rackspace

SCALING OPENSTACK CLOUDS WITH NOVA CELLS

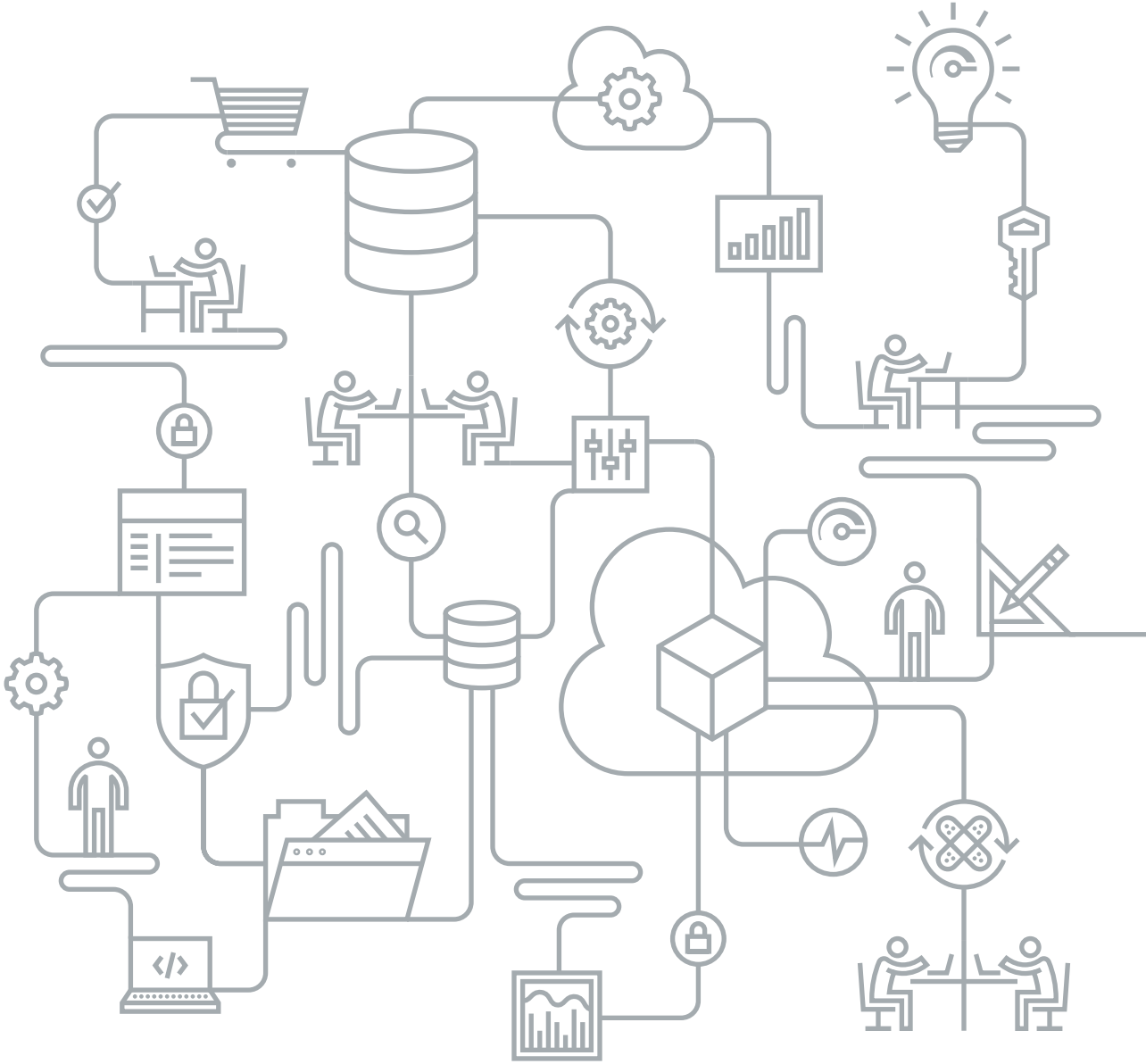


TABLE OF CONTENTS

INTRODUCTION 3
WHAT ARE NOVA CELLS? 3
WHY RACKSPACE CREATED CELLS 3
WHY CHOOSE CELLS? 3
HOW CELLS WORK 3
SIZING OF CELLS 4
ISSUES WITH CELLS 4
CELLS V2 - A COMMON PATH FORWARD 5
CONCLUSION 5

INTRODUCTION

According to the OpenStack Foundation's OpenStack User Survey 2016 , Nova is the most commonly used project in OpenStack deployments. However, many users find Nova to have hidden capabilities that aren't always apparent or easy to understand. In this white paper, we will explore Nova Cells, which offer compelling advantages for streamlining compute capabilities in OpenStack.

WHAT ARE NOVA CELLS?

Simply put, cells are an experimental construct within the OpenStack project that represents a grouping of compute resources. Their functionality enables you to scale an OpenStack compute cloud in a more distributed fashion without complicated technologies like database and message queue clustering.

With cells, you can choose to group compute resources based on characteristics that include hardware specs, network limitations, regional diversity and/or operator preferences. Regardless of why each cell is created within a particular deployment, each cell typically represents a subset of the capacity behind a single API endpoint. They are a step in the scheduling process — cell first, individual host second.

WHY RACKSPACE CREATED CELLS

Not long after we started building our Nova-based Cloud Servers offering, Rackspace created cells. We did this for several reasons. One was convenience — cells allowed us to group capacity in a manner similar to the way we managed hosts in our legacy Cloud Server product built on Slicehost technology. In our legacy product, we called them Huddles. Introducing cells gave us data sharding, service distribution and let us manage our OpenStack-based cloud the same way we had its predecessor. Cells also fit nicely with our supply-chain model. Finally, it gave us a way to group not only similar specs together, but also to keep a cell homogeneous from a vendor/model perspective.

WHY CHOOSE CELLS?

There are a number of reasons that you may choose to use cells, though addressing scaling issues seems to be a common theme. For deep commentary, review this presentation at the Paris summit where operators from CERN, NeCTAR and Rackspace discuss why each chose to use cells (https://www.OpenStack.org/summit/OpenStack-paris-summit-2014/session-videos/presentation/multi-cell-OpenStack-how-to-evolve-your-cloud-to-scale).

Here are a few of the reasons a cloud may use cells:

- Grouping by Flavor/Specs/Feature – As mentioned previously, grouping hardware that supports similar flavors of instance types is a common use case. In Rackspace's case, the vendor and model are matched within a cell as well. This approach helps facilitate live migration and simplifies management since an update for any particular type of server is easy to map to a portion of the fleet. Growth can then be managed on a cell-by-cell basis, as opposed to one massive collection of hypervisors.
- Failure Domains – Within each cell, there are services that are local to each (DB, scheduler, etc.) This gives you some failure tolerance. If a service crashes or has issues, then problems can be isolated to a single cell while other areas can continue to function. This doesn't solve issues at the top level (more on how cells work later), but it does help build in some resiliency by spreading things around.
- Geographic Distribution – Some other organizations utilize cells to group by location. You can make a specific location a cell and manage all of the sites under one set of APIs, etc. In theory, cells can be used to represent subsections of a region or specific datacenter within one geographic location.
- Network Design Constraints – Often, the need to segment a cloud can be driven by the underlying network it resides on. Between VLANs, IP blocks and routing designs, there are a number of reasons an operator may choose to separate one collection of hosts from another. In Rackspace's case, these factors also influence how we size our cells.

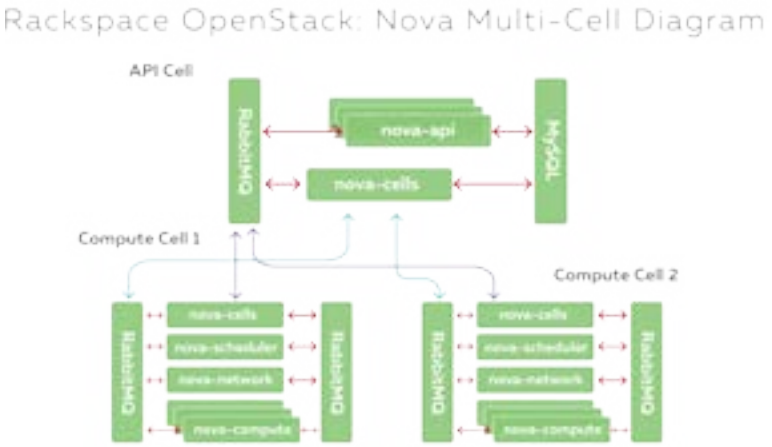
These are just a few reasons you might choose to use cells, but overall, the feature facilitates building and scaling a cloud in a controlled manner.

HOW CELLS WORK

Nova has a pretty straightforward process for scheduling builds:

1. The APIs, in conjunction with the database and scheduler service, route the build on to a message queue.
2. The designated host machine picks the message off the queue and begins the build process.

A more detailed examination can be found here , but below is a high-level architectural diagram to help you understand how cells function:



The changes from a non-cells deployment are largely:

- The scheduler moves from the API level down to the individual cells.
- A new service — cells — is used to help broker messages from the API-level message queues to those within the child cells.
- Databases are introduced in each cell.
- A message queue is introduced in each cell.

So, builds are very similar in a cloud with cells as in one without. An extra step occurs, where the top-level services first determine which cell the build must go to. Once there, the specific host is identified and scheduled to once the build reaches that portion of the infrastructure.

SIZING OF CELLS

What is the best approach for sizing cells? A number of factors contribute, including:

- VM Density – Based on the underlying hardware specs and the options an operator chooses to advertise for the flavors in that cell, the number of hosts included could be a function of the expected VM population. In Rackspace's case, this is one of the factors in our cell sizing, but it is used in conjunction with other network related constraints.
- Cabinet density – In some cases, the number of hosts in a cabinet can govern the size of a cell. This could also be driven by the number of cabs that can be put in a particular part of a datacenter (if the cells represent any physical locality). A collection of N cabinets can easily become the repeatable template used for subsequent deployments. In our case, we factor in the host density per cabinet of our various flavor types. Ultimately, VM density, cabinet density and network constraints lead us to an ideal number of cabinets per cell, per product offering (e.g., General Purpose, High I/O, Compute, Memory, etc.).
- Network Restrictions – A number of network factors may can influence cell sizing. Here are several we think about at Rackspace:
  - Number of Mac/IP addresses at the aggregation router level – All networking devices have a limitation on the number of addresses they can process. When a new flavor is being planned, we look at the following to decide the number of hosts, rounded to the nearest full cabinet and subsequently, the number of cells supported by any one network device:
    - Estimated VM density
    - Specs/options of flavor sizes being offered
    - Number of hosts per cabinet and the capabilities of the upstream router
  - Router ports – Sometimes the number of available locations upstream to plug in a cabinet can define cell sizing. If the average number is five, but one particular cell grew to eight on a particular network device, then another cell on that router may have to be sized smaller to account for this.
  - IP space – This is a little different from the point above. A specific example is the ServiceNet we run at Rackspace. Here, we give every instance a private IP address on a secondary network that allows it to talk to other Rackspace services. When we design new flavors, we like to use this private space as efficiently as possible. So, the number of addresses and average VM density can allow us to back into a number of hosts/cabs as well if it means using an entire /18 (for example) versus three-quarters of a /19.
  - Broadcast domains – Early on, we had some cells that were sized pretty substantially (600+ hosts). They worked fine for the most part. One issue we ran into, however, was that bad actors and/or compromised instances could cause broadcast storms that were difficult to contain because of the total size of the cells and the underlying number of VMs on them — especially with respect to the secondary, internal network. This, along with those issues already mentioned, caused us to rethink the maximum size we would allow a cell to grow.

Based on all of these things, most new cells at Rackspace fall between the 100 and 200 host range. For some of the older flavors, the largest size we chose was around 400. Best practices are always evolving as new hardware options and product offerings are introduced. However, you can take these sizes as a useful rule of thumb.

ISSUES WITH CELLS

Nova Cells is not perfect. Yes, it solved several challenges we had with scaling, but it also introduced a few quirks to keep in mind. Here are a few that Rackspace and others have encountered:

- **Feature gaps –** Nova Cells isn't "fully supported" from a Nova perspective. Some features haven't always worked as expected. Some prime examples are Security Groups (when using Nova-network) and Host Aggregates (both mentioned in the presentation linked earlier). Availability zones are also tricky when using cells. There are others, but these two illustrate that many features are added to Nova that are written and tested solely by developers working in a non-cells environment. There are upstream cells tests, but historically, they have had issues. This has led to the proliferation of features that don't work for cells users. In fact, OpenStack documentation listed the feature as "experimental" for some time and now refers to the original implementation as Cells V1. (http://docs.OpenStack.org/developer/Nova/Cells.html)
- **Project interoperability –** Several other OpenStack projects are not aware of cells. Neutron (network management) is most significant example. To work around this limitation, Rackspace built Quark, our own plug-in (https://github.com/rackerlabs/quark). Upstream support of cells in Neutron is ongoing and likely to not be solved until after Nova releases Cells V2 (more on this later).
- **Double the data, double the fun –** Cells allows you to segment or shard your data. Each collection of hosts has a local database that tracks all the relevant information about its instances. There is a trade-off, however. Because of the simple, two-tier design, all the instance data from the cells is replicated at the API/top layer. This means that an operator is still required to manage a large database of duplicate data in every deployment.
- **Synchronization consternation –** Because there is a regular amount of instance data being sent from the child cell to the API or parent level, there are opportunities for things to get out of sync. For example, an instance might perform a snapshot. The task state for the VM is updated in the databases at both the cell and API layers to reflect this. Occasionally, both do not get the update when the work is complete. When this happens, the database at the cell level might show the task state is NULL, even as the top level shows the task is underway. This can cause issues if automations or other actions require an instance to have an active task state. There are other synchronization issues that may occur between the two layers, and they are the exception, but if your cloud grows large enough, they can become tangible.
- **Message overload –** All this data synchronization also introduces a large amount of messages that have to flow from child cell to parent. Also, features have been introduced to influence the amount of messages as fixes to the synchronization issues above can drastically increase this volume of configurations. This can cause short-term delays in updates from builds and deletes reaching the API. Long term, an operator has to add and size cells service nodes at the API layer as the overall volume increases over time.
- **Set it and forget it –** Cells, as a feature, lacks useful controls. To enable a new cell, a database entry is made to link the child to the parent. After this, though, there is no extremely useful way to control builds. There is the concept of relative weighting of one cell against the others by adjusting DB values, but in practice, we've found it isn't always successful if any one cell has significantly more resources available. There are plenty of times (service failures, IP limitations or other issues) when an operator needs to prevent builds from being scheduled to portions of the cloud. Since there is no "off" switch for cells, one solution could be to just remove the DB entry that linked the child to the parent. The downside is that users on the existing instances can take no action. There are cases (like IP depletion) where builds aren't desired, but the remainder of the functions should still be fully available and working as expected. Ultimately, Rackspace had to create a custom filter for our public cloud that helped the scheduling services know to ignore any particular cell.

You can find a few of the ways Rackspace and other OpenStack cells users get around some of these issues in this presentation from the Vancouver summit: https://www.OpenStack.org/summit/vancouver-2015/summit-videos/presentation/tips-tricks-and-tactics-with-Cells-and-scaling-OpenStack. By no means is this a complete list, but it illustrates some of the challenges and tricks we have learned along the way.

Many members of the Large Deployment Team (https://wiki.OpenStack.org/wiki/Large_Deployment_Team) have worked in past development cycles to introduce patches to fix some of these problems. Recently, however, all Cells V1 work has halted in preparation for the next version.

CELLS V2 – A COMMON PATH FORWARD

At the Paris summit (Kilo), the community agreed that cells features needed to be fully supported, or abandoned. Fortunately, most of the Nova Development Team saw cells as the best way to scale clouds. In conjunction with operators, the blueprints for the beginnings of Cells V2 were fleshed out in some of the design sessions. This work is still ongoing, but looks to solve many of the issues above. For Cells V2 today, the takeaways are:

- Cells will be default. Everyone will get the feature — whether they use it or not. No longer will there be issues with other features not being supported because the functionality will be in the base install, and therefore, gating.
- Cells are already in the Mitaka release. As of the most recent release, the functionality for Cells V2 in a SINGLE cell configuration is present. Multi-cell support could come as early as the Newton release, but most believe it will be at least the Ocata version before multi-cell functionality is completed.
- For those running Cells V1, a migration path won't be available until after the multi-cell features are complete. This is likely to mean that it will be the P release or later before there is an upstream method for current cells users to move. It also implies that an operator thinking of making the switch now should try to wait for the V2 features to land, if possible.
- The top-level DB is being reduced largely to a mapping of instance to cell. This helps reduce the duplication of data and rapid growth of the top-level DB in large environments, but it also means many features have to be added to the APIs to support communication with the child cells.

CONCLUSION

Cells are an excellent example of how Rackspace drives innovation and agility out to the broader OpenStack community. Cells has become an invaluable feature that helps scale the Rackspace Public Cloud. The fact that the OpenStack community is pushing forward with Cells V2 as a standard demonstrates the value of cells, opens the door to continued growth and accelerates our path toward a common feature set across our offerings. We are looking forward to participation in cells development and implementation, and to sharing what we've learned with organizations like yours and the OpenStack community at large.

Learn more about our work to enhance OpenStack clouds and deliver industry-leading reliability, unmatched scalability and innovation-driven agility through a superior approach. Sign up for our free OpenStack strategy session at http://go.rackspace.com/OpenStackExperts.

ABOUT RACKSPACE

Rackspace (NYSE: RAX), the #1 managed cloud company, helps businesses tap the power of cloud computing without the complexity and cost of managing it on their own. Rackspace engineers deliver specialized expertise, easy-to-use tools, and Fanatical Support® for leading technologies developed by AWS, Google, Microsoft, OpenStack, VMware and others. The company serves customers in 120 countries, including more than half of the FORTUNE 100. Rackspace was named a leader in the 2015 Gartner Magic Quadrant for Cloud-Enabled Managed Hosting, and has been honored by Fortune, Forbes, and others as one of the best companies to work for.

Learn more at www.rackspace.com or call us at **1-800-961-2888**.

© 2016 Rackspace US, Inc.

This whitepaper is provided "AS IS" and is a general introduction to the service described. You should not rely solely on this whitepaper to decide whether to purchase the service. Features, benefits and/or pricing presented depend on system configuration and are subject to change without notice. Rackspace disclaims any representation, express or implied warranties, including any implied warranty of merchantability, fitness for a particular purpose, and non-infringement, or other commitment regarding its services except for those expressly stated in a Rackspace services agreement. This document is a general guide and is not legal advice, or an instruction manual. Your implementation of the measures described may not result in your compliance with law or other standard. This document may include examples of solutions that include non-Rackspace products or services. Except as expressly stated in its services agreements, Rackspace does not support, and disclaims all legal responsibility for, third party products and services. Unless otherwise agreed in a Rackspace service agreement, you must work directly with third parties to obtain their products and services and related support under separate legal terms between you and the third party.

Rackspace cannot guarantee the accuracy of any information presented after the date of publication.

Rackspace®, Fanatical Support® and other Rackspace marks are service marks or registered services of Rackspace US, Inc. and are registered in the United States and other countries. Other Rackspace or third party trademarks, service marks, images, products and brands remain the sole property of their respective holders and do not imply endorsement or sponsorship.

NOVEMBER 10, 2016 | PRI-CWP-SCALING_OPENSTACK_CLOUDS_WITH_NOVA_CELLS-4069-v01