Update ELK

aim = update the current production version of ELK to the latest version.

method = install the current version on two testing servers, and then update them.

TLDR - files at devoptools/docker/elk, change to the same version all components.

Elastic search

- find the current running version that is stable
  - Install Elasticsearch with Docker | Elasticsearch Guide [8.17] | Elastic
- `ssh root@root@devops-srv-01` + `ssh root@root@devops-srv-02`
- config files at `devopstools/docker/elk_update_version_test/elasticsearch`
  - certification in /opt/certs
- run elasticsearch container on each of the two servers (if you run only on one it wont work) using the `run_elasticsearch_container.sh` script.
  - if there is a problem with virtual memory - Docker - ELK - vm.max_map_count
- access at https://server_ip/9200 with elastic:elastic
  - verigy health of cluster -
    curl -u elastic:elastic -X GET "https://localhost:9200/_cluster/health?pretty" -k
  - check which streams are listening -
    curl -X GET "https://devops-srv-01:9200/_data_stream?pretty" -u elastic:elastic

Kibana

- `ssh root@root@devops-srv-01`
- config files at `devopstools/docker/elk_update_version_test/kibana`
- change the kibana_system users password
  - In elastic search -
    `curl -u elastic:elastic -X POST "https://localhost:9200/_security/user/kibana_system/_password" -H "Content-Type: application/json" -k -d '{  "password": "elastic" }'`
    - verify change -
      `curl -u kibana_system:secure_new_password -X GET "https://localhost:9200/_security/_authenticate" -H "Content-Type: application/json" -k`
  - In kibanas configuration -
    `devopstools/docker/elk_update_version_test/kibana.yml`
- run the kibana container using the `run_kibana_container.sh` script
  - no need for certification for kibana, SSL will be handled by nginx (unlike elasticsearch, which has its own SSL, not through nginx)
- access at http://server_ip:5601 with elastic:elastic (same as elasticsearch user and password)

metricbeat

- `ssh root@root@devops-srv-01`
- config files at `/devopstools/docker/elk_update_version_test/metricbeat`

- copy credentials from those of elasticsearch - .crt + .key + root.pem - into `/devopstools/docker/elk_update_version_test/metricbeat/ca`
- run the metricbeat container using the `run_metricbeat_container.sh` script
- testing connectivity of metric beat
  - log into kibana and see if there are new entries in the side bar “discover” under data view “metricbeat-*“
  - Verify Elasticsearch Outputs in Metricbeat Configuration
    `docker exec -it metricbeat cat /usr/share/metricbeat/metricbeat.yml`
  - Test Elasticsearch Accessibility from the Metricbeat Container
    `docker exec -it metricbeat bash`
    `curl -u <username>:<password> -X GET "<https://<elasticsearch_host>>:9200" -k`
  - Check Elasticsearch for Incoming Metricbeat Data
    `docker exec -it elasticsearch bash`
    `curl -u <username>:<password> -X GET "<https://<elasticsearch_host>>:9200/_cat/indices?v" -k`
  - Use Metricbeat’s Test Mode
    `docker exec -it metricbeat metricbeat test output`
- docker exec -it metricbeat_test bash
  - metricbeat setup
  - metricbeat -e
  - if path already taken -
    - rm -f /usr/share/metricbeat/data/metricbeat.lock
    - change the path in metricbeat.yml

Heartbeat

- `ssh root@root@devops-srv-01`
- config files at `/devopstools/docker/elk_update_version_test/heartbeat`
- copy credentials from those of elasticsearch - .crt + .key + root.pem - into `/devopstools/docker/elk_update_version_test/heartbeat/ca`
- run the heartbeat container using the `run_heartbeat_container.sh` script
- to check connectivity -
  - kibana → discover → create a new data view → pattern “heartbeat*“
  - kibana → devtools → GET heartbeat-*/_search

Add a document with time-data:

- PUT /test_yalee
- POST /test_yalee/_doc
  {
  "full_name": "com.skybox.view.agent.provisioning.cisco.csm.CiscoSecurityManagerModifyRuleTest.testModifyRulePosition_goodPath",
  "name": "com.skybox.view.agent.provisioning.cisco.csm.CiscoSecurityManagerModifyRuleTest",
  "classname": "com.skybox.view.agent.provisioning.cisco.csm.CiscoSecurityManagerModifyRuleTest",
  "run_time": "2024-11-25T10:25:44.297915+02:00",
  "duration": "0.003",
  "source_branch": "SKY-272017_content_validation_check_executable",
  "target_branch": "master",
  "suite_name": "provisioning-suite",

```
    "skybox_module": "TBD",
    "developer": "alona.danutsa@skyboxsecurity.com",
    "failure": "None",
    "pipeline_id": "89012",
    "merge_request_id": "52904",
    "job_name": "source_to_target",
    "start_time": "2024-11-25T10:25:43.576133+02:00",
    "success": "true"
}
```


- GET test_yalee/_search

Certifications

- .crt = public certificate = public key signed by a trusted **Certificate Authority (CA)**
  - used to identify your machine to clients.
  - proves the identity of the machine or service and is used to establish trust with clients that connect to it
- .key = private key
  - decrypt messages that were encrypted with the public key
- ca/ subdirectory
  - ca.crt = **public certificate** of the Certificate Authority (CA)
    - issuing digital certificates (like the `*.crt` file) after verifying the identity of the subject.
    - Clients must trust the CA to trust your server's certificate.
  - root.pem = **root certificate** of the Certificate Authority (CA)
    - **highest authority** in a certificate chain.
    - used to **sign** other certificates (like intermediate certificates and server certificates) and acts as the ultimate **trusted authority**.
  - .cer = .crt
    - It could be an **intermediate certificate**, which is used to create a chain of trust between your server's certificate (`{machine name}.crt`) and the root certificate (`ca.crt`).
    - ensures that a client can trace the server certificate back to a trusted root certificate.

Certificate	Server Side	Client Side
Server certificate (`{machine name}.crt`)	Present on the server to identify it.	Typically provided by the server during the handshake. The client verifies it using trusted CA certificates.
Private key (`{machine name}.key`)	Present on the server to decrypt and sign data.	Not needed on the client side.
CA certificates (`ca.crt`)	Optionally used for client authentication or verifying client certificates.	Optionally needed to verify the server's certificate or trust an intermediate CA.

Intermediate certificates (`something.cer`)	Optionally included to form the full certificate chain.	Optionally included to validate intermediate certificates in the chain.

Root certificate (`root.pem`)	Optionally used for verifying client certificates (in mutual TLS).	Must be trusted by the client to verify the server's certificate.

- **Server Side**:
  - The **server certificate** and **private key** are essential for identifying and securing communication.
  - The **CA certificates** and **intermediate certificates** are needed if client authentication or certificate chains are involved.
- **Client Side**:
  - The **root certificate** is critical to trust the server's certificate.
  - **Intermediate certificates** and **CA certificates** may be required to establish a full trust chain from the server's certificate.

The actual production update:

- ssh root@
  - sb-es-node-06
  - sb-es-node-07
  - sb-es-node-08
  - sb-es-node-09
  - sb-es-node-10
- in each:
  - ./stop_es.sh
  - docker stop metricbeat
  - ./rm_es.sh
  - nano run_es.sh
    - change version to 8.16.1
  - ./run_es.sh
- Test access
  - https://sb-elasticsearch/
    - elastic
    - https://sb-pum.il.skyboxsecurity.com/SecretServer/app/#/secrets/1805/general

Kibana

- ssh root@sb-elasticsearch (https://sb-pum.il.skyboxsecurity.com/SecretServer/app/#/secrets/2917/general)
- docker stop kibana
- docker rm kibana

- docker run -d --name "kibana" --volume "/root/kibana/kibana.yml:/usr/share/kibana/config/kibana.yml:ro" --volume "/etc/localtime:/etc/localtime:ro" --restart "unless-stopped" --publish "5601:5601" "docker.elastic.co/kibana/kibana:8.16.1"
- Test access
  - https://sb-kibana/ - same user and password as elastic

Heartbeat

- ssh root@sb-es-08
  - heartbeat directory with
    - Docker file (devoptools/docker/heartbeat)
    - heartbeat.yml (devoptools/docker/heartbeat + change passwords)
      - elastic - https://sb-pum.il.skyboxsecurity.com/SecretServer/app/#/secrets/1805/general
      - devopsadmin -
    - ca
      - root.pem (from /opt/certs)
      - sb-es-nodes.crt (from /opt/certs)
      - sb-es-nodes.key (from /opt/certs)
  - build and push the new image
    docker build --build-arg HEARTBEAT_VERSION=8.16.1 -t docker-nexus3.il.skyboxsecurity.com/devops/heartbeat:8.16.1-secured . && docker push docker-nexus3.il.skyboxsecurity.com/devops/heartbeat:8.16.1-secured
- ssh root@sb-elasticsearch (https://sb-pum.il.skyboxsecurity.com/SecretServer/app/#/secrets/2917/general)
  - docker stop heartbeat
  - docker rm heartbeat
  - run the new image
    `docker run --restart=unless-stopped -d --name heartbeat docker-nexus3.il.skyboxsecurity.com/devops/heartbeat:8.16.1-secured`

Metricbeat

- ssh root@sb-es-08
  - metricbeat directory with
    - Docker file (devoptools/docker/metricbeat + change version)
    - metricbeat.yml (devoptools/docker/metricbeat + change passwords)
    - system.yml (devoptools/docker/metricbeat)
    - ca
      - root.pem (from /opt/certs)
      - sb-es-nodes.crt (from /opt/certs)
      - sb-es-nodes.key (from /opt/certs)
  - build and push the new image
    `docker build --build-arg METRICBEAT_VERSION=8.16.1 -t docker-nexus3.il.skyboxsecurity.com/metricbeat:8.16.1 . && docker push docker-nexus3.il.skyboxsecurity.com/metricbeat:8.16.1`
- Ansible to all computers