



# Resource Visualization

**August 2016**

Author:  
Dinika Saxena

Supervisor(s):  
Ricardo Brito Da Rocha  
Cristovao Cordeiro  
Jan Van Eldik

CERN openlab Summer Student Report 2016

## Acknowledgements

For the two months that I worked on this project, I never, for once felt left in the lurch and I do not know how to thank my supervisors enough for that. I truly feel lucky to have the support of the three coolest supervisors at CERN. I know it in my bones that I could not have completed this project without Ricardo's help. I can't imagine how a person can be as patient as he was with me. It must take a really big heart to shape little minds like me.

I feel like I owe both, a thanks as well as an apology to Cris for never thanking him when I absolutely should have. He helped me get over my self-doubts, give a presentation like I was the Mike Bostock of d3, but the one thing that I want to thank him the most for is for inspiring me in more ways than I can tell.

A big thanks to Jan, my third supervisor. Despite clearly being very busy, I knew that he'd always be there to help me when I needed him.

I'd also like to thank Maria and her friends. The two pieces of advice that she gave me helped me feel so much more confident at work.

No amount of words would suffice to thank Jelena. She has been my rock this summer.

Noor, Patricia, and Kevin, thanks for making work so much more fun.

Thanks to Mehdi for helping me fix a bug in my code at a crucial moment.

I'd fail this section if I don't thank my mom and dad for going beyond their means and making it possible for me to live this experience. I feel so blessed to have you. Thanks to my siblings too, for never making a huge fuss about my constant absence. I'll make it up to you.

I have to thank Scott Murray for writing such a great book on d3. It saved me from burning to ashes with embarrassment.

And, of course, I want to give a big shout-out to CERN Openlab Summer Student Program, its organizers, and every student who was a part of it. This was, without a doubt, the best and the most enriching summer of my life.

## Project Specification

The goal of this project was to build a web based dashboard for visualizing CERN Openstack cloud resources.

I worked on the project for 9 weeks, starting from 27 June, 2016 till 26 August, 2016.

Members of the CERN Openstack team, cloud users, and visitors interested in understanding the architecture of CERN Openstack cloud were targeted as the primary audience for the project.

The technologies and tools deployed to build the dashboard are HTML5, CSS, SVG, JavaScript, D3.JS, and Dimple.

## Abstract

With over 7300 hypervisors in two data centers in the CERN Openstack cloud there is a need to easily visualize the current usage and allocations. My project was to investigate and prototype a service dashboard after collecting the topology information of the CERN cloud. Standard monitoring building blocks which assist in resource planning and visualization of Openstack cloud resources by the cloud administration team and WLCG resource management, were used for this purpose.

## Table of Contents

1.Introduction.....	6
2.Architecture.....	8
2.1Tools Used .....	9
2.2Features Added .....	9
2.3Testing.....	9
3.Result .....	10
4.Future Work .....	11
5.Conclusion .....	11
6.References.....	12

## 1 Introduction

CERN has two data centers: the Meyrin data center, which is the heart of the organization's entire scientific, administrative, and computing infrastructure and its remote extension hosted at the Wigner Research Centre for Physics in Hungary which provides the extra computing power to comply with CERN's needs [i]. In the last decade virtualization started playing a bigger role in the computing infrastructure, with several projects exploiting these technologies. Since 2013 CERN has an OpenStack based cloud moving most of its computing from real hardware to virtual machines which are its new standard deployment objects

The topology of cloud resources at CERN follows hierarchical format. At the highest level there is a central cell, which will be referred to as cell00 in the rest of the report. At the next level there are several child cells which have been categorized on the basis of their computing speed, performance, and storage capacities. Each of these sub cells have hypervisors running on them, and every hypervisor in turn hosts one or several virtual machines.

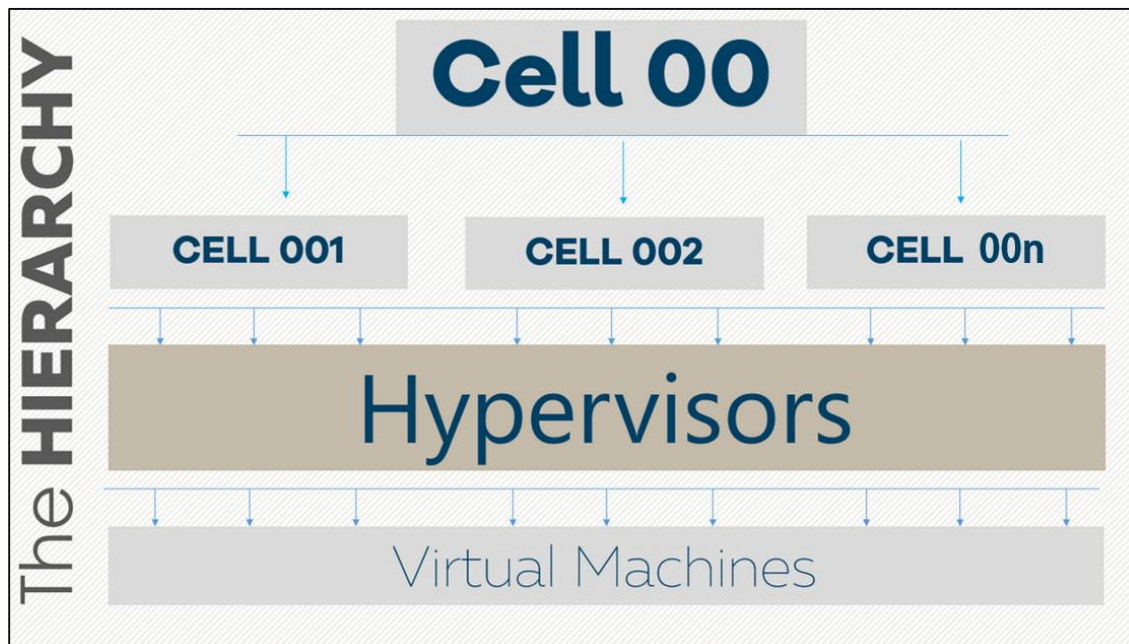


Figure 1: The hierarchical structure of the cloud topology at CERN

There are 44 cells, 7300 hypervisors, and around 25000 Virtual Machines currently running in the CERN's Openstack cloud. Every 10 seconds a virtual machine gets created or deleted [ii].

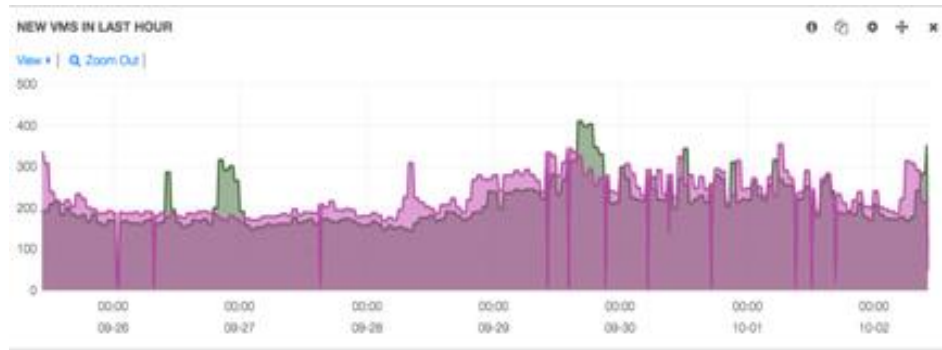


Figure 2: Graph showing creation and deletion of VMs at CERN during an arbitrary time period

All of the data on the topology of the cloud and the details on all the cloud resources are not available in a single place, being spread in different databases. A topology collector was written which dumps it into a single json file. It is very inefficient, if not impossible to dig through an enormous file every time some information is required. Also, a json file is not the best or the most comprehensible representation of data for humans. Therefore, a dashboard that represents the relationship between several cloud components in an easy to understand format was required.

The proposed solution was to design a dashboard that translates the hierarchy in the data stored in the json file (topo.json) into a graph in which each node would represent a cloud component and every edge would show the relationship that the components share with each other.

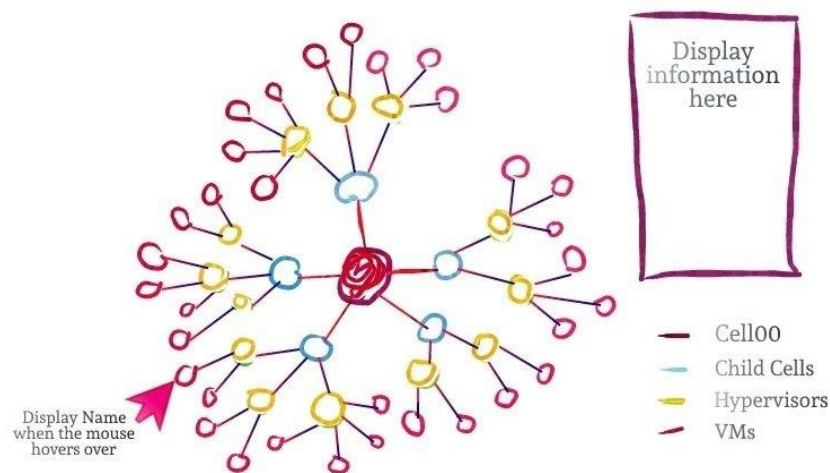


Figure 3: The illustration drawn as a design proposal for the dashboard

## 2 Architecture

In order to create an effective representation of data it is significant to draw a distinction between the exploratory and explanatory data visualization. While working on the exploratory side of visualization it was important to get a sense of what the data is (in this case it was a nested data structure laid out in a json file format which represented the cloud topology at CERN) and what it can convey to its audience (the number, type, status, etc of various cloud components and the relationship they share with each other). This was a process of turning over different rocks to try to find one or two interesting nuggets that the audience would be interested in. As mentioned before, this helped us design an initial, rough idea of what the central graph in the dashboard must look like, represent, and highlight.

In the next phase, the explanatory space of data visualization had to be explored. Attention was paid to allow users to explore the data in interesting ways and look at it from different angles, in an unbiased way. In order to do that four key points were considered:

- i) having a really robust understanding of the context - The primary audience was identified as the members of the IT team and/or CERN employees or visitors who have prior understanding of cloud technologies and are trying to understand or find out details about the topology of cloud at CERN;
- ii) choosing an appropriate type of visual - A collapsible force-layout graph<sup>1</sup> was chosen as the best option for the representation of the data in topo.json. Allowing the users to zoom into faulty virtual machines was also considered. Simple features like clicking on the nodes to display their details and hovering over them to see their name were decided to be added to the dashboard;
- iii) removing the clutter - It is a primary design principle to eliminate things that don't add informative value to the visuals. Doing so decreases cognitive load, does not overwhelm the user, and also causes the data to just stand out more;
- iv) drawing the audience's attention to where we want them to pay it - It was at this point that the strategic decisions about the color, size, and placement of various elements on the page were made. Also, efforts were made to follow principles that make the design comprehensible to a wider audience, for example, mixing and using too much of red and green was avoided to make the visuals easy to read for color-blind people.

---

<sup>1</sup> Force-directed layouts are so-called because they use simulations of physical *forces* to arrange elements on the screen. A physical simulation of charged particles and springs places related characters in closer proximity, while unrelated characters are farther apart. (Murray, 2013)[iii], (Bostock, 2016)[iv]



## 2.1 Tools Used

Since a web based dashboard had to be built HTML and CSS were used for creating, structuring, and styling various elements on the web interface. Javascript was used to add interactivity. The dashboard was made responsive by using Twitter Bootstrap. SVG was used to describe certain set of vector graphics that were to be used on the dashboard. D3.JS, a JavaScript library, which is used to create powerful visualizations with HTML, CSS, and SVG, was used to build and add animations/transitions to the central graph at the heart of the dashboard [iv]. Dimple, which is a powerful and flexible open-source charting API was also used to create mock-ups [v].

## 2.2 Features Added

Following is the list of features added to the dashboard:

- i. the collapsible force layout graph was designed by customizing the open-sourced code available at <https://bl.ocks.org/mbostock/1062288>. Every child node of a particular parent node collapses and merges into its parent when the latter is clicked;
- ii. the users can hover over the nodes to read their name that is displayed in a tool tip, or they can click on them to read their details displayed in the table on the top-right of the dashboard;
- iii. on clicking the 'Zoom into faulty VMs' buttons on the top, a transition that loops endlessly begins focusing on the faulty VMs by zooming into them.

Features yet to be added are:

- i. providing a 'pause' button to stop the zooming-in transition and display the original (zoomed-out) graph;
- ii. Giving users the capability to search for a cell / hypervisor / virtual machine by entering its name in the search box;
- iii. Displaying a list of faulty VMs for the user to scroll through and choose the one that he/she wants to zoom into.

## 2.3 Testing

The code has been tested to run smoothly on Google Chrome (Version 52.0.2743.116 m) Microsoft Edge (Version 25.10586.0.0), except that on the latter, the changes made to the scroll don't apply. Adding appropriate browser extensions in the CSS should fix the issue. The code is yet to be tested on different versions of Safari, Opera, and Firefox, and older versions of Chrome and Edge. Also, it has be run and tested on various screen sizes.

### 3 Result

The screen-shots below display the dashboard that was built:

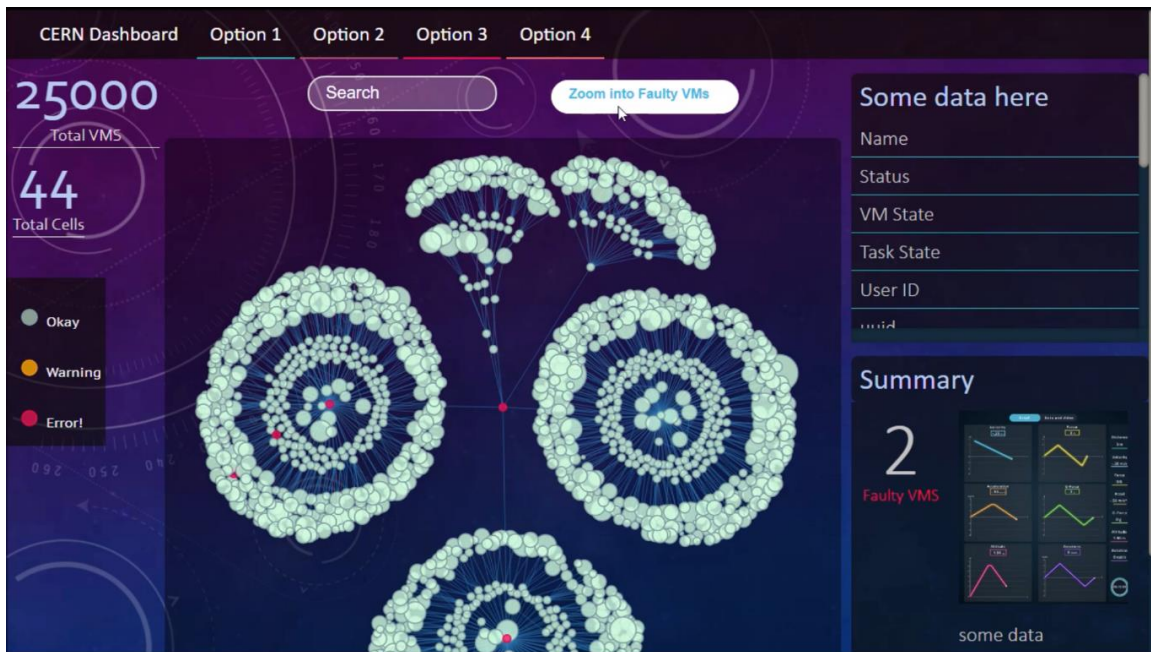


Figure 4: The dashboard displaying the collapsible force layout graph which shows the topology of CERN cloud.

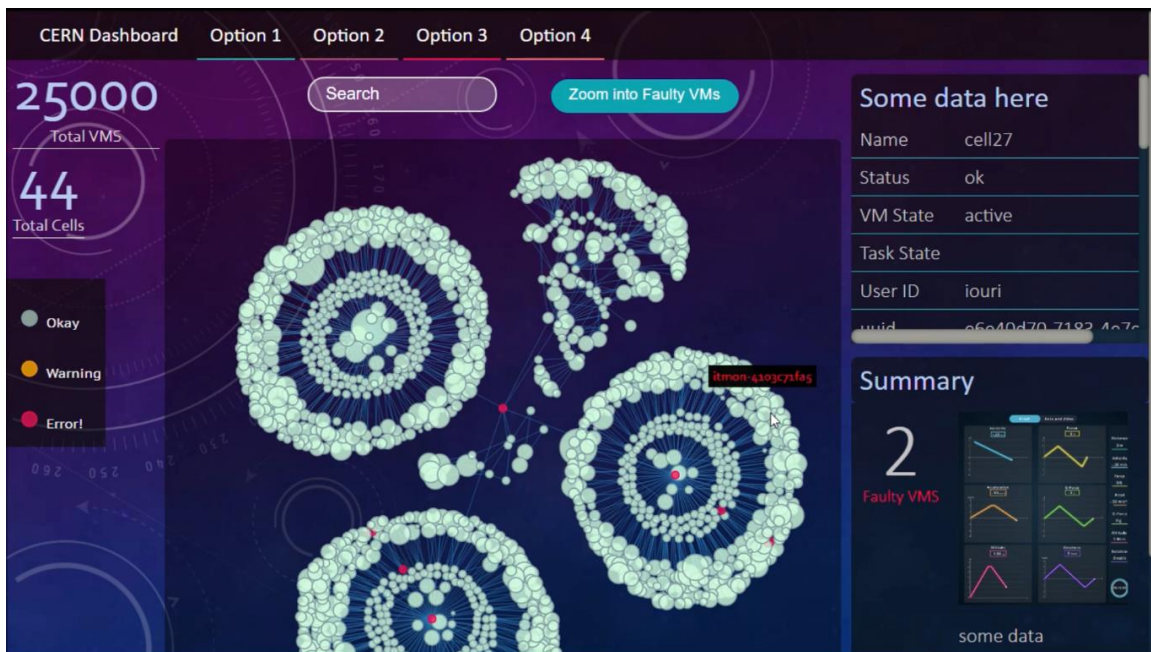


Figure 5: Screenshot of the tooltip displaying the name of the VM that the mouse hovers over and the table on the right displaying other details of the VM on mouse click

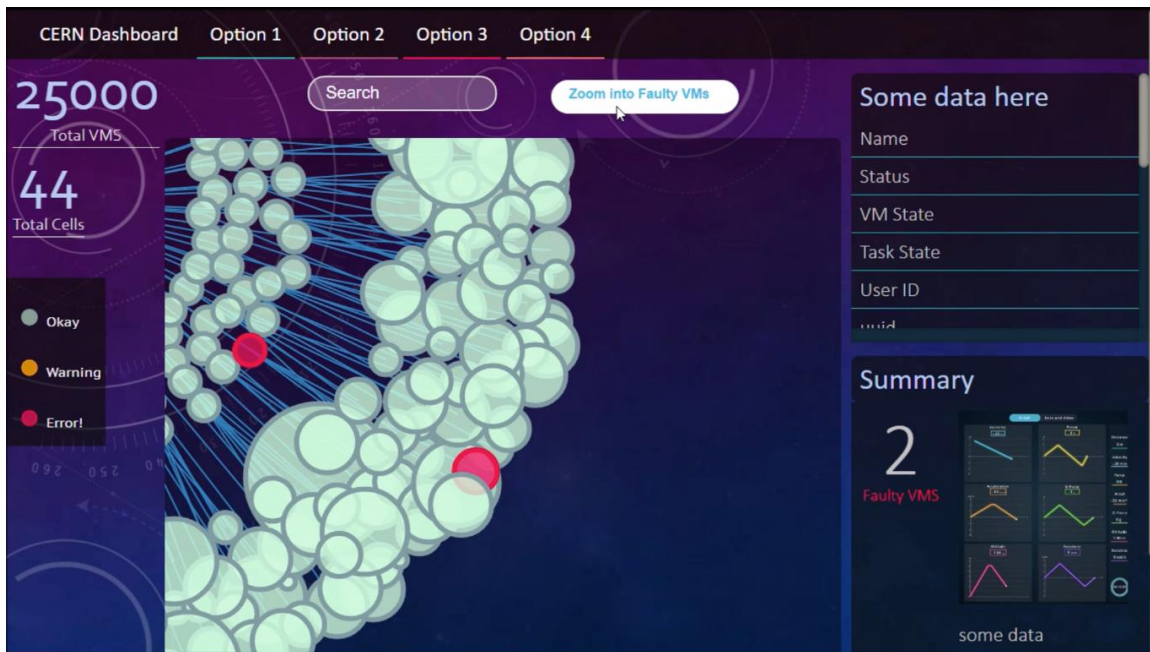


Figure 6: Screenshot of the dashboard zooming into a faulty VM

## 4 Future Work

The scope for future works and updates to the dashboard is broad. Apart from the possibility of adding more features and visualizations to it, it would be really great (and practical) to have the changes in the topology of the cloud or in the operation of cloud components to be dynamically uploaded to the dashboard, i.e. without requiring the user to hit the reload button. These changes must also be reflected in topo.json. Additionally, popular JavaScript frameworks can be used to convert the dashboard into a single-page application that work seamlessly in the browser without ever reloading the page [vi].

## 5 Conclusion

The prototype of the dashboard helps us to conclude that there is indeed a very practical representation of the hierarchy of cloud resources at CERN which can make their monitoring and management very functional and efficient. Now that we have a working prototype we can dive deeper into the visualizations and smoothen them, add more functionality and features to them, and produce a more usable web interface.

## 6 References

- i. CERN, *CERN Cloud Infrastructure Report*, Wiebalck Arne , HEPiX Autumn Meeting BNL, Upton, N.Y., U.S.: N.p., 2015. Web. 12 Oct. 2015. (Available at: [https://indico.cern.ch/event/384358/contributions/909256/attachments/1170138/1689503/CERN\\_Cloud\\_Report\\_-\\_HEPiX14OCT2015.pdf](https://indico.cern.ch/event/384358/contributions/909256/attachments/1170138/1689503/CERN_Cloud_Report_-_HEPiX14OCT2015.pdf))
- ii. "Data Centre | IT Department". *Information-technology.web.cern.ch*. N.p., 2016. Web. 25 Aug. 2016.
- iii. Murray, Scott. *Interactive Data Visualization For The Web*. Sebastopol, CA: O'Reilly Media, 2013. Inc.
- iv. Bostock, Mike. "D3.js - Data-Driven Documents". *D3js.org*. N.p., 2016. Web. 25 Aug. 2016.
- v. "Dimple - A Simple Charting API For D3 Data Visualizations". *Dimplejs.org*. N.p., 2016. Web. 25 Aug. 2016.
- vi. "Front End Frameworks | Udacity". *Udacity.com*. N.p., 2016. Web. 25 Aug. 2016.