# INCEPTION NOTES

*avilla-m@student.42roma.it*

**I. PROJECT SCOPE : SUBJECT & CORRECTION** **2**
KEYWORDS FROM SUBJECT 2
QUESTIONS ASKED ON CORRECTION SHEET 3
RESOURCES FROM OTHER STUDENTS - TUTORIALS AND GITHUBS 3
**II. RESEARCH : LIST OF RESOURCES** **4**
SYSTEM ADMINISTRATION 4
NETWORKS 4
APPLICATION DESIGN ARCHITECTURE 4
CONTAINERS 4
DOCKER 5
NGINX 6
MARIADB 7
WORDPRESS AND PHP-FPM 7
RECAP OF CONCEPTS AND RELATIONSHIPS 7
**III. NOTES** **8**
DOCKER COMMANDS - ONE PAGER RELEVANT TO PROJECT SCOPE 8
VIDEO: APPLICATION DESIGN ARCHITECTURE - from monolithic to microservices 9
MONOLITHIC 9
MULTI-TIER 9
MICROSERVICES 9
VIDEO: WHAT IS DOCKER 11
DIFFERENCE BETWEEN DOCKER AND VIRTUAL MACHINES 11

# I. PROJECT SCOPE : SUBJECT & CORRECTION

*= need to be able to explain all of them*

## KEYWORDS FROM SUBJECT

System Administration
infrastructure
application
services (vs application)

container
virtual machine
Docker
Docker vs Virtual Machine
Docker images
Docker-compose
Dockerfiles
DockerHub
latest tag
Alpine Linux vs Debian Buster
Dockerfiles best practices
Docker-network

Volumes
entrypoint
ports
daemons

NGINX
static website
TLSv1.2 vs TLSv1.3
WordPress + php-fpm
MariaDB
Data bases
PID 1
cache

**forbidden**
tail -f
network: host
--link
links:
sleep infinity

# QUESTIONS ASKED ON CORRECTION SHEET

***Before starting evaluation (script available in git repo):***

```
docker stop $(docker ps -qa);
docker rm $(docker ps -qa);
docker rmi -f $(docker images -qa);
docker volume rm $(docker volume ls -q);
docker network rm $(docker network ls -q) 2>/dev/null
```

# not given but important

```
sudo rm -rf /home/$(USER)/data/wp_db/* /home/$(USER)/data/wp_files/*;
```

**The evaluated person has to explain to you in simple terms:**

**Project overview**

- How Docker and docker-compose work ?
- The difference between a Docker image used with docker-compose and without docker-compose
- The benefits of Docker compared to VMs
- The pertinence of the directory structure required for this project (an example is provided in the subject's PDF file)

**Simple set-up**

- Show NGINX can only be accessed by port 443 only
- Show a SSl/TLS certificate is used *(show through browser)*
- Show Wordpress is properly installed and configured
- Show no ready-made image has been used

**Docker Network**

- Show that a user-defined network is visible
- The evaluated student has to give you a simple explanation of docker-network

**Wordpress - PHP-FPM - Volumes**

- Show that there is a Volume
- Add a comment with the available wp user
- sign in with the admin account, edit a page, check on the website the page has been updated
  *to sign in to wordpress panel* https://avilla-m.42.fr/wp-login.php *or* https://www.avilla-m.42.fr/wp-login.php
  *make sure there is both* `127.0.0.1 www.avilla-m.42.fr` *and* `127.0.0.1 avilla-m.42.fr` *on /etc/hosts*

**NGINX - SSL/TLS**

- Show that you cannot access the service via http
- Show that a TLS v 1.2/v 1.3 has been used (in .conf)
- Explain why it may not be recognized
- Explain what is a self signed certificate

**MariaDB - Volumes**

- Show that there is a Volume

```
docker volume ls
docker volume inspect [DATABASE_NAME]
```

- Explain how to login to the database

```
docker exec -it mariadb sh
mysql -u [username] -p[password]
```

- Try to login into the SQL database as root without a password, it should not work

```
mysql -u root -p$MYSQL_ROOT_PASSWORD
```

- Try to login into the SQL database as user with password, check that database is not empty

```
mysql -u WP_DB_USER -pWP_DB_PASSWORD
SHOW DATABASES;
SHOW TABLES FROM wp_db;
SHOW TABLE STATUS FROM wp_db;
```

**Persistence**

- reboot VM and launch docker-compose again, check everything is functional, wp and mdb are configured
- Changes done to wp (comments) should still be there

*useful wpCLI commands*

| | |
|---|---|
| show user list | wp user list |

useful *shell*

| | |
|---|---|
| install sudo | apk add sudo |
| create a new user | sudo adduser avilla-m |
| add sudo privileges | sudo adduser avilla-m sudo |
| check current user | whoami |
| change user | su avilla-m |
| check user list | cat /etc/passwd |
| add writing rights | chmod +w /etc/hosts |

# RESOURCES FROM OTHER STUDENTS - TUTORIALS AND GITHUBS

GitHub - vbachele/Inception: School 42 inception project. Project + tutorial + bonus in my read.me
GitHub - Forstman1/inception-42: 🖥️ Docker images with services such as MariaDB, Nginx, WordPress, Redis, FTP-server, Adminer and cadvisor in virtual...
GitHub - raccoman/inception: 42Cursus | This project aims to broaden your knowledge of system administration by using Docker

# II. RESEARCH : LIST OF RESOURCES

*Must-reads*

## SYSTEM ADMINISTRATION

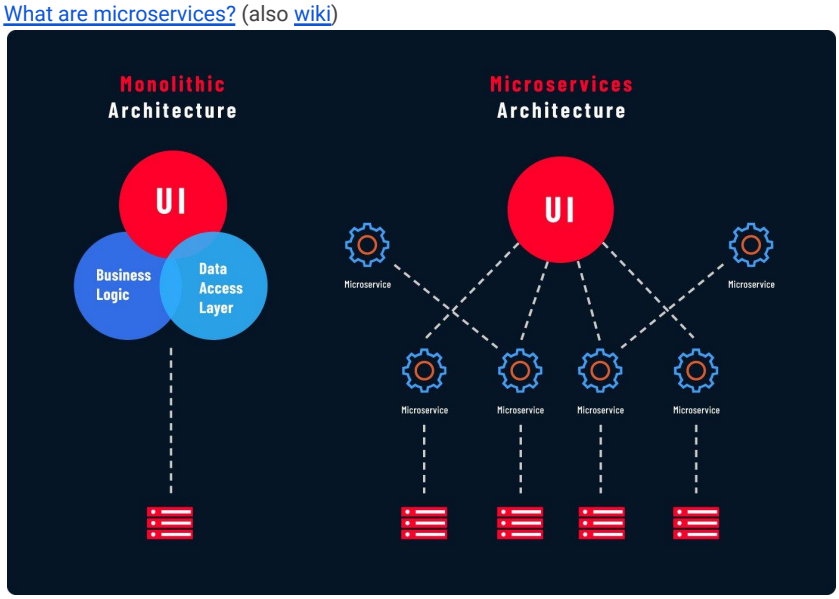
What is System Administration?
What is IT infrastructure?
What is an application?
What are the differences between services and applications?
📺Regarder la playlist de Cookie connecté sur les concepts clefs autours de l'infrastructure IT

## NETWORKS

What is networking ?
What is the server-client model ?
What is a proxy server ?
What is a reverse proxy server ?
What is caching ?
What is the difference between a web server vs. an application server: ?
What is a dynamic web page ? What is the difference between a static website and a dynamic website ?
What is a REST API? (and here)
📺Watch overview of backend web development

## APPLICATION DESIGN ARCHITECTURE

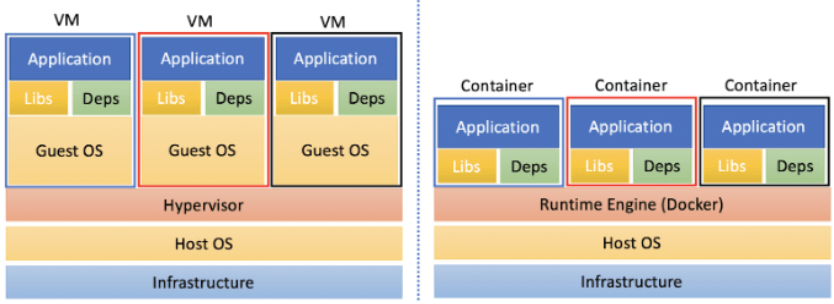
What are microservices? (also wiki)




What is DevOps ? (also wiki)
📺Watch TechWorld with Nana playlist around DevOps

## CONTAINERS

📺Watch IBM playlist on microservices, virtualization, containerization etc
📺Regarder la playlist de Cookie connecté sur les concepts clefs autours de Kubernetes et Docker
What is virtualization ?
What is containerization ?
What are hypervisors ?
What are containers?
What are virtual machines ?
What is the difference between containers and virtual machines ?




What is container orchestration ?
What is Kubernetes ?
Why do we need containers ? The benefits of using containers.

# DOCKER

Docker Crash Course for Absolute Beginners [NEW] TechWorld with Nana - 1h
Regarder une formation plus détaillée en français - Docker formation de A à Z - YouTube - Playlist 37 videos

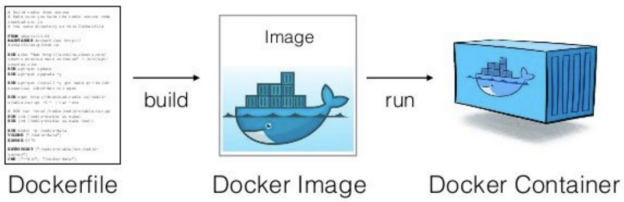
**Docker overview**

What is docker? (also wiki and official docker introduction)
How does docker work? What are the steps to create a container ? How are images built ? What are the basic commands of Docker ?

- Docker reference documentation
- https://www.freecodecamp.org/news/the-docker-handbook/
- https://docs.docker.com/get-started/
- https://www.digitalocean.com/community/tutorials?q=%5BDocker%5D

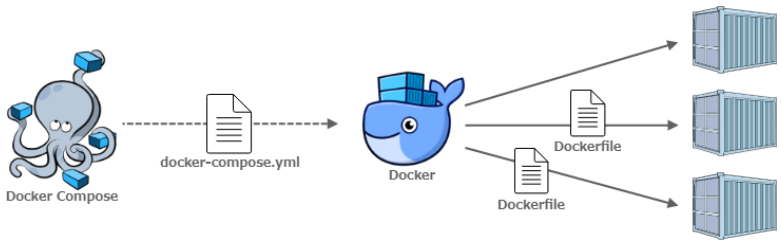
Understand differences between **images**, **layers** and **container** (also here and here)

**Steps to create one container**



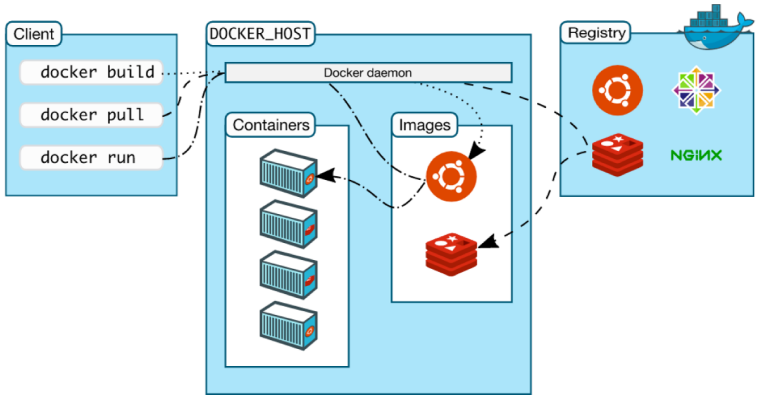

Dockerfile is based on a pre-existing image and add layers to that image to create a new one with the relevant configuration to suit our project/app/service
Image is a static 2D capture of the state of an environment / Container is an interactive 3D virtualization of that environment

**Docker-compose vs Dockerfile**




Docker-compose will handle docker to create the desired architecture of the infrastructure, it will create all objects and run the containers

How does processes work with Docker ?
Docker stop container => Docker daemon sends SIGTERM to **PID 1**, children stop and return SIGCHILD w/ exit status. Difference btw shell and exec format
What are **daemons** ? The client and daemon communicate using REST API calls.
What is the difference between a daemon process and a standard process ?
What is the difference between Alpine Linux and Debian Buster?
What is **Docker CLI**? (command line interface that runs docker commands)




**Volumes / Persistent storage**

What are the different methods of handling persistent data with Docker ?
What are **bind mounts** ?
What is the difference between using a bind mount and using **Docker Volumes** ?

**Networks**

What are Docker Networks ?
How does networking work with Docker ? tutorial
How to configure **ports** ?
List of TCP and UDP port numbers

**Configure Docker**

How to write a Dockerfile ?
What are best practices for writing Dockerfiles ? (go further with multistage builds )
What is a Docker-compose file ?
Which languages are used in Dockerfiles and Docker-compose ? Docker-compose : YAML overview, YAML syntax

**Use Docker**

see full recap of commands below -> Inception - List of Docker commands / or enter docker in terminal / docker --help

# NGINX

NGINX is a **lightweight**, high-performance web server designed for **high-traffic use cases**.
The most common use cases are HTTP cache at scale, load balancing, and reverse proxy.
What makes NGINX stand apart is its capability to serve **static content** such as HTML and Media files effectively.
It uses an **event-driven model** to provide predictable performance even when the load is high: rather than creating new processes for each web request, Nginx uses an asynchronous, event-driven approach where requests are handled in a single thread.
With Nginx, one master process can control multiple worker processes. The master maintains the worker processes, while the workers do the actual processing. Because Nginx is asynchronous, each request can be executed by the worker concurrently without blocking other requests.
The location of all NGINX configuration files is in the `/etc/nginx/` directory. The primary NGINX configuration file is `/etc/nginx/nginx.conf`.
To set NGINX configurations, use:

- **directives** - they are NGINX configuration options. They tell NGINX to process actions or know a certain variable, such as where to log errors.
- **Blocks** (also known as contexts) - Groups in which Directives are organized

What is a **web server** ? (also this and wiki)
What is the difference between a **HTTP server** and a web server

**What is NGINX ?**

https://kinsta.com/knowledgebase/what-is-nginx/
https://en.wikipedia.org/wiki/Nginx
https://www.nginx.com/resources/glossary/nginx
https://nginx.org/en/docs/
https://nginx.org/en/docs/beginners_guide.html

What are the main differences between **NGINX** and **Apache** ?
What is the difference between **NGINX** and NODE.JS  What is Node.js ?

**How to configure NGINX ?**

https://www.linode.com/docs/guides/how-to-configure-nginx/
https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
https://www.digitalocean.com/community/tools/nginx
https://github.com/10up/nginx_configs
https://www.nginx.com/resources/wiki/start/topics/examples/full/

What is **FASTCGI** and how to configure it? (proxying protocol used by Nginx to process PHP requests)
What does **URI** means ? Difference vs **URL/URN**?
What is **regex** match ? and here to see nginx configuration regex examples
Debug : check /var/log/nginx/error.log
List of HTTP status codes

**How to use NGINX within Docker**

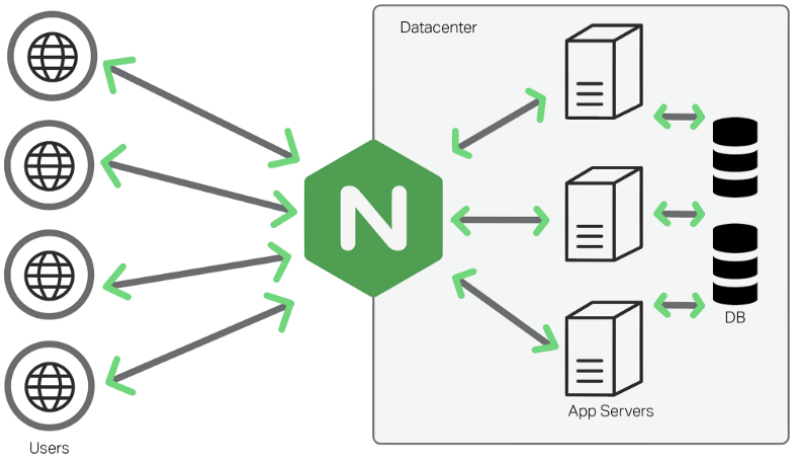
https://docs.nginx.com/nginx/admin-guide/installing-nginx/installing-nginx-docker/
https://www.baeldung.com/linux/nginx-docker-container
DOCKER : hébergez vos sites web avec NginX
Why use daemon off / How to run Nginx within a Docker container without halting?

**What is SSL ?**

What is an **SSL certificate** ? Handshake protocol, TLSv1.2 vs TLSv1.3, Transport Layer Security, Secure Socket Layer
https://en.wikipedia.org/wiki/Transport_Layer_Security#SSL_1.0,_2.0,_and_3.0
https://www.digicert.com/what-is-ssl-tls-and-https

**How to configure SSL certification and key ?**

https://codingwithmanny.medium.com/configure-self-signed-ssl-for-nginx-docker-from-a-scratch-7c2bcd5478c6
https://www.digicert.com/kb/ssl-support/openssl-quick-reference-guide.htm
Why a self-signed certificate is not recognized outside of local environment ?
Why use self-signed certificate ?




*avilla-m@student.42roma.it*

# MARIADB

Databases Explained in 5 Minutes - Relational and NoSQL Databases
What is MariaDB ?
NoSQL database

**Configuration**
Connecting to MariaDB
How to Install and Start Using MariaDB on Ubuntu 20.04 - Cherry Servers
How to Connect to MariaDB.

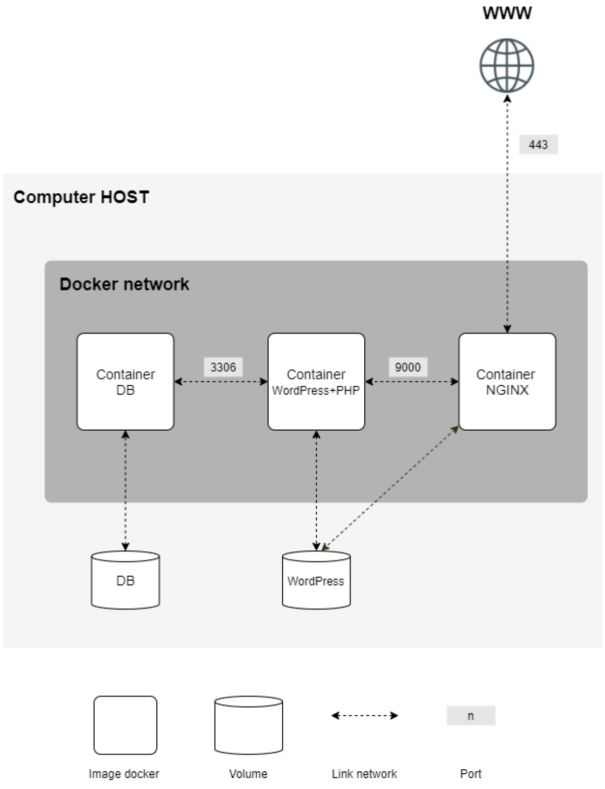
**Using Mariadb**
SHOW DATABASES - MariaDB Knowledge Base
SHOW TABLE STATUS - MariaDB Knowledge Base
How can I access my docker maria db? - Stack Overflow

# WORDPRESS AND PHP-FPM

**What is WordPress ?**
Open source **content management platform** (CMS) used for creating websites, blog sites, and even apps.
Written in hypertext preprocessor language (PHP) and paired with a MySQL or MariaDB database with supported HTTPS.

**What is php ?**
PHP is a **general-purpose scripting language geared toward web development**. PHP was originally an abbreviation of Personal Home Page, but it now stands for the recursive initialism PHP: **Hypertext Preprocessor.** PHP code is usually **processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable**. On a web server, the result of the interpreted and executed PHP code – which may be any type of data, such as generated HTML or binary image data – would **form the whole or part of an HTTP response**. Various web template systems, web content management systems, and web frameworks exist which can be employed to orchestrate or facilitate the generation of that response.
**Interpreter**: In computer science, an interpreter is a computer program that directly executes instructions written in a programming or scripting language, without requiring them previously to have been compiled into a machine language program.
**PHP-FPM** (FastCGI Process Manager) **is an interpreter of PHP on the server side**

# RECAP OF CONCEPTS AND RELATIONSHIPS




| | |
|---|---|
| Nginx | **webserver**, intermediary between client and server |
| HTTP/HTTPS | Hypertext Transfert Protocol, **request-response protocol** |
| SSL/TLS | **cryptographic security protocol** to regulate access |
| HTML | Hypertext Markup language. **web content** is written in html |
| PHP | PHP is a server-side scripting language embedded in HTML, **connect w/ db** |
| PHP-FPM | FastCGI Process Manager. **PHP interpreter**, processes PHP requests, **protocol** as proxy between client and server for HTTP response |
| Wordpress | **content management platform** (CMS) |
| MariaDB | database |

**Recap on HTTP protocol**
What is HTTP ? The Hypertext Transfer Protocol (HTTP) is an application layer protocol design within the framework of the Internet protocol suite model. HTTP functions as a **request–response protocol in the client–server model**. A web browser, for example, may be the client whereas a process, named web server, running on a computer hosting one or more websites may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body.
HTTP is a stateless application-level protocol and it requires a reliable network transport connection to exchange data between client and server.[19] **In HTTP implementations, TCP/IP connections are used using well known ports** (typically port 80 if the connection is unencrypted or port 443 if the connection is encrypted, see also List of TCP and UDP port numbers).
Data is exchanged through a sequence of request–response messages which are exchanged by a session layer transport connection.[19] An **HTTP client initially tries to connect** to a server establishing a connection (real or virtual). **An HTTP(S) server listening on that port accepts the connection** and then **waits for a client's** request message. The **client sends its request** to the server. Upon receiving the request, the **server sends back an HTTP response** message (header plus a body if it is required). The body of this message is typically the requested resource, although an error message or other information may also be returned. At any time (for many reasons) client or server can close the connection. Closing a connection is usually advertised in advance by using one or more HTTP headers in the last request/response message sent to server or client.[21]

**Ports**

| | |
|---|---|
| 3306 | MySQL protocol |
| 9000 | PHP-FPM |
| 443 | HTTPS |
| 80 | HTTP |

# III. NOTES

## DOCKER COMMANDS - ONE PAGER RELEVANT TO PROJECT SCOPE

Tutoriel Docker, l'essentiel pour débuter

management commands : liste les groupes de commande (compose, container, image, network, volume etc)
commands : commandes qu'on peut exécuter seules, sans groupe (build, run, exec, ps, infor etc)

**docker** — list all docker commands
- docker **info** — display system-wide information
- docker **ps** — list all active containers
- docker **ps --all/-a** — list all containers
- docker **build** [path] — build an image from a dockerfile (-t container_name to define a name) (. for current dir) (-t to tag the img)
- docker **run** — create and run a container from an image ()
- docker **rm** [cont_id] — removes one or more container (name, full or partial id, return name/id to confirm removal)
- docker **rmi** — removes one or more image (name, full or partial id, return name/id to confirm removal) (only if image not used)
- docker **inspect** OBJECT_NAME
- docker **system prune** remove unused images, containers and networks

**docker image** — list all docker image commands
- docker image **ls** — (=docker images)
- docker image **prune**
- docker image **rm** IMAGE_NAME (= docker rmi)
- docker **history** IMAGE_NAME

**docker container** — list all docker container commands
- docker container **ls**
- docker container **prune**
- docker container **rm** CONTAINER_NAME
- docker container **port** CONTAINER_NAME [PRIVATE_PORT[/PROTO]]
- docker container **inspect** / **logs** CONTAINER_NAME
- docker container **start** / **restart** / **pause**/ **unpause** / **stop** / **kill** CONTAINER_NAME
- docker container **exec** [OPTIONS] CONTAINER_NAME COMMAND [ARGUMENTS]
- docker container **attach** CONTAINER_NAME *(running container)*
- **docker run -it –rm** CONTAINER_NAME **bash**
- **docker exec -it** CONTAINER_NAME **/bin/bash** *(running container)*

**docker network** — list all docker network commands
- docker network **ls**
- docker network **inspect**
- docker network **prune**
- docker network **create** / **connect** / **disconnect** / **rm** NETWORK_NAME

**docker volume** — list all docker volume commands
- docker volume **ls**
- docker volume **inspect** VOLUME_NAME
- docker volume **prune**
- docker volume **create** / rm VOLUME_NAME

**docker-compose**

| | |
|---|---|
| docker-compose -f srcs/docker-compose.yml **up -d** | build if img don't exist + start in detached mode |
| docker-compose -f srcs/docker-compose.yml ps | lists the containers |
| docker-compose -f srcs/docker-compose.yml **up --build** | forces to build even if not needed |
| docker-compose -f srcs/docker-compose.yml **build** | only builds the images, does not start the containers |
| docker-compose -f srcs/docker-compose.yml **stop** | stop |
| docker-compose -f srcs/docker-compose.yml **down** | stop + remove all objects |
| docker-compose -f srcs/docker-compose.yml **down --volume --rmi all** | |
| sudo rm -rf /home/$USER/data | delete all enerated data |
| docker system **prune -a -f** | remove all unused objects |
| sh srcs/requirements/tools/configure.sh | sets-up data directories |

**RUN OPTIONS**

| | |
|---|---|
| -a | attach to stdin/out/err |
| -d | detach, run container in background and print container id |
| -e | set environment variables |
| -i | interactive mode (keep stdin open even if not attached) |
| -t | create a simile terminal |
| -p | publish port [host]:[container_port] |
| -P | publish all exposed ports |
| -v | bind mount a volume |
| -rm | remove when exits |

**USEFUL**

| | |
|---|---|
| running a container and attaching it to the terminal : | **docker run -it –rm** CONTAINER_NAME **bash/sh** |
| getting in interactive mode (putting the container into foreground) : | **docker attach** CONTAINER_NAME |
| getting out of interactive mode (putting back the container into background) : | **ctrl + p, ctrl + u** |
| **-f** | **force an operation** |
| display only user-defined networks | **docker network ls --filter type=custom** |

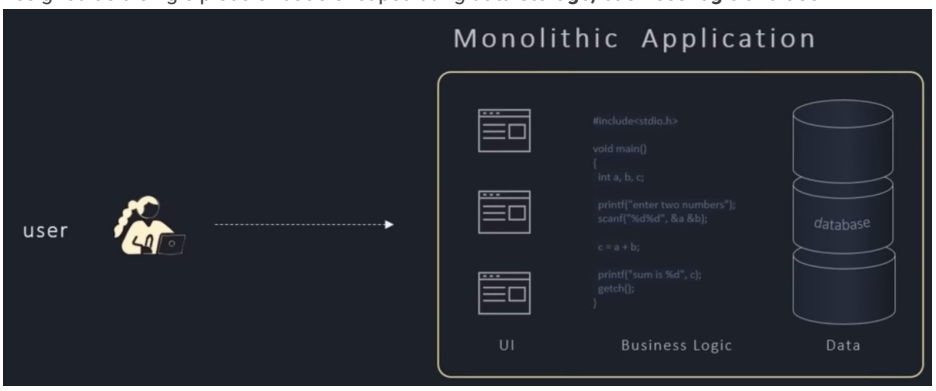
**BASH OR SH**

# VIDEO: APPLICATION DESIGN ARCHITECTURE - from monolithic to microservices

Microservices Explained in 5 Minutes

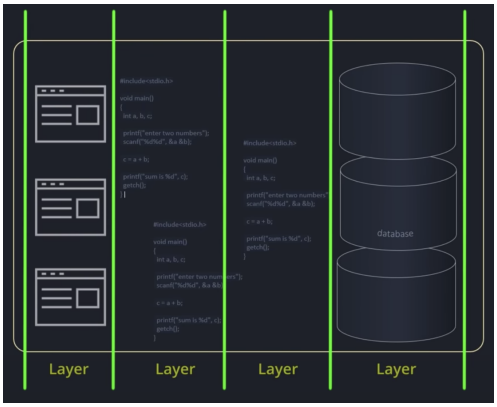
## MONOLITHIC

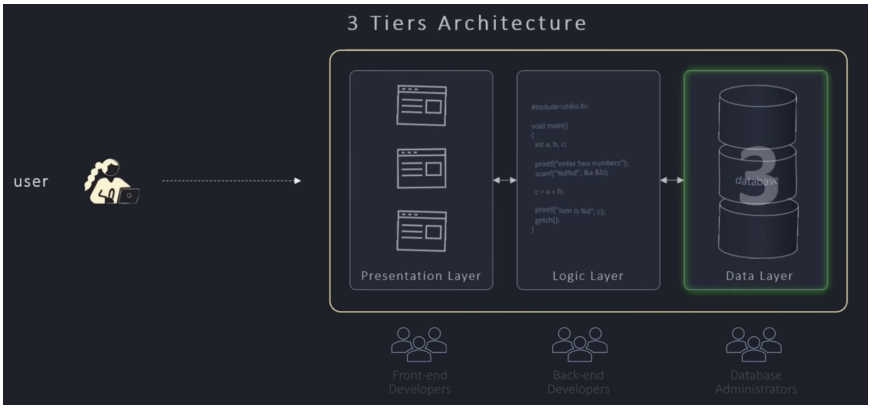
Designed as a single piece of code encapsulating **data storage, business logic and user**




Everything is tangled together : difficult to maintain, evolve, scale

## MULTI-TIER



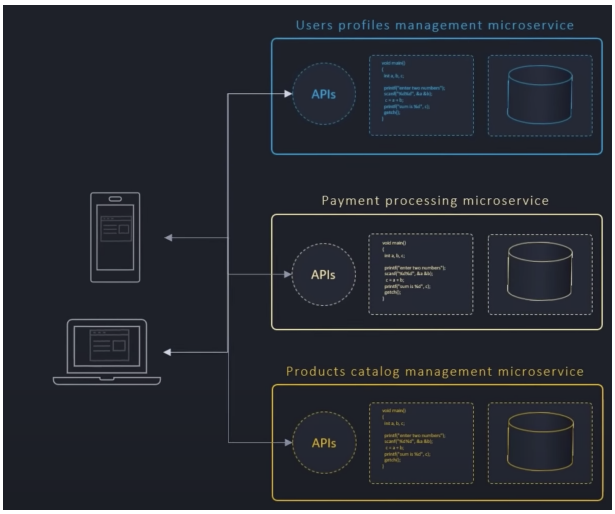




still a centralized way to design applications

## MICROSERVICES

### BENEFITS

- Addresses the limitations and drawback of complex applications
- Every microservice deals with one business function end to end, independently from other microservices
- Have simple, easy to understand API, communicate with each other through lightweight common protocol such as HTTP or message queues
- Enables teams to works independently and more efficiently as they don't rely on each other
- Theoretically teams could use diverse programming languages and deploy to different infrastructure even though it's more efficient on all aspects to use the same language and same infrastructure
- communication between microservices
    - synchronous com: via API calls -> HTTP requests
    - asynchronous: message broker (rabbitMQ)
    - service mesh




### CHALLENGES

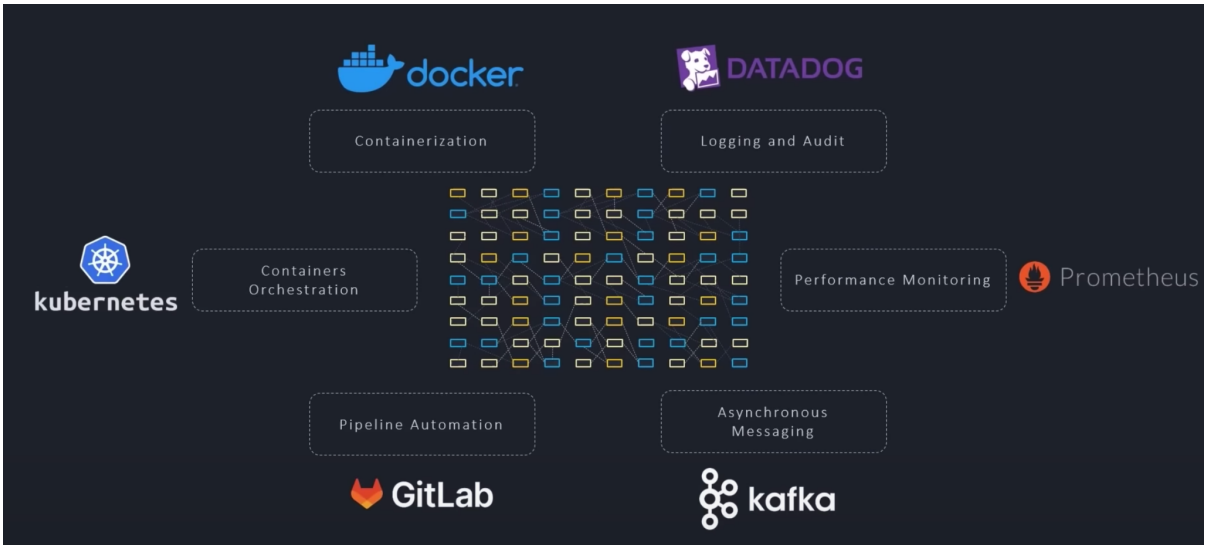
*avilla-m@student.42roma.it*

As applications continue to grow, the number of microservices inside an organization can increase to a point where complexity is yet again too high and makes the system inefficient and too hard to maintain. if a microservice fails it could be very hard to troubleshoot.

New tools to manage that problem :

- Containerization : helps deploy microservices in a minimalist self-contained runtime (**docker**)
- Container orchestration systems : manage containers life cycles (**kubernetes**)
- Pipeline automation : CI-CD (**Gitlab**)
- Asynchronous messaging : further the couple's microservices by providing message brokers and queues (**kafka**)
- Application performance monitoring tools : track microservices performance (**Prometheus**)
- Login and audit tools : help keep track of everything happening within the system (**Datadog**)

Monorepo vs Polyrepo(Gitlab groups)



# VIDEO: WHAT IS DOCKER

What is Docker in 5 minutes

- software development platform
- a kind of virtualization technology
- allow to develop and deploy apps inside neatly packaged virtual containerized environments = apps run the same whenever wherever
- software stays system agnostic = simpler to use and less work to develop
- containers act like micro-computers
- own os, cpu processes, memory and network resources
- containers usually run one specific task
- a specific task can be MySQL database, NodeJS app,
- and are then networked together and could be scaled
- dockerhub : online cloud repository of docker containers

## DIFFERENCE BETWEEN DOCKER AND VIRTUAL MACHINES

- allow to run many docker containers while only a few vm can be run because of limited space
- vm has to quarantine off a set amount of resources, HDD space, RAM memory, CPU processes power, emulate hardware and boot an entire operating system
- vm communicate with the host computer via a translator application (HYPERVISOR) running on the host operating system
- Docker: resources are shared directly with the host
- Docker communicates natively with the system kernel, bypassing the middleman on linux machines, win 10, win serv 16+ = you can run any version of linus in a container and it will run natively
- Docker uses less disk space as it's able to reuse files efficiently by using a layered file system. multiple docker images that use the same base image will take info from one single copy of the files needed

## Dockerfile:

Dockerfile, a blueprint, very simple file that describes how the docker image will be built

FROM : select a base image from Dockerhub (os)

RUN commands

docker build -t myDockerImage

check that image has been built with : docker images

with a built image, can be used to run a container, or can be shared externally for others to use

to run a containers with an image created by other

pull the image + run the container

docker pull <image name> <tag to specify which version, latest if null>

docker run <options>

options : detached -d, -p assigning ports for web services

view running containers with 'docker container ls'

Docker Compose : control several containers as part of a single application

web server : Nginx

database server : MySQL