Alternatives to the components

Given the large environment of available tools to standardise and correlate logs, we can often create a setup using other components that talk to each other.

- fluentd is an alternative for logstash
- Grafana is an alternative for Kibana
- Druid is a high-performance, column-oriented, distributed data store as an alternative Elasticsearch
- Riemann aggregates events from servers and applications with a powerful stream processing language.

Full stack alternatives

- TICK Stack
- Graylog

General Best Practice

Elasticsearch Best Practice

- set `$ES_HEAP_SIZE` env var to **1/2** of RAM (but < **32GB**)
- Disable memory swapping by enabling `bootstrap.mlockall`
- Set user's file `ulimit` to `unlimited` (Need reboot to check)
    - You can check with an API call as well `/_nodes/process`
- Use the default configuration and make small changes as required
- Multicast is great, but when you are going to production make sure to use `Unicast` discovery mode
- To eliminate the "Split-brain" problem use **3** lower resource master-eligible nodes in larger cluster environments (dedicated)
- You don't need beefy machines, simple machines are ok due to the distributed nature of elasticsearch
- Add lightweight client nodes (no data)
- Use Snapshot and Restore. This is very useful (but different from `replication`)

Logstash Best Practice

- Watch out for Grok Filter data (GREEDYDATA) as they use a lot of resources especially CPU and Memory. Try to get as specific as possible
- Test your configuration with `-e input{...}... output{...}`
- Use `-b` flag to send bulk requests to elasticsearch
- Use `-w` flag to utilise multiple cores. This is especially useful for multi core and bulk processing
- Use the generator input for benchmarking (https://github.com/matejzero/logstash-benchmark) and to understand performance and optimisation metrics

- If something goes wrong try `-- debug` for more detailed output (don't forgot to turn this off when you are done)

Kibana Best Practice

- Tune Queries in elasticsearch for maximum performance
- Configuring number of threads in pool
- Save and Export dashboards as a JSON File for reuse
- Deploy a proxy so that you can do basic authentication and other load balancing services
- While Kibana is an exploration tool, make sure you watch out for over-eager users affecting performance

Production

- Access Control / Security
    - use nginx/apache to setup basic authentication
    - You can block `POST` / `PUT` / `DELETE` operations
    - Disable Scripting (Version < 1.2) `script.disable_dynamic: true`
    - Disable destructive actions `action.destructive_requires_name: true`
    - Use aliases to allow users access to subsets of indices
- VM vs Metal
    - VM's are convenient (Auto scaling, no management, etc)
    - Bare metal is generally more configurable and higher in performance
    - Metal can utilize SSD's
    - Cloud VM's can suffer from `noisy neighbors`
    - But you should start using what you're most familiar with!
- Disks
    - Spinning disk's are cheaper per GB
    - SSDs have better IOPS
    - SSDs are cheaper wrt: IOPS
    - SSD manufacturing tolerance can vary (vendor based)
    - SAN / NAS can work, if IOPS are sufficient (throughput, iops, etc)
    - Don't necessarily need RAID, ES handles redundancy
        - But striping can help with performance
        - You can use shards and replicas in ES

Security

- Harden the base server with traditional security techniques
- Use SSH key for login
- Remove root login
- Use randomly generated passwords

```
openssl rand -base64 24
```

- Enable the host firewall and allow only connections from specific IPs
- Use SSL certificates and enable HTTPS for Elasticsearch, Logstash & Kibana (E.g.: Lets encrypt)
- Use search guard for granular permissions and role based authentication for ELK (Shield is an alternative)

Monitoring Services

- Enable service level monitoring for Elasticsearch, Logstash and Kibana
- Use monit (or) uptime robot for monitoring services (you can also use Icinga)