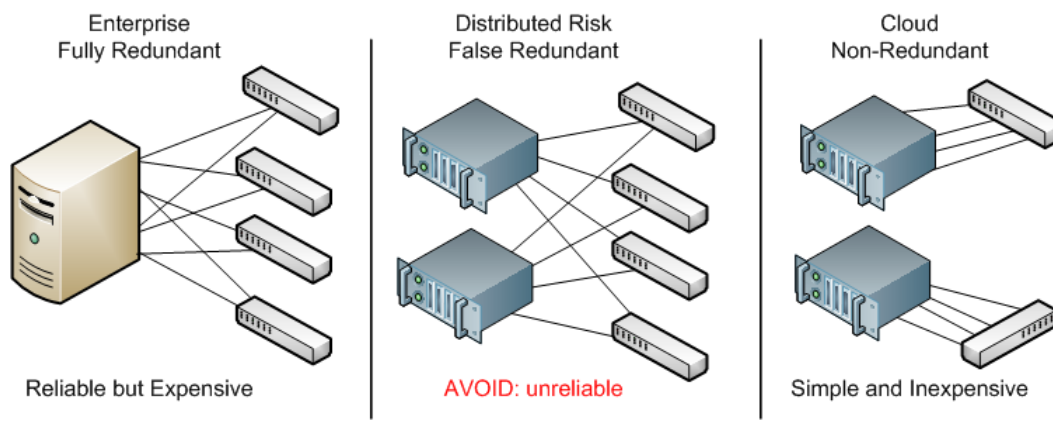
As 10-Gb networking becomes more affordable, we expect to see a trend to map these logical networks into one or two physical 10-Gb NICs. Dropping to a single interface is good design since a single 10-Gb port can carry more than twice the traffic¹ of the four-NIC configuration. In addition, the single high-speed NIC design has more elastic bandwidth: One network can burst to consume up to 80 percent of the capacity and still leave the other networks with 1-Gb of bandwidth. Remember that the Admin network must access the motherboard Intelligent Platform Management Interface (IPMI) and management and likely cannot ride on secondary interfaces.

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 will more than double costs. Adding complexity also costs money. Managing overlapping VLANs, complex routing rules, and sophisticated active-active paths injects the need for manual labor or expensive management tools. In an inelastic enterprise data center, the investment in fully



About False Redundant

While on the surface, creating a one-to-one switch-to-server mapping may seem risky, it is actually more than three times more reliable than spreading the server's network load to four switches.

Unless you have redundancy, adding components to a system will make it less reliable.

¹ While 10 Gb NICs do provide more bandwidth, they have limited packet count that may prevent them from handling the full capacity. For example: VMs sending heavy loads with small packets may saturate a single 10-Gb NIC, where multiple 1-Gb NICs could handle the load.



redundant networks (usually requiring eight or more interfaces connecting to four interleaved switches) makes sense; however, cloud scale makes it economically unfeasible to buy, install, and manage the extra (and more expensive) equipment.

A hyperscale cloud network can use simpler switches because we have embraced fault zones. In our recommended configuration, each server connects to just one switch. This means you need fewer switches, fewer ports, less sophisticated paths, and even shorter wires. More components touching a node makes that node less reliable. The only way to increase reliability is to add expensive and complex redundancy. Hyperscale clouds choose system-level redundancy plus more (low-cost) resources as a way to improve fault tolerance.

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 (4 per core) then each rack will sustain 1,152 VMs and require an allocation of nearly 2,500 IP 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 short distance 14-Gb 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 core.

Rule 3: Filter at the edge.

Since VLANs do not scale, we need another way to prevent unwanted cross-tenant communication. The 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 new approaches using OpenFlow or Open vSwitch.

Rule 4: Design fault zones.

You should be able to easily identify fault zones in your network topology. Remember that fault zones are used to both isolate the impact of failures and simplify your design. Your 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 concern can be mitigated with additional outbound links, stacking top-of-

30-Second 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.



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 the cloud users. 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 cloud’s 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 than you should plan to scale out your help desk and Operations (Ops) team.

Operations Infrastructure

One of our 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 and then to provisioning and access.

The Administration Server

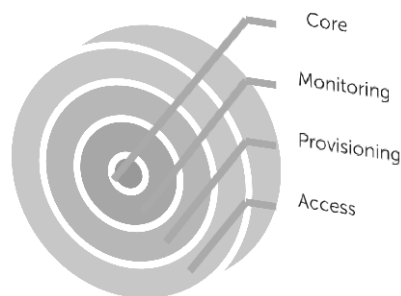
Before we jump into specific services to deploy, it’s important to allocate a small fraction (one for each 100 nodes) of your infrastructure as an Administration (Admin) service. In all of our deployment scripts, this server is the first one configured and provides the operations services that the rest 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 |



We have gone back and forth about using 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 |
|----------------------|--------------|---|
| | | 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 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 Access | Bastion Host | <i>Recommended:</i> 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). |
| Network Install | PXE | <i>Recommended:</i> Beyond lab installs, PXE is required because it's impractical to install bits on large number of servers from media. |



| Service | Name | Comments |
|----------------|------|--|
| Outbound Email | SMTP | <i>Recommended:</i> 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 (aka 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 are 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, our cloud installer 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 do have a financial impact because they 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 both a health and performance monitoring system for your cloud infrastructure. This is a well understood and highly competitive market. If you have an existing Ops infrastructure then you should leverage your existing systems. For our automated OpenStack lab systems, we've integrated open source tools Nagios (health) and Ganglia (performance) into our automated deployment scripts. As we began testing our clouds, we found it very helpful to have these capabilities immediately available.



Beyond Bootstrapping: Laying Down OpenStack

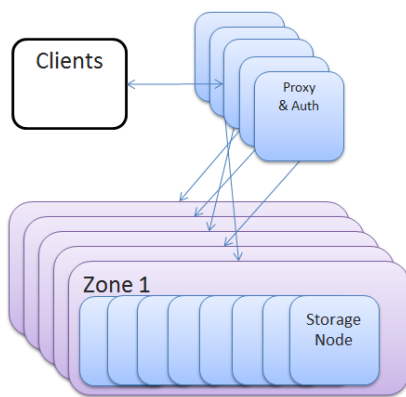
So far, we have focused on preparing the beds for our hyperscale garden: fault zones, servers, networks, and operations infrastructure all contribute to a rich crop of cloud services.

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. 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.

Deploying Object Storage (Swift)

The OpenStack storage component, Swift, is designed to store very large volumes of data at low cost. The simplest way to describe Swift is as a data center-level RAID system where redundancy is moved up to the fault zone level instead of the individual nodes. This means that you must have *at least* three fault zones (“rings” in Swift parlance) in your deployment because Swift distributes copies of your data across fault zones to ensure access.



There are two types of server role for Swift: proxy and storage.

The **proxy** nodes provide the Swift API and authentication services. They are a network-heavy web server.

The **storage** nodes maintain the data copies that are updated and retrieved by the proxy. They are a storage-heavy server.

Networks for Swift are very flat. The proxy must have access to all the storage fault zones because it will retrieve data across all the storage nodes. The storage nodes communicate across fault zones to distribute data

copies.

When planning your Swift deployment, keep in mind that expanding and migrating is relatively easy. When you expand by adding more nodes, Swift will balance in the new capacity. If you need to migrate then just remove the departing nodes and Swift will replicate the “lost” copies to the new nodes.

Swift’s primary function is inexpensive long-term storage; it is not intended as a fast object or block storage device. For these functions, you may need a function-specific storage service, such as SQL, NoSQL, SAN, NAS, or AMQP. It is also possible to front end Swift with a cache such as OpenStack Glance or a content distribution network (CDN).

Deploying Compute (Nova)

OpenStack Compute, Nova, is complex and rapidly evolving. We highly recommend reviewing the latest material available online.

Want to move faster?

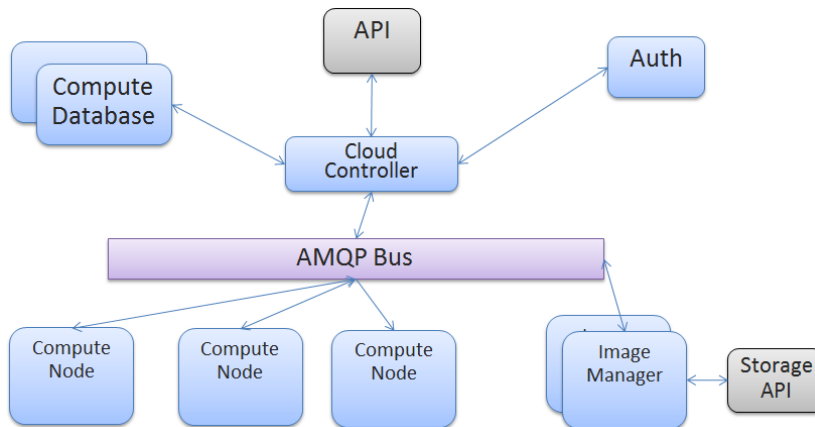
Dell has created tools that automate the cloud bootstrapping processes and install OpenStack components described in this white paper.

We will be making those tools available to customers as part of Dell’s OpenStack getting-started kit.

To influence the design recommendations, users should attend the regular OpenStack Design Summits. Contributors are encouraged to submit “blueprints” for discussion at the summit.



At the most basic level, Nova has a management and orchestration layer that coordinates resources (VMs) for users. This layer supports multiple APIs (Amazon, Rackspace/OpenStack) and hypervisors (currently KVM, XenServer, and Hyper-V, with VMware expected soon). It coordinates with the compute nodes of the resource layer using a message bus (AMQP).



A pluggable scheduler is the logic behind Nova's VM orchestration. Nova will include a component called the Scheduler that decides where to put the VMs in the system. The Scheduler's job is to perform basic load distribution. The open version of the Scheduler will likely use a round robin or most available RAM algorithm. Hosts will extend the Scheduler based on their unique requirements. The current design guidance around the Scheduler is that each Scheduler should manage a resource cluster of approximately 100 nodes (at 36 nodes per rack would map to 108 nodes). Schedulers would coordinate between clusters.

Other Services

OpenStack provides an infrastructure foundation for hyperscale cloud; however, it is not a total solution. Depending on your objective, additional components will be required to operate your cloud. These components may enable software developers, integrate with internal systems, provide prebuilt templates, and extend customers' 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 customers' operations challenges by offering load balancers, firewalls, security services, backups, access monitoring, or log collection can be a substantial benefit for customers while leveraging your economy of scale.

There are an overwhelming number of opportunities to expand beyond the foundation provided by OpenStack. By investing in an open cloud infrastructure platform, you expand the ecosystem



of services and partners. Having a shared platform reduces duplicated effort and having a large ecosystem encourages innovation and investment to solve difficult problems.

Key Takeaways

Designing a hyperscale data center requires thinking about operational problems differently. The large amount of resources 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 ensuring an effortless out-of-box experience. Combined with Dell's industry-leading cloud optimized hardware, our cloud installation automation helps ensure that you can confidently build a sophisticated infrastructure solution.

Dell is an active participant in the OpenStack community because we believe that OpenStack has the potential to bring our customers 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

©2011 Dell Inc. All rights reserved. Trademarks and trade names may be used in this document to refer to either the entities claiming the marks and names or their products. Specifications are correct at date of publication but are subject to availability or change without notice at any time. Dell and its affiliates cannot be responsible for errors or omissions in typography or photography. Dell's Terms and Conditions of Sales and Service apply and are available on request. Dell service offerings do not affect consumer's statutory rights.

Dell, the DELL logo, and the DELL badge, PowerConnect, and PowerVault are trademarks of Dell Inc.