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

**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.

As with 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 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 (including supporting Splunk elements like DS, CM or LM). Or 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 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 up upward of 40+ Splunk docker containers for a classroom lab using low powered Intel NUC device (i3 16G ram, 128G SSD). What’s impressive about docker is that the footprint on my docker-host was extremely small compared to similar activity using a VM solution.

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. However, I 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 get familiar with simple 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 features list:**

-Extensive Error checking

-Load control (throttling) if exceeds 4xcores

-Built-in dynamic host names and IP allocation

-Create and configure large number of Splunk hosts very fast

-Different logging levels (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-docker-build>

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 on this code on MAC OSX or Windows, nor I recommend doing so at this point in time (unless more native solution is developed for these two platforms). Windows and OSX do not support c-block 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 if don’t use Linux as docker 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; it will create the 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).

**How to use:**

The only configurations you will need to make this thing work are listed below. Or 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 run my dnsmasq caching DNS server. If you don’t want use dnsmasq you; then just use actual IPs in your browser. The container you create will start at 192.168.1.130 and last one will end at 192.168.1.250. If you wish to set 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"

**STEPS:**

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

-Run the script *create-splunk.sh –v3*

-Create few hosts using “C” option then point your browser to test. Push the server to the limits to see how many host can you create before your system crashes.

- Check the status of splunkd or restart splunkd

- (D)elete or (R)emove container

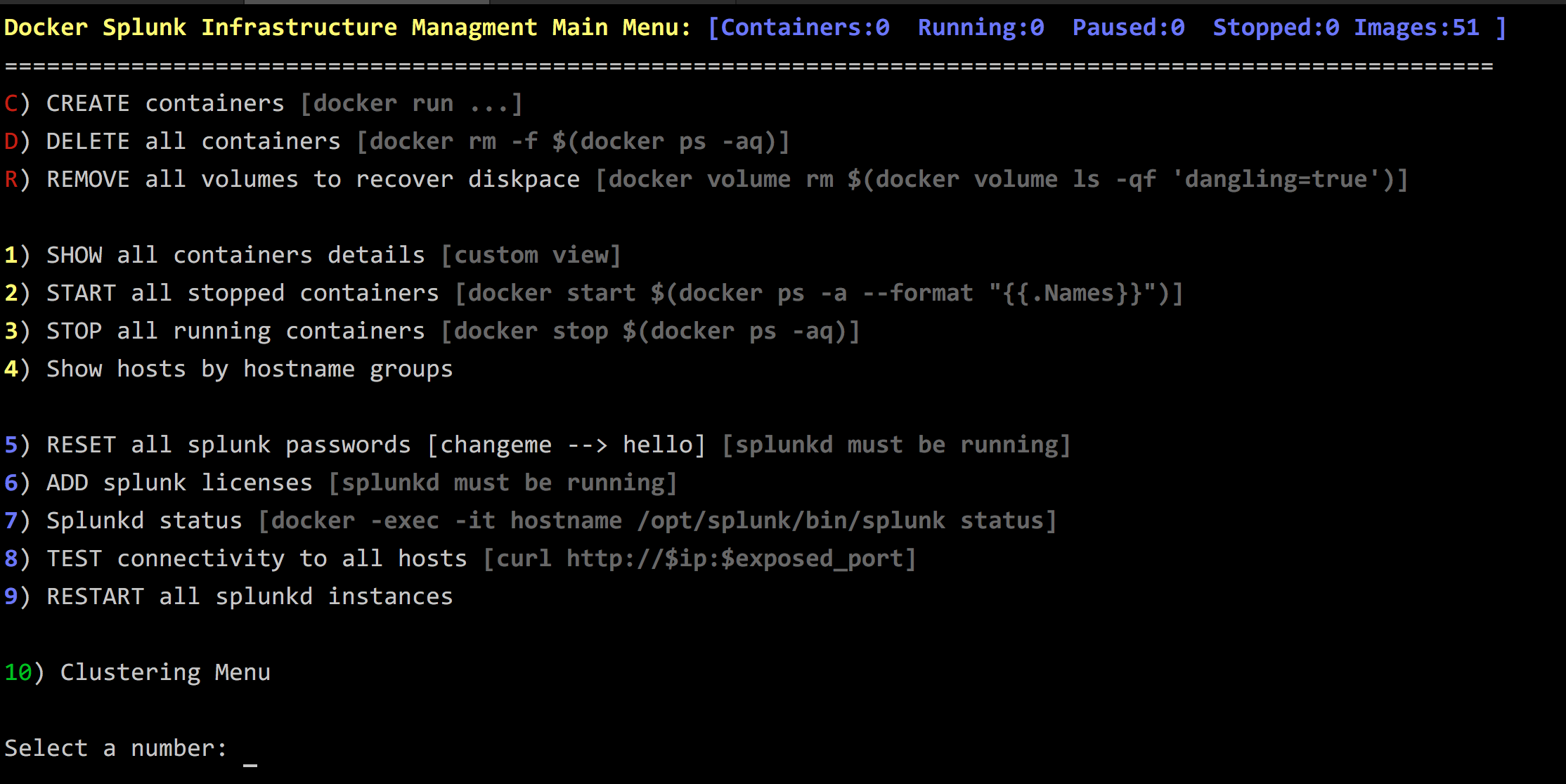
The real fun start on the second clustering-menu, it’s where you will be able to instantly creating the entire environment. Select any item from the first 4 choices and watch the script creating everything for you.

-Once you get familiar with things move on to the manual option (5-8), their where you will be able to specify exact hostname and how many “hosts” to create.

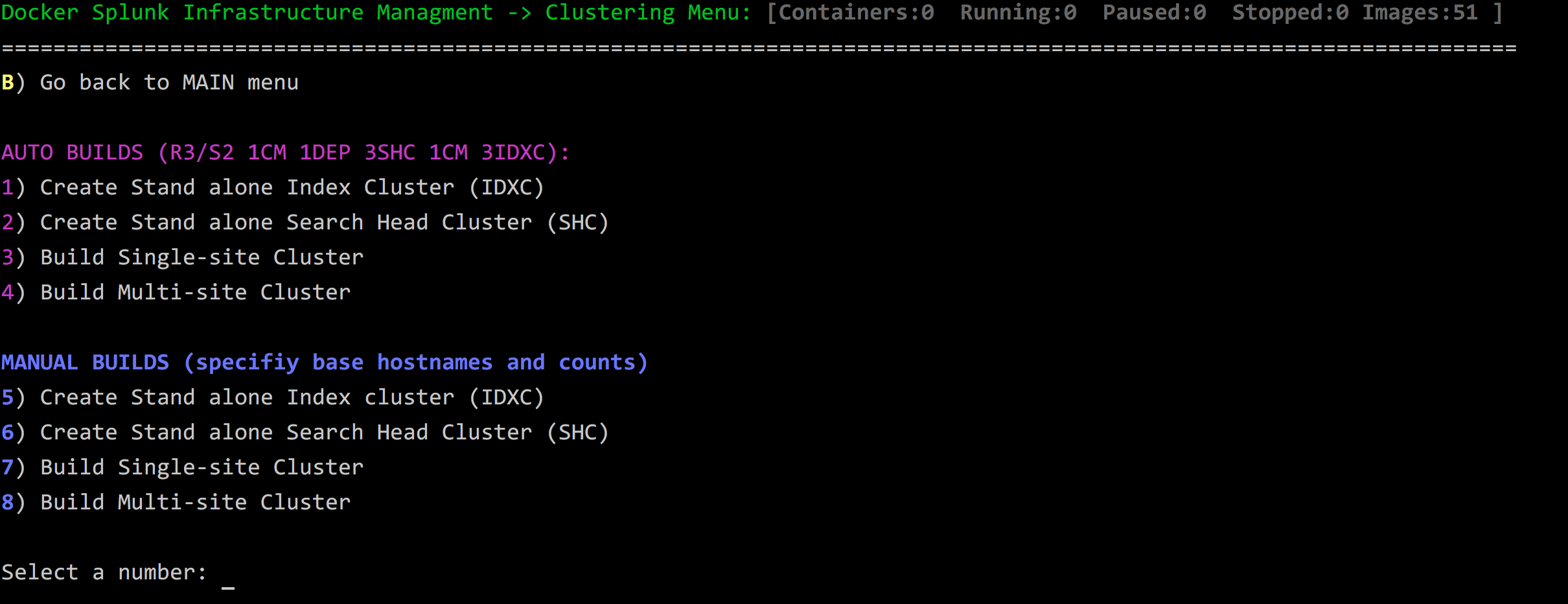
-Validate the actions by pointing your browser to any host you create.

**Navigation:**

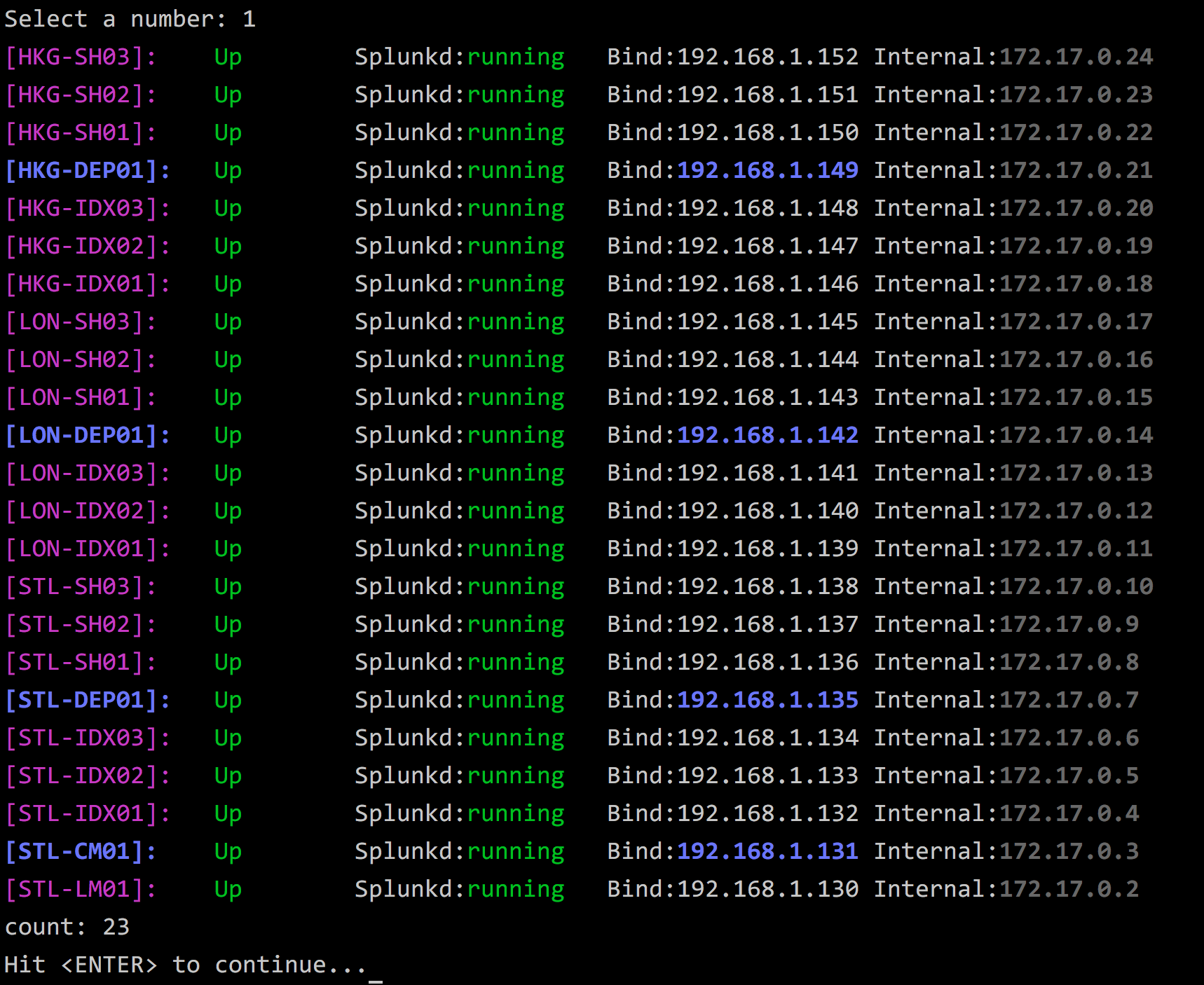
There are two menu screens the main menu

****

**cluster-building menu:**

****

**Sample list of all hosts created:**

****

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 option if you are to doing a search party of just classroom with standard 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

The second menu is dedicated to cluster building. The first four choices (1-4) will build the environment you request with some defaults (hosts counts and names)

The second portion (option 5-8) is a repeat of the above but you get some control to how many hosts to create and how to use the CM.

**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 filel 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

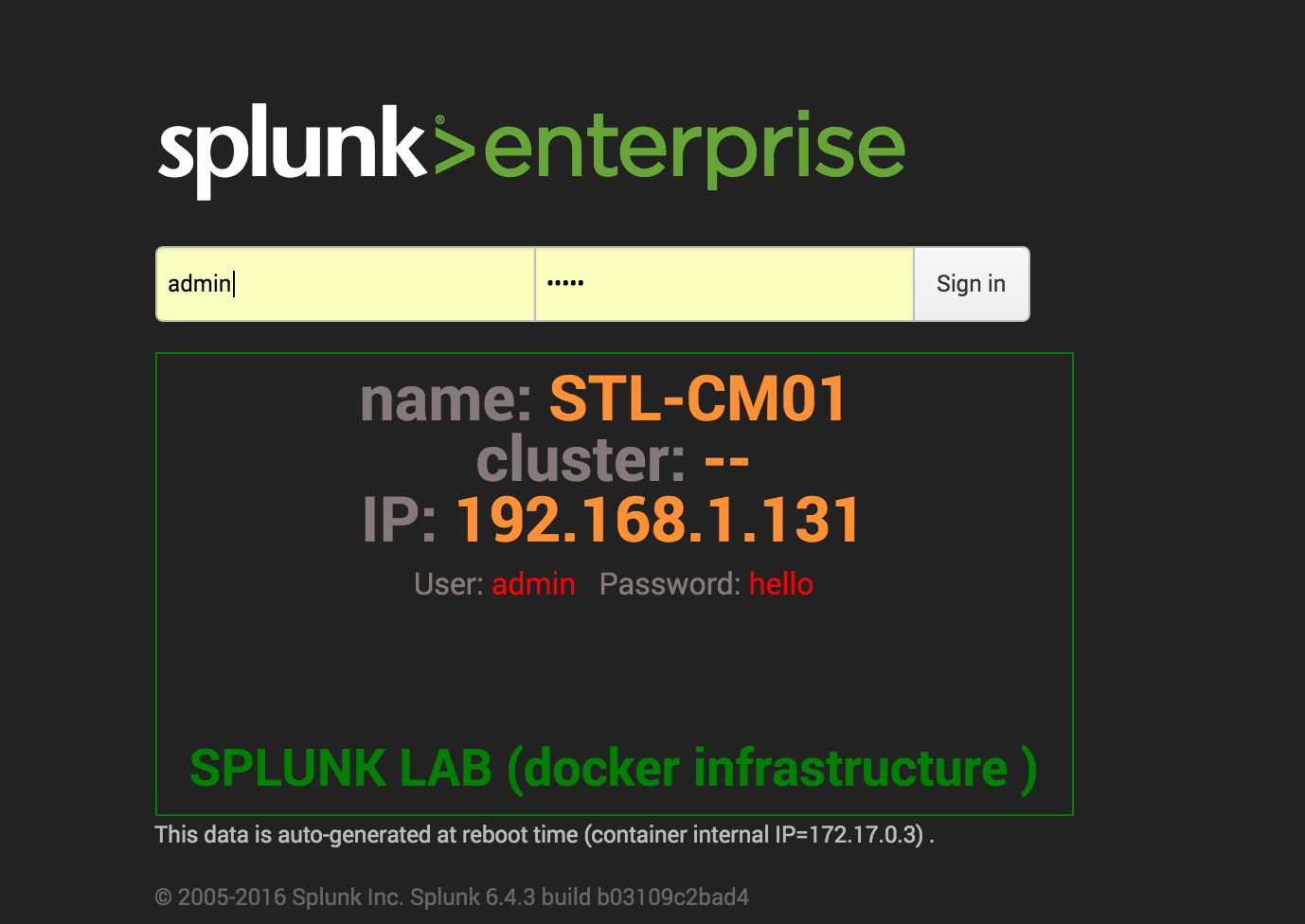
You will need to point your laptop (where the browser is running) to point to the docker host (Ubuntu server). If you want to have a portable environment (consumer router, Intel NUC and your laptop) then simply configure the router to get its DNS resolution form the docker host.

**Testing:**

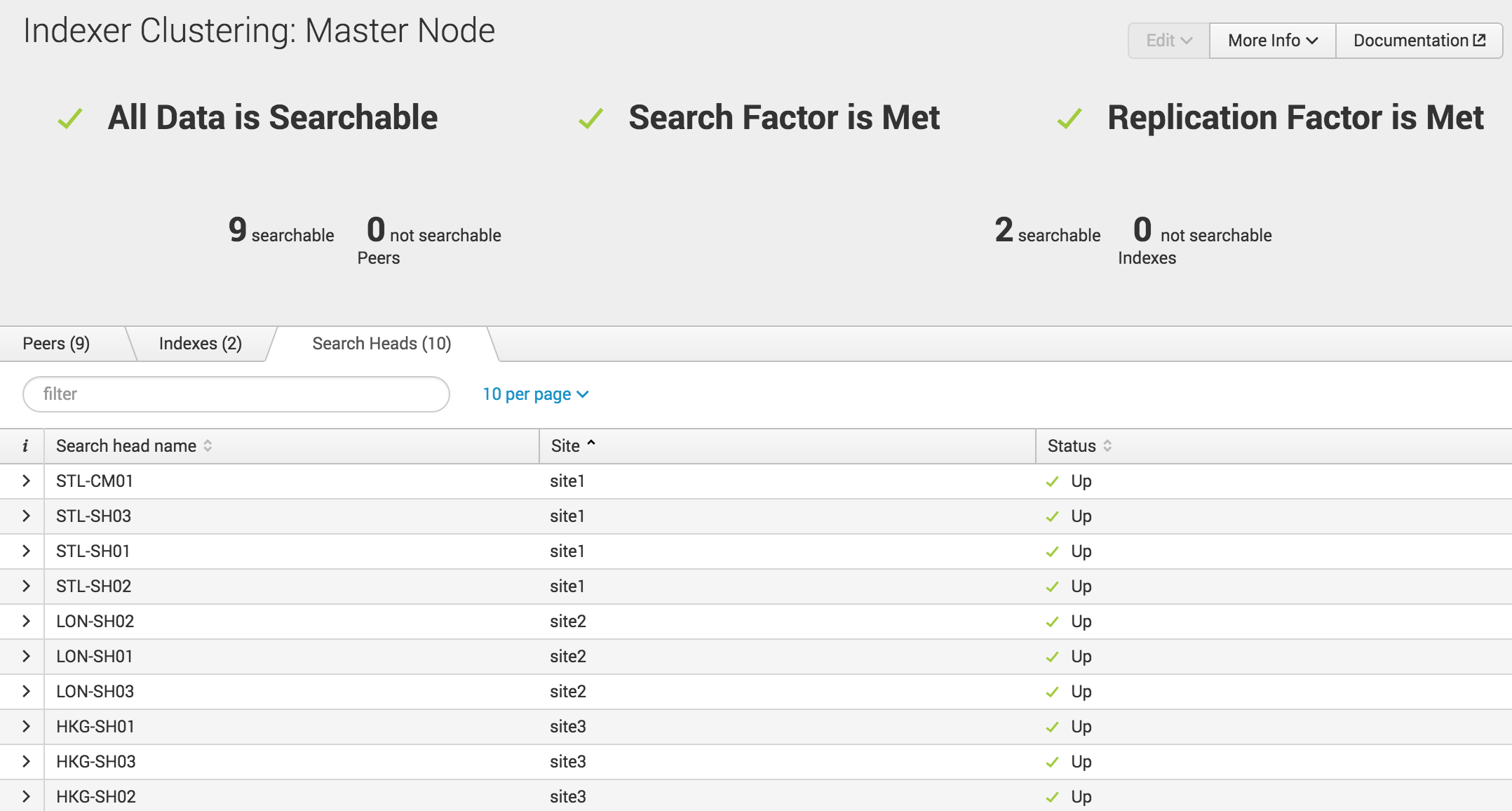
During testing I was able to push upward of 60 Splunk docker containers on single 16G i3 machine. Creating 23 hosts multi-site environment took about 30 minutes. Similar number of hosts but standalone instances will take significantly less time to create. I used 8-port Netgear switch (which will be replaced by consumer Wi-Fi router), one Intel NUC device. My goal is to have a portable environment that I can bring to the customer site and have 20-30 people using the system. The entire infrastructure can be destroyed and re-created in few minutes. Of course if you expecting higher load (ie live data ingested or large number of users); then you may need beefier hardware.

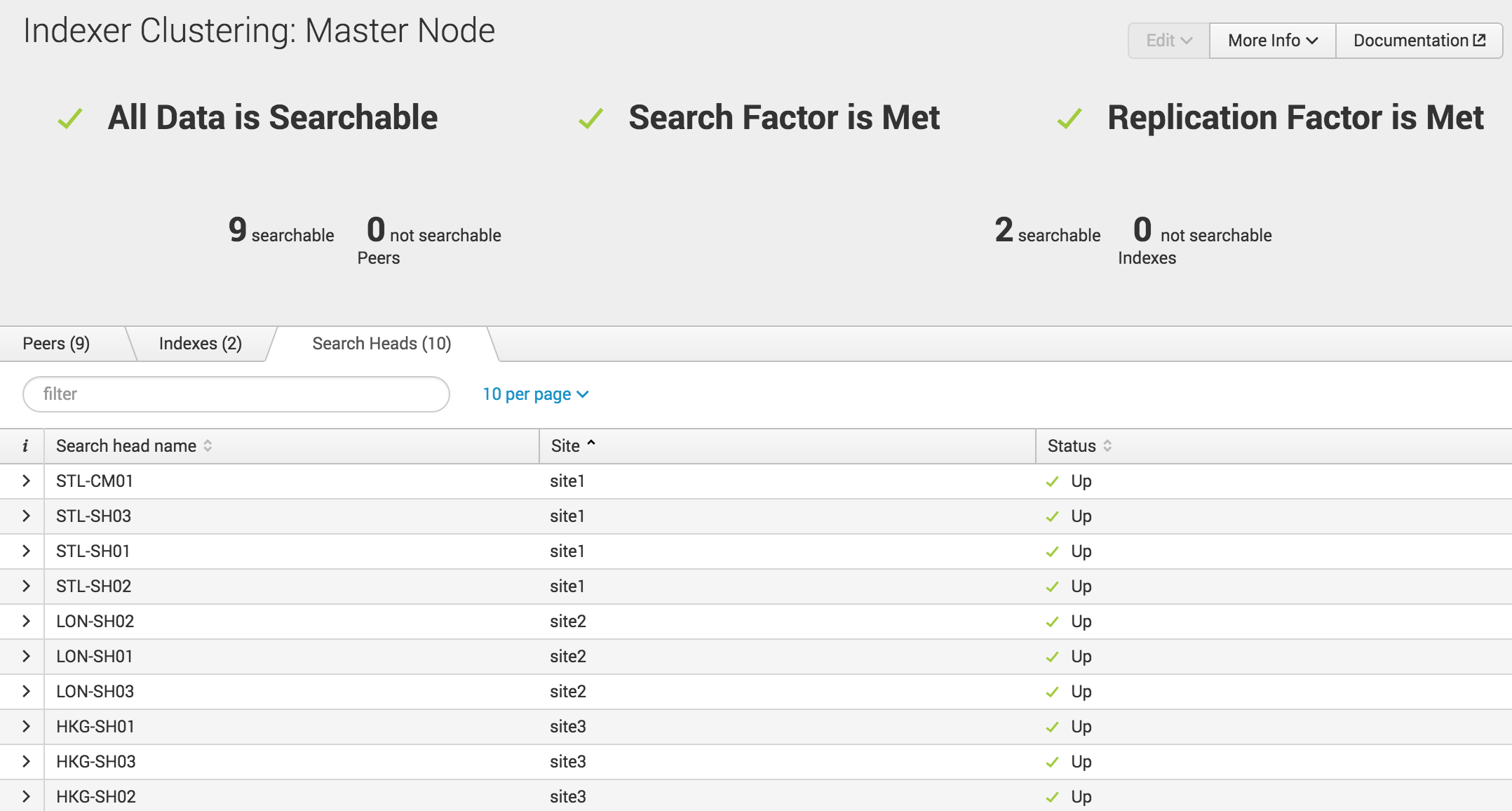
Hostname rules:

Sample login screen:



Sample showing three sites site-2-site cluster:





**TODO:**

-UF and HF containers with eventget.py

-RF/SF