Forgejo with SSO

Virtualization and Cloud Computing
a.a. 2023/24

Authors:
Gabriele Dellepere – S4944557
Lorenzo Foschi – S4989646
Kevin Cattaneo – S4944382

Main challenges

01
Docker

02
NFS

03
Swarm

04
Registry

05
Traefik

06
PostgreSQL

07
Keycloak

08
Forgejo

09
Grafana

10
Prometheus

11
Loki

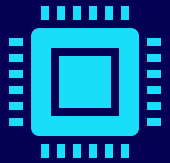
12
Promtail

...yes, **everything**

Oh I forgot... Vagrant

Lacks of pre-requisites

vagrant plugin install ***vagrant-sshfs*** command was needed before *vagrant up*



Tweaking hardware configuration

In order to start the giant we needed to set the right combination of RAM allocated to each VM

Vagrant bugs

#1

Default provider

Solution was to set ***DEFAULT_PROVIDER*** = *'vmware'* instead of *DEFAULT_PROVIDER = 'libvirt'* in Vagrantfile

#2

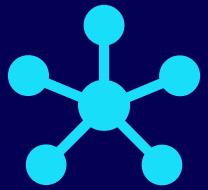
AudioAdapter

The box.ovf created by the VagrantFile crashes when Linux + Virtual Box are used on the incompatibility of the **AudioAdapter** (by default it is used the one of Windows)

Docker, Swarm, NFS

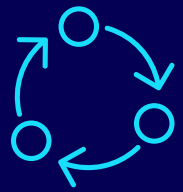
Refining the solution

There was a lot of work on trying to refine multiple times the solution while understanding the **Ansible syntax**, since these were the first touches



Diving in scalable solution

Some effort was involved when we delved into purple task: scalable solution, trying to understand who **loop** on what variable



Is really exported in that way?

While developing NFS tasks, we questioned about the workflow creation, export, update, and the necessity of **fstab**

Docker Registry #1

Pre-tasks

We needed to install **apache-utils** for htpasswd and also dependecies like **passlib** to make the tasks work

Permissions

We also spent some time searching for the optimal permission, in particular for the ‘0400’ of the **RSA private key** (only readable by owner and root)

Docker Registry #2

TLS

To enable TLS we added some environment variables that allowed us to refer to certificate and key, but we also needed to specifiy **tls: true** in the Run Registry task

Backfire of our work

Some fixes arrived after developing the following tasks, for example the **creation of CSR before** the creation of the self-signed certificate

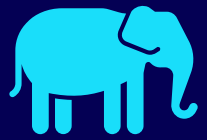
Swarm Services: Database / PostgreSQL

Which entrypoints?

One database for each service, it was clear; but a lot of time was wasted in order to understand we were required to touch the .sql and to create volumes

The dilemma of Postgre ‘role’

By looking to the logs we understood and loved the fact that Postgre uses different syntax for **role** and **user** creation



The elephant logo has a meaning

As we developed the following tasks we noticed the **slowness** of Postgres

Swarm Services: Traefik #1

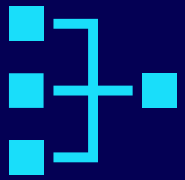
The right certificate…

By looking to the logs we discover a bug on the format of the certificate: we needed a **.pem** instead of .cert

…in the right directory

We also added **/etc/ssl/traefik** to create the proper directory for placing and locate the newly created certificates

Swarm Services: Traefik #2



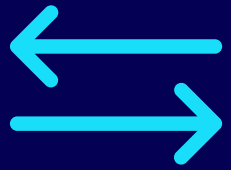
Overlay network

In order to allow the services to talk each other by passing through traefik, we put them in an **overlay_net**

The hosts file

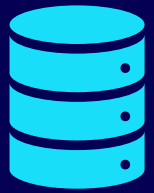
In order to make curl retrieving correctly data and make https services reachable, we wrote the right host (*.vcc.local) in a file, so that we can mount the **/etc/hosts** volume in the services that need it

Swarm Services: Keycloak #1



Indentation is important

Mostly when you talk about YAML, infact we had Keycloak db failure since our labels are referred to compose and not the swarm service



Removal of transactions

When we implemented the .sql files we also used **transactions** but this created some conflicts so we removed them

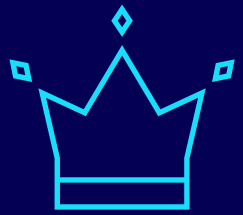
Swarm Services: Keycloak #2

Postgre permissions

We discovered that some permissions are not given to a user unless he is the **creator** of that db; so we scheduled first the creation of user and then of the db

Why so shy?

Keycloak was not visible on auth.vcc.local, that was caused by the unknown port to be contacted to access the service from the load balancer; so we specified it with: ***traefik.http.services.auth.loadbalancer.server.port=8080***



Creation of realm

The creation of the realm was one of the most difficult task: we then succeeded only when the problem occured in the successive tasks. Until that our SSO was not working

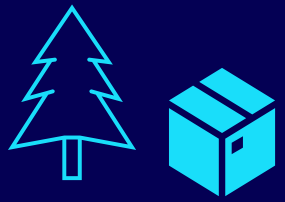
Swarm Services: Forgejo

Update ca-certificates

Was an open debate until we found out that Forgejo is an alpine distribution and so we wrote the right way of installing the command to update the certificates

Forcing the wait

We had also some db problems related to the slowness of Postgres, so we **forced some sleep** during the Forgejo process



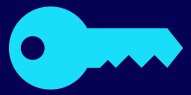
Alpine container

Just during developing phase we also created an alpine container to better understand the default package installer of the distribution and also if the services were reachable

Swarm Services: Grafana #1

Unclear documentation

Online there is **unclear documentation** about Grafana, so trying to master the interface has been challenging



OAuth

The major difficulty was enabling the SSO on Grafana and the whole problem can be translated to choosing the **right environment variables**

Other OAuth problems

- For Grafana: missing the line **export GF_SERVER_ROOT_URL =https://mon.vcc.local**

- For Forgejo: the **--name** option at the end of entrypoint.sh calls was missing

Swarm Services: Grafana #2

Journal the container

Due to a bit unclear documentation we couldn't manage to start the container: we used **journalctl -u docker.service** founding out that we were mounting the wrong volume

Dashboards

Dashboards took a lot of time, since they need to show the monitoring. Also it was all done by hand because **no template was suitable**

Swarm Services: Grafana #3

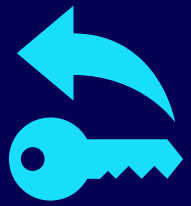
SSO Grafana as a real admin

To access with Keycloak's SSO in Grafana as Admin users (so being able to create dashboards accessing with SSO) we had to retry a lot of realms. Then we found the right realm configuration and the right '**attribute role**' string syntax.

|= error let us discover that...

We weren't selecting the proper UIDs for data sources (Loki and Prometheus) in dashboard json's, linking with the manually created ones in the data sources YAML

Swarm Services: Prometheus



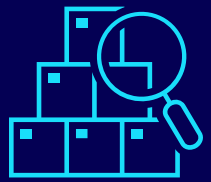
OAuth reverse proxy

The major difficulty has been how to create the reverse proxy to allow Prometheus to be accessed with Keycloak

Scraping registry metrics

Accessing metrics from registry was hard. We added **dockerswarm_sd_configs** at the end of prometheus.yml, with a **port:5051** replacement (the one that we opened in task 12 as DEBUG PORT for metrics)

Swarm Services: Promtail, Loki, Fluent-bit



State your purpose

Overall they have not been so difficult to develop apart the understanding about which was the **purpose** of each service in our scenario

Loki's default UID/GID

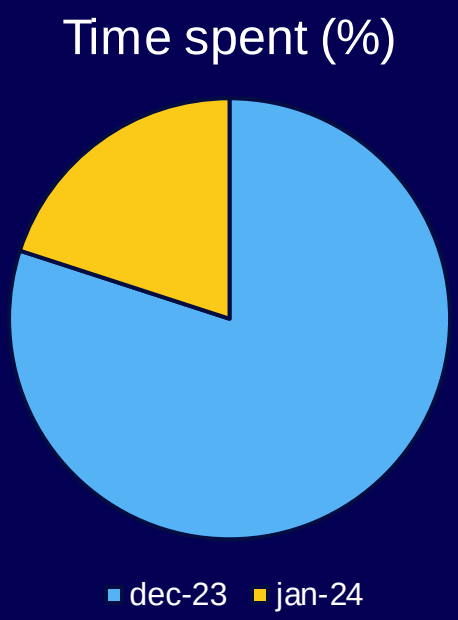
Loki uses as default UID/GID = 10001 that lacks of mounting permissions. We decided not to leverage the user to root, so we specify the mounting with the **right directories** while having the permissions

Some more details

We can also confirm that:

- Our system works when we **scale up** to N_TARGET > 2
- We committed all the suggestion provided by **Ansible Lint**, so our scripts follow the good-programming manners for Ansible
- Our privacy-critical files have been **ansible-vaulted**

Time spent for the project



First month

December 2023 – 80%

Second month

January 2024 – 20%

18 days

spent from the 20 December to 7 January

Thanks!

Credits for presentation template to **Slidesgo** and **Freepik** for infographics