**Splunk LAB In a Box (draft)**

9/19/2016

**Introduction:**

Have you ever wanted to create a multi-site cluster environment in your lab but you don’t have the resources for it? Have you ever wondered how does bucket replication translates on the file system level? Have you ever wanted to create portable Splunk classroom but it is cost prohibitive? How about changing critical configuration without worrying about messing up your production environment?

If you are like me; you must have dealt with similar challenges.

Like most people, you probably attempted to solve the problem by either throwing more hardware at it, or by using some sort of VM technology that does not scale well without additional resources and cost. Well, I have a solution for you! But before that I would like to welcome you to the world of docker, the game changer that brought micro services to reality. Imagine that with a click of a button you are able to create 3-sites cluster, each location running 3-SHs and 10-IDXs. Or maybe just instantly *create a portable lab environment* for testing or training purposes.

You may have heard of docker, or you may even experiment with it trying to figure out how can I use it to help my Splunk needs. But learning docker technology by itself is not helpful unless its used in the contest of specific app like Splunk. To help my customers (and myself) I have created a wrapper bash script (around 1200 lines) to manage Splunk instances builds. The script will allow you to *create a pre-configured large number of Splunk infrastructure components without having to learn a single docker command and with minimal resources requirements (CPU, memory, HD space)*.

In my small test environment I was able to quickly bring upward of 40+ Splunk docker containers for a classroom lab using low powered Intel NUC device (i3 16GB ram, 128G SSD). What’s impressive about docker is that the footprint on my docker-host was extremely small compared to similar buld using a VM solution. I need to emphasize that I have not tested my script under heavy load (either user interaction or data ingestion), however, I believe is just a matter of sizing the hardware appropriately, but at much lower footprint than the alternative methods such as duplicating your production hardware or using virtualization technology. And that is promise of micro services!

While this article is not intended to teach how to use docker, it’s more about how can you leverage this open source tool to make your life easier. I do, however, strongly encourage you to learn how to navigate around docker. There are massive amount of documentations and YouTube videos for docker on the Internet. Instead I will spend the time talking about the menu system I wrote to enable you to build Splunk infrastructure elements on the fly. Before we start you may want familiarize yourself with few docker commands like:

docker **run** : to create a container

docker **stop** : to stop a container

docker **rm** : to remove a container

docker **exec** : to execute remote commands on a container

docker **info** : to get more info on running containers

**Script feature list:**

-Extensive Error checking when configuring the containers

-Adaptive load control (throttling if exceeds 4xcores) during cluster build

-Built-in dynamic host names and IPs allocation

-Automatically create & configure large number of Splunk hosts very fast

-Different levels of logging (show docker commands executed)

-Complete multi and single site cluster builds including CM and DEP servers

-Manual and auto (standard configurations)

-Modular design that can easily be converted to a higher-level language like python

-Custom login screen (helpful for lab & Search Parties scenarios)

-Low resources requirements

-Eliminate the need to learn docker (but you should)

**Where to get it?**

I have posted the source code on github <https://github.com/mhassan2/splunk-n-box>

Please download and install in your lab. The script was tested on Ubuntu 16.04. I am guessing running on equivalent Linux distribution will not be a problem. I have not tested the code on MAC OSX or Windows, nor I recommend doing so at this point in time (until more native docker solution is developed for these two platforms). Windows and OSX do not support c-blocks natively, therefore there is an additional layer of virtualization (Oracle VBOX to be specific) required, which really defeat the purpose of micro servers concept. Additionally the scripts heavily utilizes NATing to allow Splunk containers to be visible to the outside world, which means you probably have to NAT 2-3 times to achieve the same goal using non-Linux host OS.

**How does it work?**

Once you have your Ubuntu up and running please follow the instructions for installing docker <https://docs.docker.com/engine/installation/linux/ubuntulinux/>

Please be aware that Ubunto 14.04 did not work very well for me. There is a bug around mounting docker volumes. Your mileage may vary if you decide to use CentOS or equivalent Linux distribution.

When the scripts runs for the first time it checks to see if you have any IP aliases available (the range specified in the script). If not; then it will configure IP aliases 192.168.1.100-250. The aliased IPs will be automatically mapped, at container creation time, to the internal docker IP space (172.17.0.0/24). You should be able to point your browser to any NATed IP on port 8000 and that will get you directly to the container. During my research I haven’t seen many people using that technique and they opt for changing the ports. My approach is to keep the standard Splunk ports (8000, 8089, 9997,etc) and use iptable NATs to make the containers visible to the outside world. This will save you a lot of headache when dealing with creating large number of Splunk containers (aka hosts).

**Splunk image:**

I used outcoldman image as basis of this work. But for some reason it was pulled out of docker registry website. You still can get it here: <https://github.com/outcoldman/docker-splunk>

Here is his instruction on downloading the image (remember to include the trailing dot):

git clone https://github.com/outcoldman/docker-splunk.git

cd docker-splunk/splunk

docker build --tag="$USER/splunk" .

If that doesn’t work for you then try another “splunk” image on the registry. Or you can make your own-stripped down Splunk image. If there is enough interested I will probably post my own but at this point what’s out there should work.

**Configuration and setup:**

You may need to adjust the script to match your network space. Or you can simply use the defaults if your routed network is 192.168.1.0/24. In my lab, the docker-host is 192.168.1.100 it’s also where I ran my dnsmasq (DNS caching server). If you prefer not to use dnsmasq; then just use actual container IPs in your browser. The first container you create will start at 192.168.1.130 and last one will end at 192.168.1.250. If you wish to setup your docker-host with permanent IP aliases see this link <http://askubuntu.com/questions/585468/how-do-i-add-an-additional-ip-address-to-an-interface-in-ubuntu-14>

ETH="eth0"

BASEIP="192.168.1" #must be routed network. We are using class C here!

BASEOCTET4="129"

START\_OCTET4="130"

END\_OCTET4="250"

DNSSERVER="192.168.1.100" #if running dnsmasq on the host machine 192.168.1.100

#SPLUNK\_IMAGE="outcoldman/splunk:6.4.2" #taken offline by outcoldman

SPLUNK\_IMAGE="root/splunk"

BASEHOSTNAME="IDX"

**Container host names rules:**

When you get comfortable navigating around the options you will soon discover it so easy to pop up hosts all the time. Inconsistent hostnames will lead to confusion. That actually happened to me! Therefore I am enforcing standard naming convention. You have the option to override that behavior in the “manual” mode. But remember the script relies on host names as a way to evaluate the host role. Diverting from the standard disrupts the logic in certain functions (like show groups). The script will automatically assign a sequence number next to the base host name. For example in some functions you will be prompted to enter Indexer name; you should enter IDX only. The script will find the next unused sequence number and IP address and use it (example IDX01, IDX02, IDX03,..etc). That logic does not apply to the “site” portion of the hostname.

**How to use:**

You have the ability to control verbosity level by using –v switch. I used I/O redirection to control verbosity (and logging level).

*create-splunk.sh –v3*

Experiment with creating few hosts then point your browser to them. Push the server to the limits to see how many host can you create before your system crashes.

*Choose option C*

Add license file(s) to your containers. Make sure you have your all your license files in a directory accessible by the script. This option will be overridden if the host becomes a license-slave

*Choose option 6*

The real fun starts on the second clustering-menu. Select any item from the first 1-4 choices then watch the script create everything for you. Once you get familiar with things; then move to the “manual” mode options 5-8. In manual mode you will be able to specify exact hostnames and how many “hosts” to create. Please follow the standard described above. Validate the actions by pointing your browser to any host you create.

There are few optional items (open source) that I included in the container build that are OS related; it’s up to you if you want to install those items:

-screenfetch : banner screen show host info at ssh login

-bashrc: customized bash file

**Navigation:**

There are two menu screens the main menu. Here is a brief explanation of important options on the main menu:

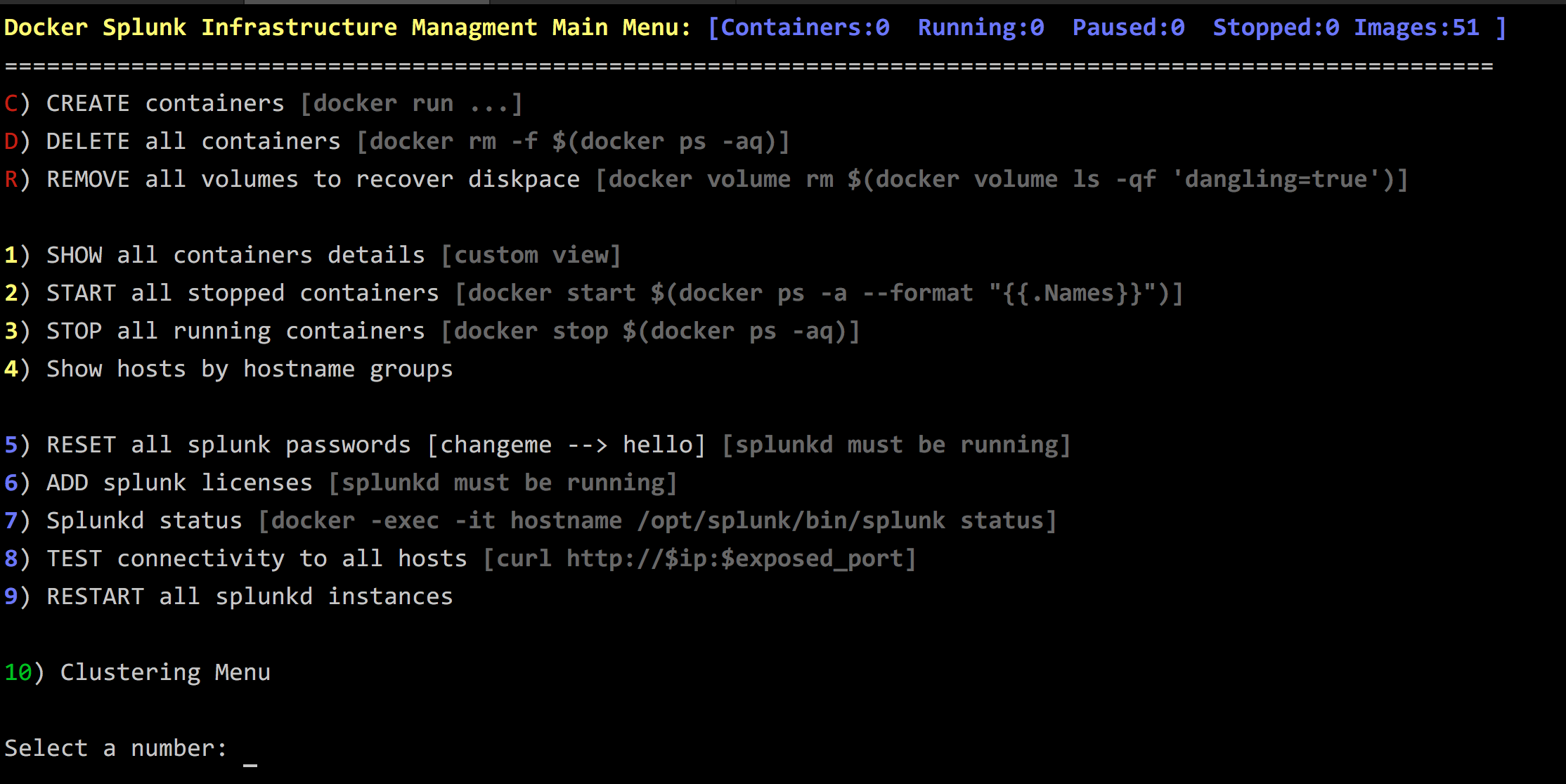
**C) Create containers**: Allows you to choose the container name and how many “hosts” to create. Good options if you are to doing a search party or just classroom with stand alone Splunk instances.

**C) Delete container**: Allows you to delete all containers in the system

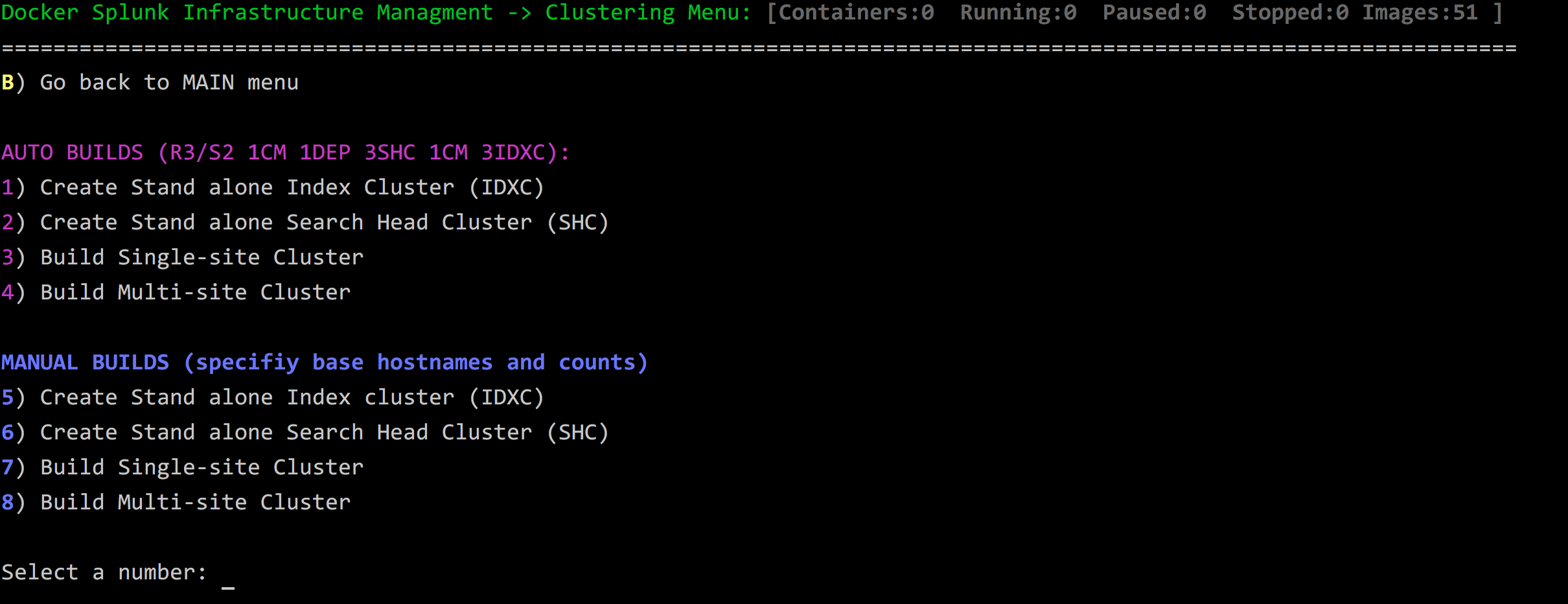
**R) Remove all volumes**: Docker does not remove any container artifact when deleted or shutdown. You can clean and save disk space using this option.

**4) Show hosts by group**: Useful to display categories of the environment by function. It will show all cluster-masters (CM) and possible members associated with it. The same goes for Deployer (DEP) servers.

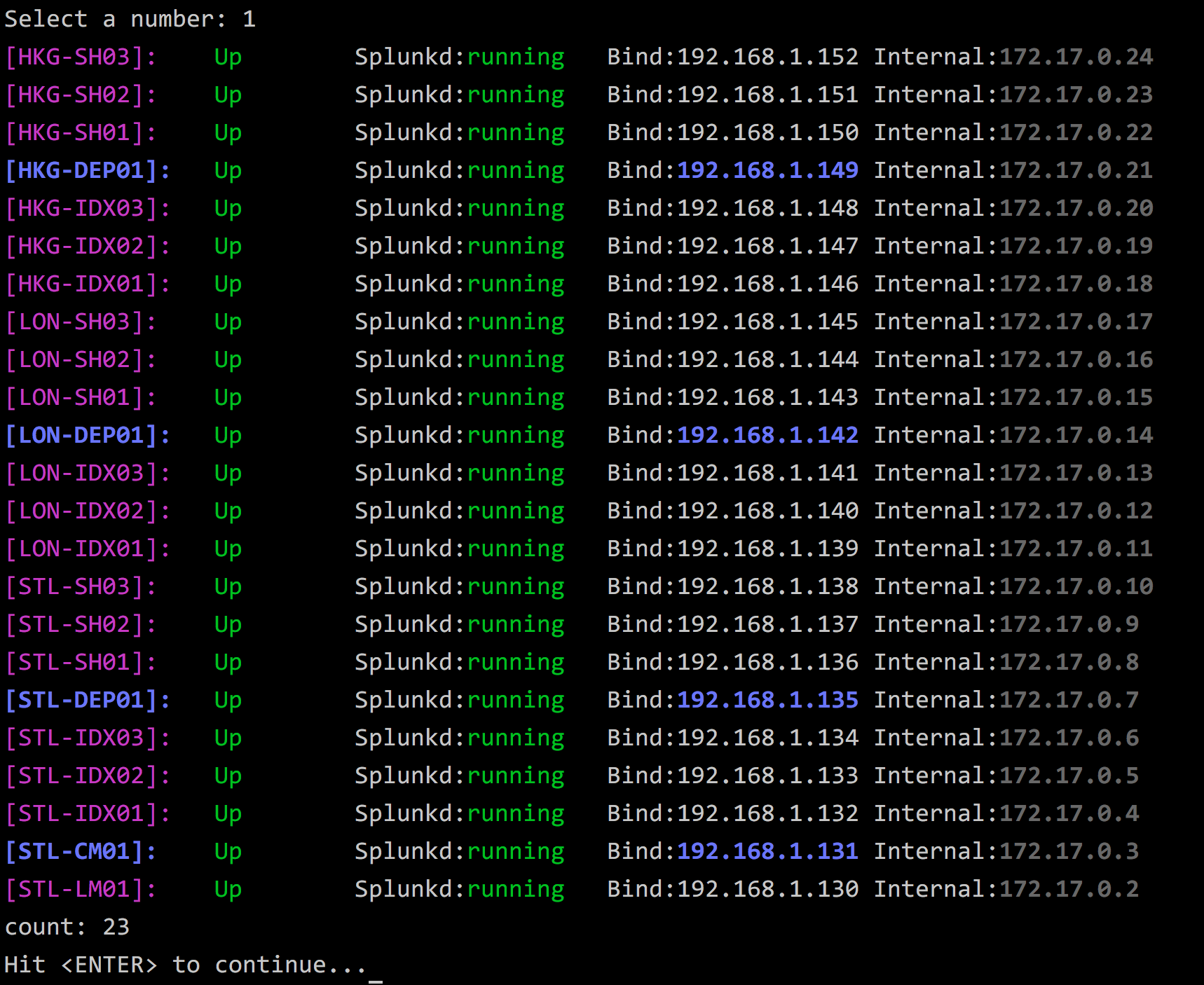
The rest of the options are self-explanatory

****

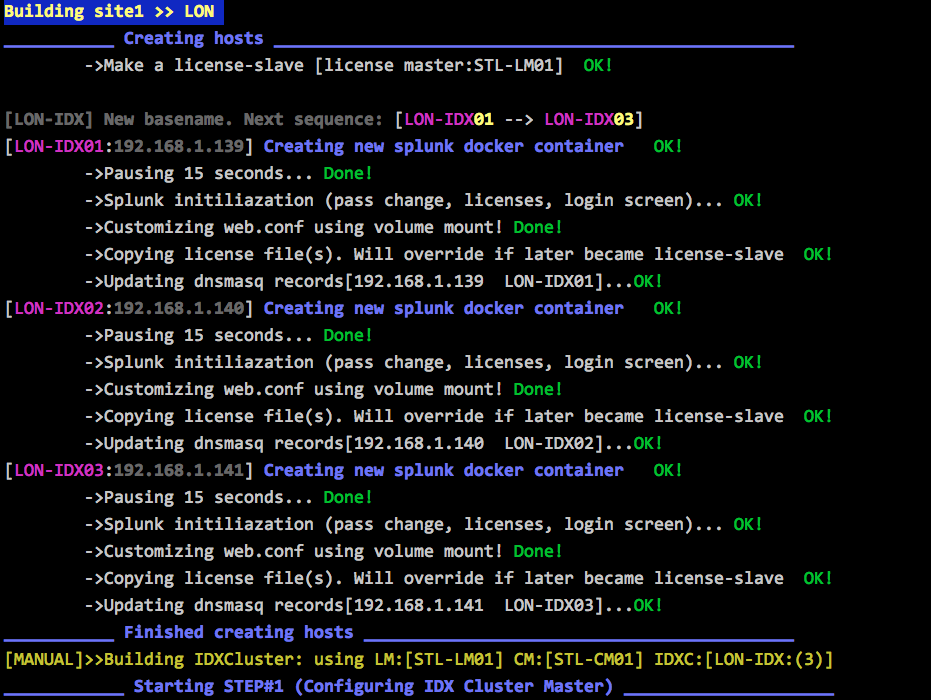
**Cluster-building menu:**

****

**Sample list of all hosts created:**

****

**Sample execution (log level 3):**

****

**Dnsmasq (optional):**

Normally you don’t need to setup DNS resolution and you can just plugin the hosts IP directly in your browser, however, setting up caching dns (dnsmasq) will simplify your life. In my lab I have dnsmasq running on the same host that’s running docker. Every time a container is created an entry is added /etc/docker.hosts.dnsmasq. Configure dnsmasq conf file to use this host file by adding this line to /etc/dnsmasq.conf:

# update eth0 to suit your Host machine

interface=eth0

cache-size=1000

#no-poll

listen-address=127.0.0.1

#listen-address=192.168.1.100

resolv-file=/etc/resolv.dnsmasq.conf

no-hosts

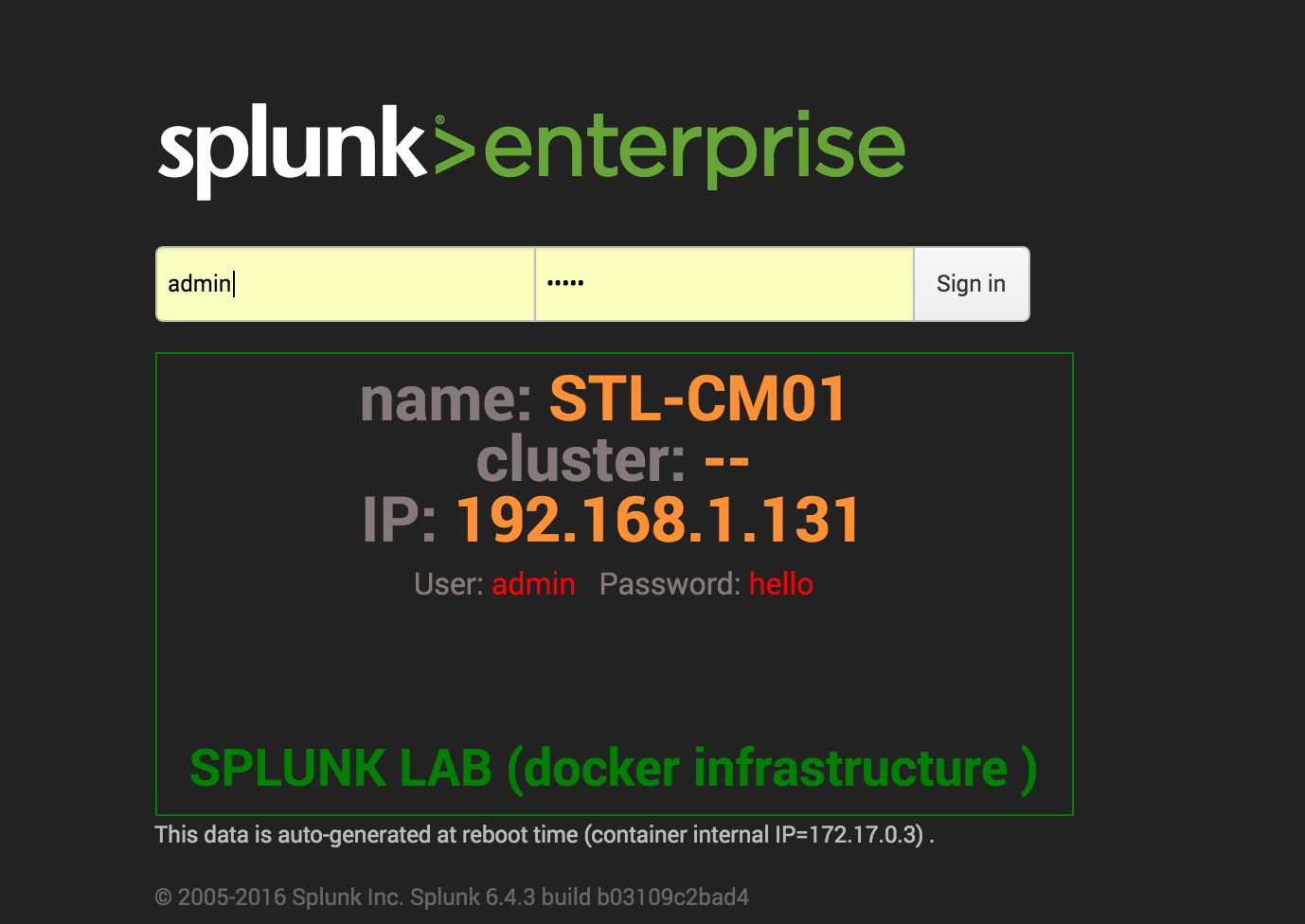
addn-hosts=/etc/docker-hosts.dnsmasq

Your laptop should be configured to get DNS from the docker-host (Ubuntu server). If you want to have a portable environment (using consumer router, Intel NUC and your laptop) then simply don’t change your laptop network setting and just have your router obtain DNS from docker-host.

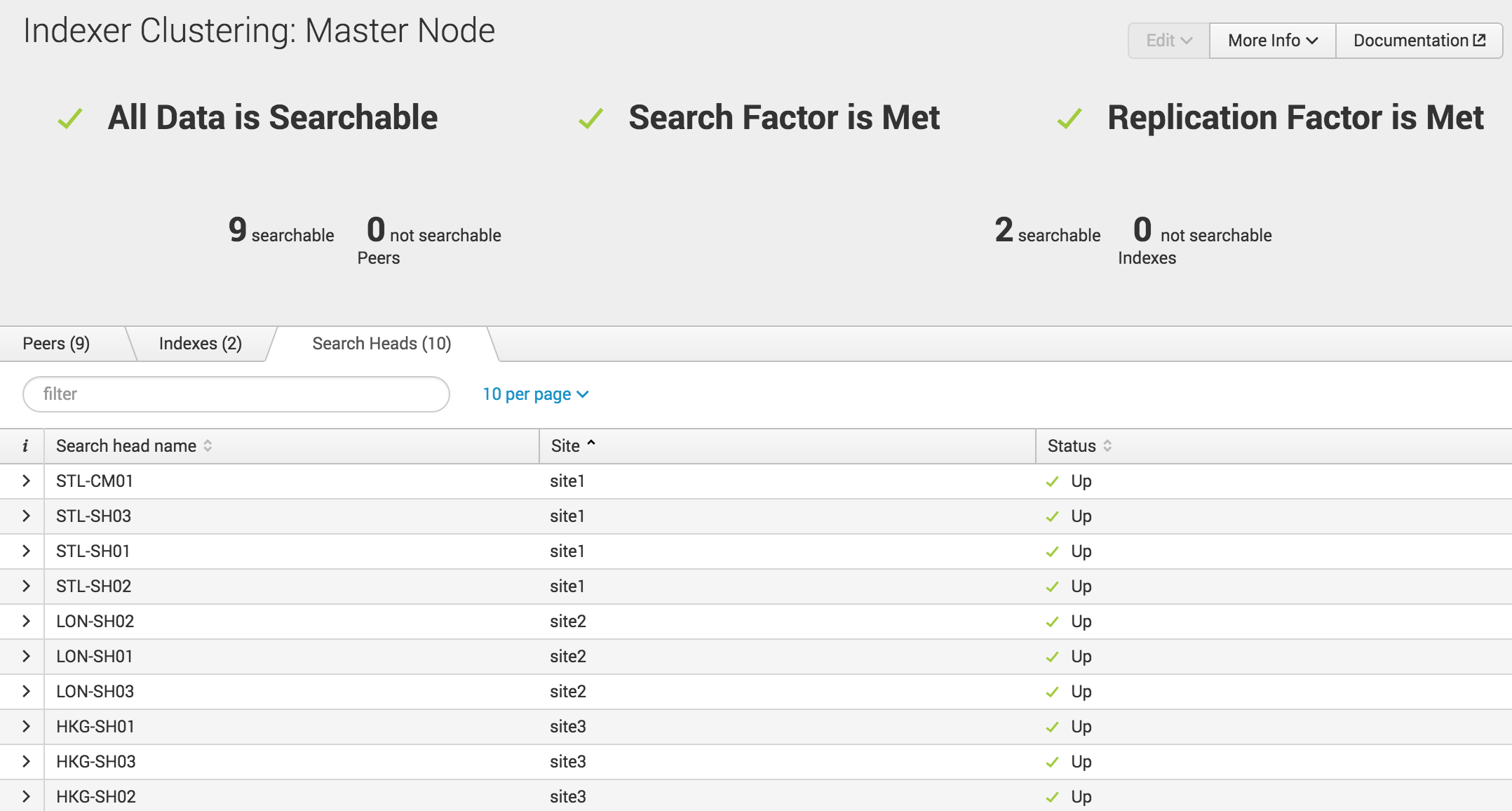
**Testing:**

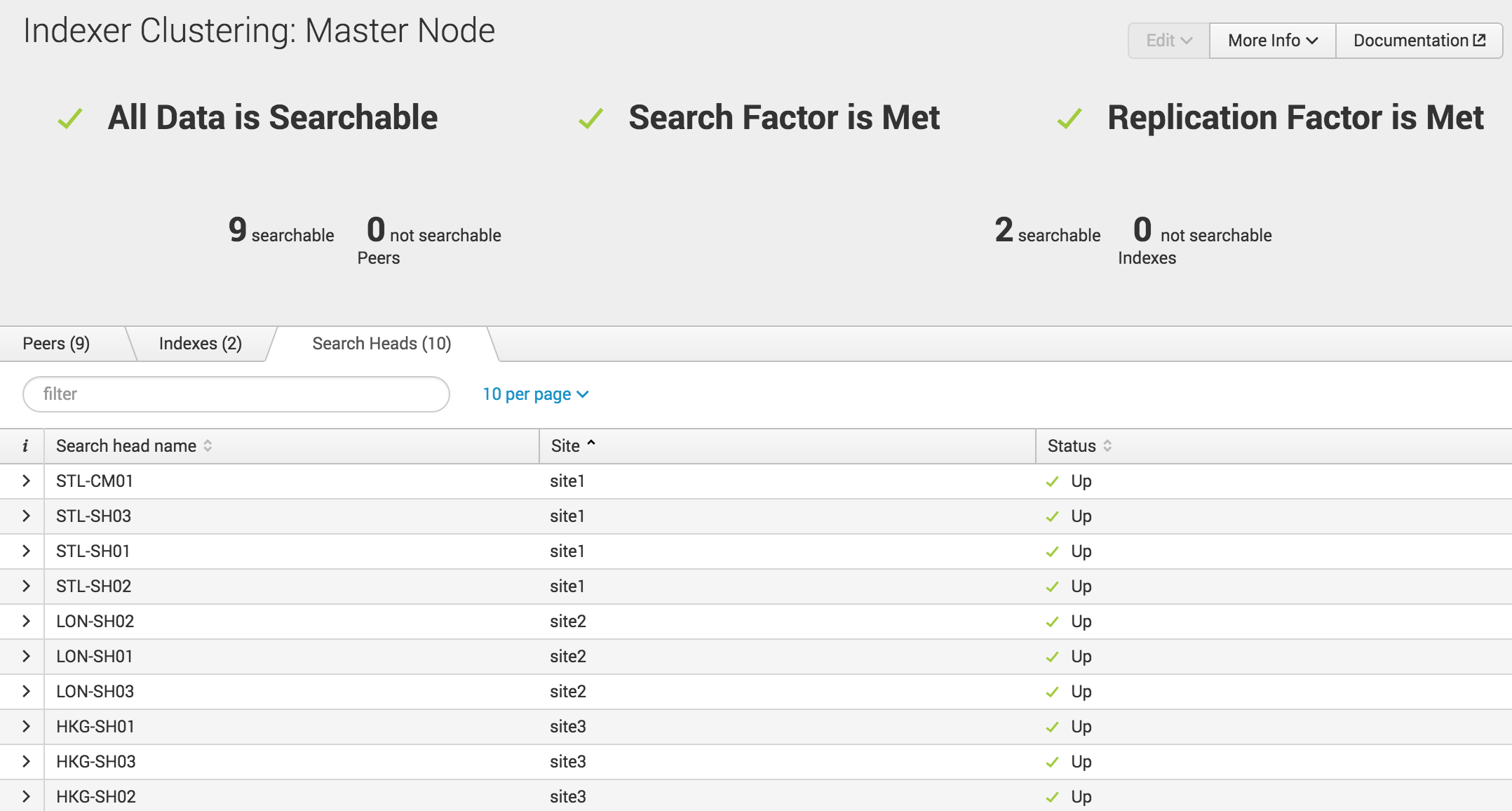
During testing I was able to push upward of 60 Splunk docker containers on a single 16GB ram, i3 machine. Creating 23 hosts multi-site cluster environment took about 30 minutes. Similar number of hosts but stand-alone instances takes significantly less time to create. I used 8-port Netgear switch (which will be replaced by consumer Wi-Fi router), one Intel NUC device. The entire infrastructure can be destroyed and re-created in few minutes. Of course if you expecting higher load (i.e. live data ingested or large number of users); then you may need beefier hardware. If anyone use this script in production like environment please let me know. I am interested in learning about your experience.

**Sample login screen:**



Sample showing three sites site-2-site cluster:





**Conclusion:**

I am really excited about the possibilities and the use cases people will come up with. My main objective during writing the code was building portable lab environment that I can just kill and re-create consistently. I also wanted to shield the user from learning docker and just focus on learning Splunk. I have no doubt in my mind that this script can probably be used in production environment that handles real traffic and real users with proper resources sizing. I want to leave you with a final though; I don’t expect Docker to replace VM technologies but in a specific Linux driven app-context like Splunk it is, in my opinion, a better choice.

**TODO:**

-UF and HF containers with eventget.py (simulate traffic)

-Dynamic RF/SF

-Load testing on faster hardware

-Test on CentOS (redhat)

-MAC OS support, followed by Windows

- Do everything with CLI only

**Additional screenshots (4 sites each location 3 SH, 20 IDX) running on NUC Skull (32GB ram, 8 core)**

