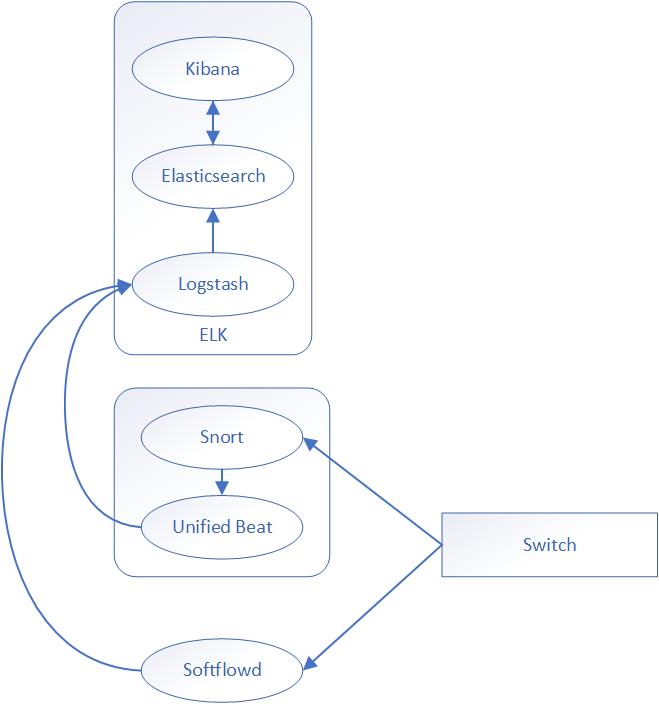
**Design Spec for ELK VM and Beats**

This document attempts to capture relevant details of the set-up, configuration, and maintenance of the ELK server and all of its beat components. The ELK installation was mostly via apt and was closely based on <https://www.rosehosting.com/blog/install-and-configure-the-elk-stack-on-ubuntu-16-04/> except for a modified nginx config.

VM setup

|  |  |
| --- | --- |
| Kronos (Ubuntu 16.04) | |
| User | elk-root |
| Password | Blackout123 |
| Interfaces | |
| ens160 | 192.168.0.94 |
| ens192 | Listens to l2 mirror port of switch at 192.168.0.87 |

The new vm, Kronos, has all required components/configs installed as services: elasticsearch, logstash, kibana, softflowd, nginx. All these services should come up on reboot within about a minute (logstash can take ~30s). Debugging and other service configuration can be done with the systemctl status command, e.g. ‘*sudo systemctl status logstash’*.

The vm receives these primary types of data, with new indices per day (see logstash section for how this is done):

* Netflows: These are captured from the switch mirror port
* Sysmon data: sysmon event data
* Winlog: Windows event data, for windows hosts on our system
* Snort: IDS data
* Bro: IDS data
* OpenVAS: not known if this is still needed, its inherited from the old vm

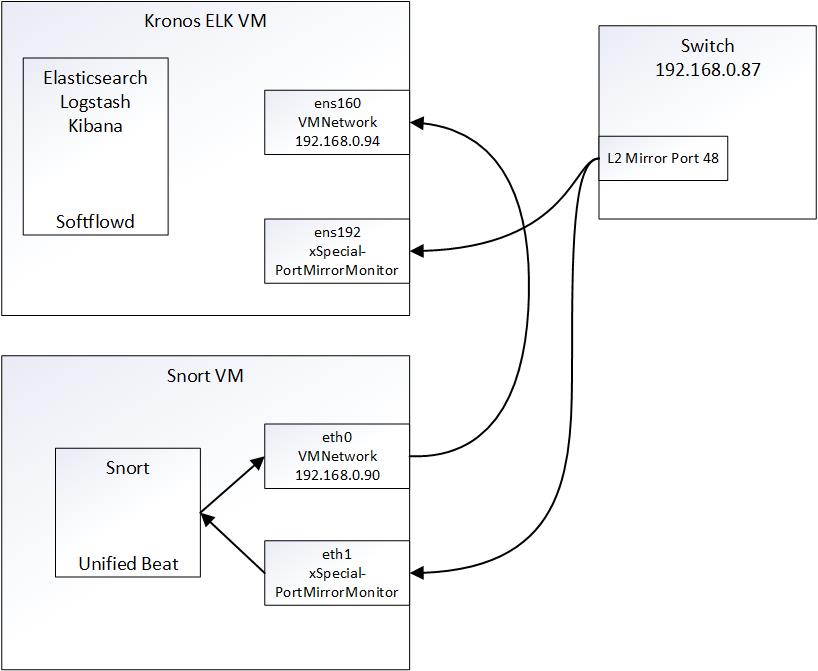


Figure 1: Basic diagram of vm network interfaces and processes.

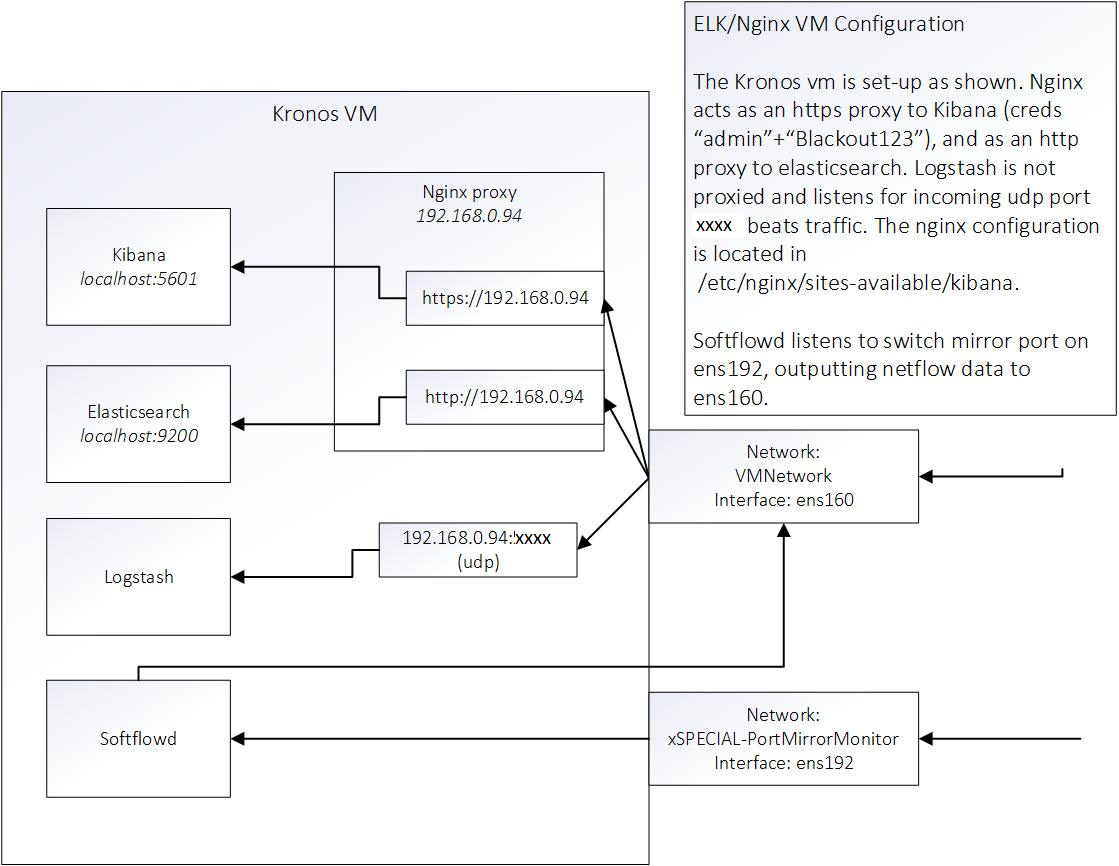


Figure 2: Nginx proxy diagram. Logstash ‘xxxx’ port represents any udp port for which logstash has been configured to listen for a particular beat. Each conf file in /etc/logstash/conf.d specifies these ports per beat.

Nginx

Nginx proxies to kibana and elasticsearch using the following setup. Nginx kibana creds are user=”admin” password=”Blackout123”.

1. sudo apt install nginx
2. echo "admin:$(openssl passwd -apr1 YourStrongPassword)" | sudo tee -a /etc/nginx/htpasswd.kibana
3. sudo rm /etc/nginx/sites-enabled/default
4. sudo nano /etc/nginx/sites-available/kibana, and add the following:



Figure 3: Config for nginx

1. sudo ln -s /etc/nginx/sites-available/kibana /etc/nginx/sites-enabled/kibana
2. sudo nginx -t

sudo service nginx restart

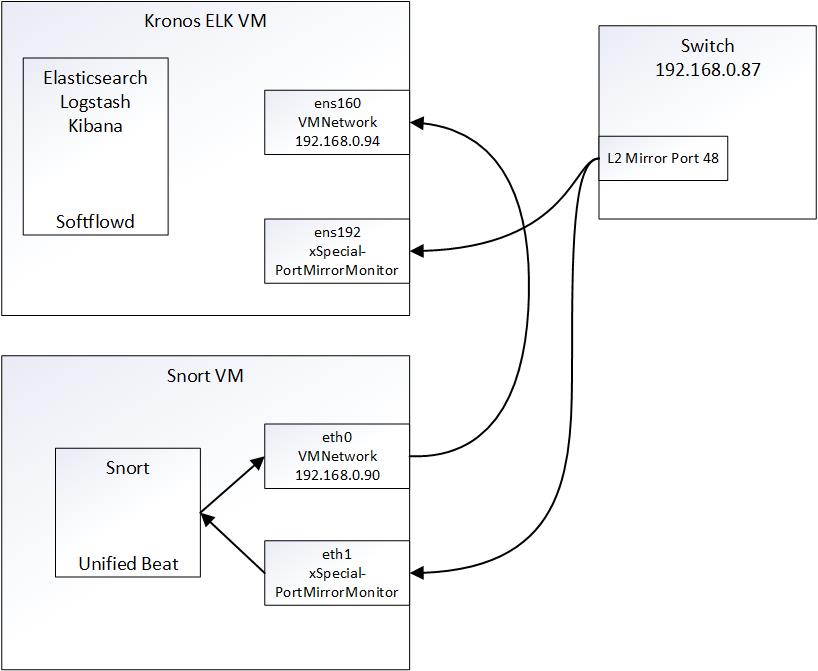


Figure 4: Netflow capture. Both Snort and Softflowd listen to netflow data from the switch mirror port (the xSpecial-PortMirrorMonitor network), process it, and forward data to their corresponding logstash shippers.

Netflow Configuration

Softflowd receives flow data via the switch mirror port. The switch is at 192.168.0.87, but note that this is the ip of the switch itself; the mirror port is link-layer port 48 (I think) which appears on the vms’ settings as the xSpecial-PortMirrorMonitor.

Softflowd is configured as a service, which just manually calls the following command on boot to generate netflow-v9 data and send it to logstash at 192.168.0.94:9999:

‘usr/sbin/softflowd –v 9 –d –i eth1 –n 192.168.0.94:9999’.

The service configuration is defined in /etc/init.d/softflowd.

Logstash configuration

Logstash receives data from beats or other sources and ships it to elasticsearch, creating a new index per day for each data type. Each logstash shipper is configured via ‘conf’ files, located in /etc/logstash/conf.d/. The files are self-descriptive and should be used as the reference for how each logstash shipper is configured. For example, daily index creation is a result of the semantics of these conf files, where each conf contains a line with a date-format string, similar to:

index => “winlogbeat-%{+YYYY.MM.dd}”

At boot, the logstash service consumes the conf files to kickoff each shipper; so restart the logstash service if changes are needed.

**NOTE:** Be mindful of user/group privileges with logstash. A headache setting up logstash as a service involved a specific problem with access privileges and groups. Installing logstash via ‘sudo apt install logstash’ installs all of logstash with root privileges; but the service is started by systemd as user=’logstash’ group=’logstash’. The result was that the logstash service would start, but would not consume the files in /etc/logstash/conf.d/ to start each of our shippers. No data was sent/received since the ‘logstash’ user/group had no privileges to access /etc/logstash/conf.d. This was fixed by changing all of /etc/logstash files and subfolders to the ‘logstash’ group; similar problems litter ELK forums.

* 1. ‘sudo chown :logstash /etc/logstash/\*’
  2. ‘sudo chown :logstash /etc/logstash/conf.d/\*’

Debugging:

1. Logstash runs as a service and running another instance manually will conflict with the running service, but running manually is useful to test a new conf file:
   1. ‘/usr/share/logstash/bin/logstash –f [path to conf file]’
2. There is also a parameter to simply test a configuration of group of configurations:
   1. –t, --config.test\_and\_exit
3. See the logstash docs online for other command line options for creating and testing new configs.
4. *“Why does logstash run manually from the command line, but doesn’t see my conf files when run as a service?”*

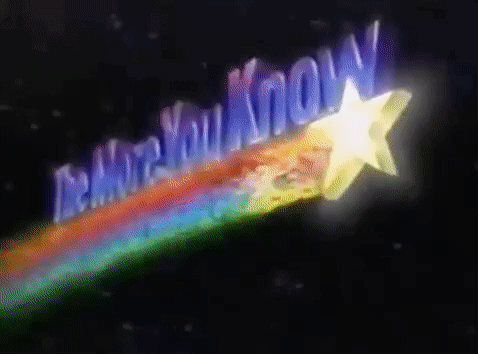
Be aware of users/groups and their permissions! See the note above. Running logstash manually (as root, for example) changes its file privileges. The service runs as user=logstash group=logstash and only has access to that user/group’s resources.

Winlogbeat

Winlogbeat must be setup for each windows host of interest, and once installed, ships data to the logstash winlog shipper at 192.168.0.94:5144.

Sysmon

Runs on top of winlog. We need to implement filters to cut down on sysmon’s ability to overwhelm logstash/elasticsearch with data.



...the more you like ELK!

Elasticsearch and Kibana

Elastic and kibana are pretty self-reliant, but some useful fun facts:

1. Curl sucks! Not really, but the kibana console is better. It pays to be slick with the kibana query console for:
   1. Developing/debugging elastic queries
   2. Deleting/managing indices
   3. Peaking at index data, e.g. *“How is winlog data structured?”*

Browse to Kibana, then click “Dev Tools”-> “Console”. Query history is saved as well, so you can view past query research you’ve done, just click “history”.

1. Transferring indices: This is a well-supported use-case in the ELK community, and there is a “\_reindex” api for it: <https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-reindex.html>. Also see tools like Elasticdump. Be sure to thoroughly research and test reindexing before attempting to update between versions!
2. See the elastic docs for raw query examples for every type of query. Most include a link to run the query in the console
3. Use \_cat (e.g. “localhost:9200/\_cat/indices”) and other endpoints to view indices and status
4. “Yellow” vs “green” status: google the “Cluster Health api”. Yellow isn’t bad, it usually just indicates that shards of indices don’t have replicas yet. Since our ELK vm is just one node, there are no other nodes over which to derive replicas, so we see lots of yellow statuses next to indices.