

[Home](#) [Archives](#) [Profile](#) [Subscribe](#)[Main](#) | [Zero Touch Infrastructure Provisioning \(ZTIP\)](#) »

11/30/2017

The IaaS Missing Link

The beginning

Almost exactly two years ago, a small group of us (including Alan Rajapa and Massarra Tannous) participated in a “Hack Attack” that was being run by one of the Storage Platform teams within EMC.

Our concept was simple; we were going to **completely eliminate the manual storage provisioning steps required to expose EMC Array based storage to an OpenStack instance.**

If you know anything about OpenStack and Cinder, you understand why we’d want to do this. Back then, manually adding a storage array “backend” to OpenStack required about 17 steps per compute (server) instance. Not only that, the amount of information you needed to collect from each backend (Storage array) required you to know EXACTLY what you were doing.

Very few people did.

As a result, we were hearing about failed OpenStack implementations and frustrated customers asking “why does this need to be so HARD??”.

We felt that it didn’t need to be.

In fact, we were pretty sure that we could accomplish our goal using open source tools and very little additional development would be required.

In the end, I think we were wildly successful with our concept of “Zero Touch Storage Provisioning” (ZTSP) and actually won third place in the event.

Now, you’re probably wondering why you’ve never heard of this project (at least on this blog) and I’ll just say that the ZTSP backstory along with the TDZ backstory will have to wait for another day.

In any case, these experiences HAVE led to a ton of insights that I’ll be talking about over the next few blog posts. One of the biggest discoveries (for me) was the existence of an IaaS “Missing Link”.

[Erik Smith](#)

[ARCHIVES](#)

[September 2022](#)[August 2020](#)[April 2019](#)[December 2017](#)[November 2017](#)[July 2017](#)[April 2017](#)[September 2016](#)[July 2016](#)[February 2016](#)

SEARCH

[CATEGORIES](#)

[Subscribe to this blog's feed](#)

DISCLAIMER

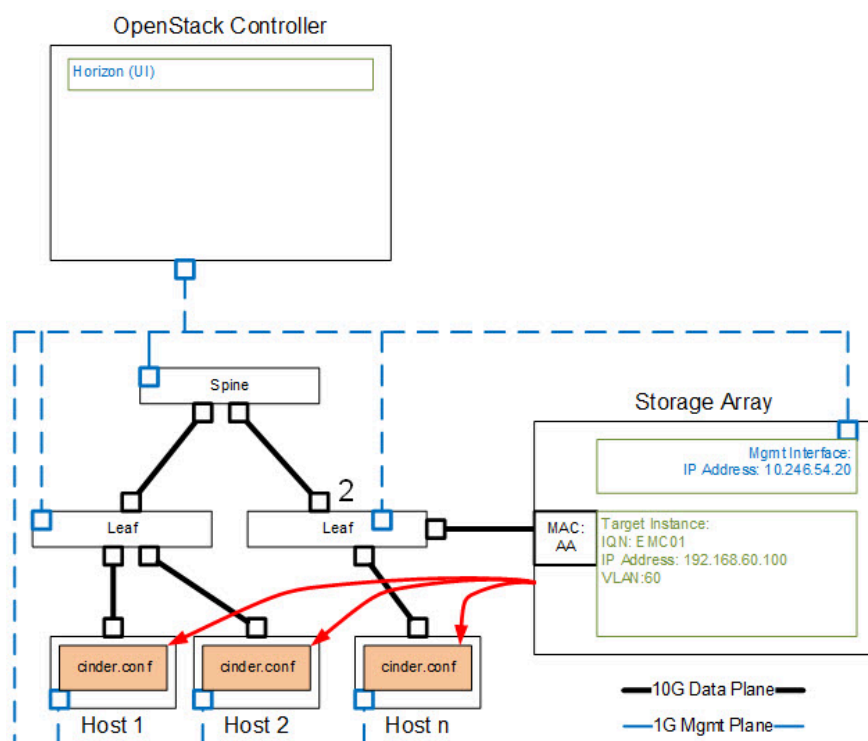
This is not an official EMC blog. The opinions expressed here are my personal opinions. Content published here is not read or approved in advance by EMC and does not necessarily reflect the views and opinions of EMC nor does it constitute any official communication of EMC.

But before we go there, I'll describe Zero Touch Storage Provisioning (ZTSP) and then I'll use this concept to help describe the missing link.

Zero Touch Storage Provisioning (ZTSP)

The following diagram shows the typical configuration we were testing with at that time. It consisted of:

1. An OpenStack controller that was running a multitude of OpenStack services including the Horizon UI. Horizon was an apache based GUI that supports end users allocating storage to VMs (amongst other things).
2. A storage array that supports [iSCSI](#)
3. Three compute nodes that were each running the [OpenStack Cinder](#) Service
4. A management network (blue dashed lines)
5. A data plane network (black solid lines)



In order to be able to provision storage (on an EMC based storage array) and expose it to a VM (running on one of the Hosts running Cinder), you needed to follow a fairly complicated process that, at least at one point, required 17 configuration steps. These steps included creating a storage group on the array, collecting information about the array and the storage group from the array, masking the host to see the storage group on the array and then once this work had been done you had to go to each host and configure the cinder.conf file to create a storage “backend”. This was a tedious, time consuming and error prone process that seemed to be getting in the way of customer adoption.

Also, about this time, VMware VSAN was really starting to get some attention. In fact, I had set it up in the lab and was completely blown away by the experience. It was just SO simple to use and represented a quantum leap forward in simplicity for storage consumers. I found it such a great experience that I set out to try and reproduce the VSAN storage consumption experience

with traditional storage arrays and hence the idea of Zero Touch Storage Provisioning was born. It was shortly after this that we received the invite to participate in the Hack Attack and so after discussing it with the team we decided to give it a shot.

We quickly realized that the storage provisioning problem we were trying to solve could be broken down into three parts:

1. Discovery - Determine when a new storage array was attached to the data plane network.
2. Configuration - Determine what information needed to be put into the cinder.conf file.
3. Delivery:
 - a. Determine how to get the information into the cinder.conf file on the hosts and then refresh the cinder service.
 - b. Determine how to refresh the Horizon service on the control node in order for it to pick up the new storage backend and display it to the end user.

We also quickly realized that Puppet was a good choice to handle part 3 (Delivery). We would just create a “Gold copy” of the cinder.conf on the Puppet master and then allow the individual hosts to detect the change, download the new “Gold copy” of the cinder.conf and then refresh the cinder service. With the exception of actually populating the contents of the “Gold copy” of the cinder.conf, all of these steps were almost trivial to accomplish with puppet.

Part 1 (Discovery) was also very simple to solve, at least on paper. Our plan was to use the [Link Layer Discovery Protocol](#) (LLDP) and to have the data plane interfaces on the array populate the LLDP packets being transmitted with just enough information to allow something (e.g., our controller) to detect that an array had been connected to the network. It turns out that getting the information to the switch via LLDP was pretty straight forward but in order to detect that a new array had been added, we had to choose to either poll for LLDP information or use an asynchronous notification mechanism. We chose the latter and used SNMP for this purpose.

Part 2 (Configuration) required a bit more work but again, nothing too complicated. Our plan was to use the information provided in the LLDP packet to login to the array’s management interface, configure the required storage pools, update the masking and then collect the appropriate information such that we could update the “Gold copy” of the cinder.conf.

Bringing it all together

With the above plan in place, here’s what we actually ended up doing during the Hack Attack.

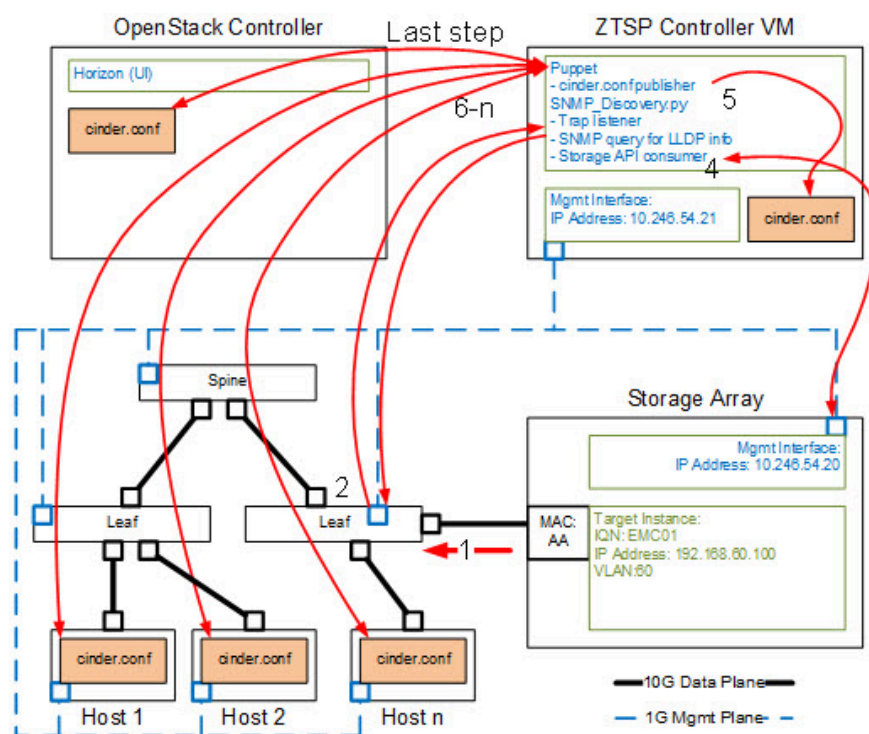
Pre-requisites

1. We updated the array firmware so that it would transmit LLDP packets that contained two pieces of information:
 - a. The [MAC Address](#) of the 10G interface
 - b. The IP Address of the array’s management interface
2. We created a ZTSP controller that contained:

- a. Puppet
 - b. A cinder.conf publisher
 - c. An SNMP_Discovery.py that we ran as a daemon and contained:
 - a. An SNMP Trap listener
 - b. An SNMP query function (to retrieve LLDP information from the switches)
 - d. A Storage API consumer
 - e. Credentials for the storage array. They were in plain text but we could have used something like vault for this (and have since).
3. Each switch needed to have the ZTSP Controller's IP Address added as a trap recipient
 4. Puppet needed to be installed and configured.

The process

The configuration process we created is shown below and explained in more detail in the text that follows.



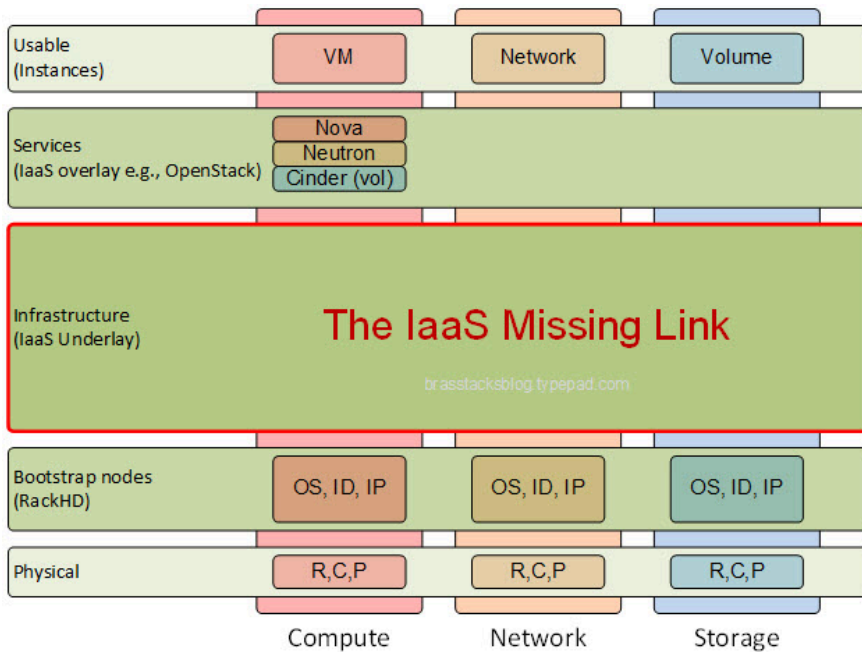
1. The 10G interface on the array was attached to a port on the leaf switch. As soon as the link was up the interface started to transmit LLDP packets that contained the MAC Address of the 10G data plane interface (that was just attached) and the IP Address of the Array's management interface.
2. As soon as the port on the switch changed state, an SNMP trap was sent from the switch to the ZTSP controller and was handled by our trap listener.
3. The ZTSP controller would retrieve the LLDP information from the switch.

4. Using IP Address from the LLDP packet that was retrieved in step 3 and the storage management interface credentials that had been stored in the controller ahead of time, the ZTSP controller would login, create a storage group, update the masking and then collect the information we needed to update the “Gold copy” of the cinder.conf.
5. We created/updated the “Gold copy” of the cinder.conf
6. Puppet took care of the delivery of the cinder.conf to each node as well as the need to refresh the cinder service on each compute node as well as the horizon UI on the control node.

With this functionality in place, all of the OpenStack specific configuration steps were eliminated and we declared success!

The IaaS missing link

Although our Proof of Concept was a huge success and we were able to demonstrate exactly what we set out to do, we noticed that “completely automating the consumption of storage” and “completely automating the configuration of Cinder specific configuration parameters” were totally different things. In fact, the work we had done during our PoC represented the latter and was really only the “easy” stuff. This work is shown in the “Services” layer below. The really hard stuff, like [configuring an IP Network](#) or configuring end-to-end connectivity for iSCSI was much harder and fell into an area that I refer to as the “IaaS missing link.”



I'll provide much more information about the IaaS missing link in my next blog post.

Thanks for reading!

Posted at 02:56 PM | [Permalink](#)

Tags: #IaaS

Reblog (0)

Comments

 You can follow this conversation by subscribing to the [comment feed](#) for this post.

The comments to this entry are closed.

[Brass Tacks](#) [Powered by TypePad](#)