

Hardware and Software Requirements

UNCTAD - DMFAS Documents

Exported on 06/08/2023

Table of Contents

1	Introduction	4
2	Architecture Overview	5
2.1	Backend.....	5
2.2	API Gateway.....	6
2.3	Frontend	6
3	Hardware Requirements.....	7
3.1	Database Server	7
3.2	Services.....	8
3.2.1	Supported Installation Options.....	8
3.2.1.1	Docker Compose	8
3.2.1.2	Kubernetes	10

- [Introduction](#)(see page 4)
- [Architecture Overview](#)(see page 5)
- [Hardware Requirements](#)(see page 7)
 - [Database Server](#)(see page 7)
 - [Services](#)(see page 8)

Target release	Release 1
Epic	
Document status	DRAFT
Document owner	Marcelo Abalos ¹
Designer	Facundo Miquel ²
Developers	
QA	

¹ https://rtce-confluence.unog.ch/display/~marcelo_abalos

² https://rtce-confluence.unog.ch/display/~facundo_miquel

1 Introduction

The purpose of this document is to describe installation options for DMFAS 7 and the minimum recommended hardware and software requirements to run the application.

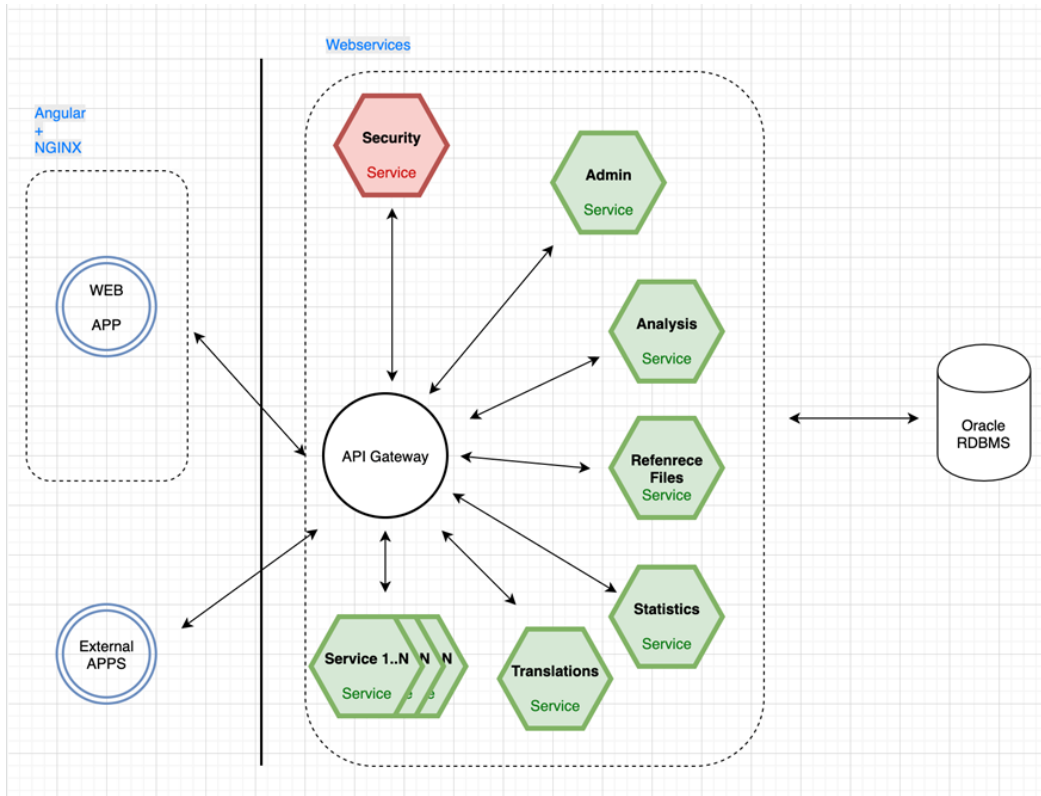
This document will be updated whenever necessary to consider changes and developments in information technology. Considering that DMFAS 7 is continuously evolving, some hardware or software requirements may change over time. Nevertheless, the DMFAS Programme will do as much as possible to maintain the foreseen hardware and software platform.

The precise requirements for each institution to run DMFAS 7 will depend on the specific situation of the institution (number of users, number of debt instruments, available connections etc.). Therefore, it is important to contact the DMFAS Programme **before** ordering equipment or software to ensure that the latest and most appropriate specifications are met.

2 Architecture Overview

The application is divided into three component groups, the persistence layer, comprised of the Database, the business layer or Backend composed of a collection of Java web services and the Frontend (presentation layer) which is an Angular Web Application.

The following diagram depicts the high-level architecture of the application and its components.



2.1 Backend

The Backend architecture is based on the "separation of concerns" principle, which states that every component should address one concern, so it is implemented by a collection of microservices

All of these microservices are independent of each other and communication between them happens using REST services.

This has the following advantages:

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities

Since all services are independent the actual microservices can be written in any programming language, it was decided that Java would be the Development Language of choice for the first round of services using Spring Boot (<https://spring.io/projects/spring-boot>) as the development stack.

2.2 API Gateway

The API gateway is the main glue that connects all the microservices together, all the traffic, either internal (from microservice to microservice) or external (from the web app or other applications to the microservices) will flow through the API, this allows for an extra layer of security being that the only exposed service is the API service. The gateway will publish the available routes and it will validate the required security attributes before the request reaches the backend service.

2.3 Frontend

The Frontend is developed using Angular ([http://\(see page 3\)angular.io](http://(see page 3)angular.io)³) as the default presentation language. Furthermore, it was originally decided that Primefaces would be the component library of choice (<https://www.primefaces.org/primeng/>), this component library has more than 80 native widgets and more than 50 user components that will facilitate the development of the frontend.

The Angular application is divided in 3 main modules:

Module Name	Description
core	holds core interceptors, services (http to the api-gateway, and other core local services), guards (authorization and authentication, etc.), constants, entity models and DTOs and some util functions
shared	holds all shared components, directives, guards and common validators. These stuff is used on all the different modules, simplify development and help maintain a single look & feel across the whole application
main application	The main application holding the login page, the main layout component, and all the modules that together make the application. These are lazy loading modules.

³ <http://angular.io/>

3 Hardware Requirements

3.1 Database Server

In general, the DMFAS system runs on every Operating System platform compatible with the latest Oracle release:

Oracle Release	Operating System	
19c	MS WS 2016	Linux/Unix/Solaris/AIX

Additional memory, processor speed and disk space might be needed if other software is installed. Hardware specifications for servers running another network operating system should have equal performance and capacity and must be compatible with Oracle RDBMS. It is recommended to have an automatic detection software for monitoring updates and proactive support notifications.

Component	Specifications
CPU ^[1]	Intel® / Core i(7/9) 9 th to 11 th Gen, 4 or more GHz Series Processor
Hard disk	3 x 480 GB or more SATA, NVMe, SSD or SAS 15K rpm with data striping through RAID X (Mixed drive types allowed matching type/speed/capacity)
DVD-ROM	DVD+/-RW
RAID/Internal Controller	PERC H(x)
Memory	From 32 GB up to 128 GB DIMM/LRDIMM/RDIMM
Screen/Video adapter	17" Flat Panel with integrated video card with 1GB or more
Backup streamer	Digital tape streamer with the same capacity as the total disk space
NIC	1 or more network card(s) supported by the network installed with capacity of 100/1000 Mbps for best performance
Computer in general	If the server is a stand-alone computer, it could be a tower model or suitable for placement in a rack. Both, with a scalable internal capacity and flexibility to adapt to changing workload conditions. Local technical support is strongly recommended.

[1] Any 100% Intel compatible processor, such as AMD can be used as well, given that they deliver equal or better performance.

3.2 Services

All services, including the Backend services, the Frontend service and the API Gateway, are deployed as Docker Containers, the deployment will need an internet connection to pull the containers from the UNCTAD container registry that holds all versions of the services.

3.2.1 Supported Installation Options









There are mainly two installation options, one simply uses Docker and Docker-Compose to install all the services in any machine running Docker, the other uses any Container Orchestration Tool, the sizing in this guide is based on Kubernetes because it is the supported tool, but any Docker orchestration tool will work (e.g. RedHat OpenShift, Nomad, etc.).

3.2.1.1 Docker Compose

This is the smallest option in terms of sizing and only requires one Server to run all the backend and frontend services, keep in mind that the database should always have it's on server.

Docker Version	Operating System	
>= 20.10	MS WS 2016	Linux/Unix/Solaris/AIX

The following distributions and architectures are currently supported for Docker installation

Platform	x86_64 / amd64	arm64 / aarch64	arm (32-bit)	s390x
CentOS ⁴				
Debian ⁵				
Fedora ⁶				
Raspbian ⁷				

4 <https://docs.docker.com/engine/install/centos/>

5 <https://docs.docker.com/engine/install/debian/>

6 <https://docs.docker.com/engine/install/fedora/>

7 <https://docs.docker.com/engine/install/debian/>

Platform	x86_64 / amd64	arm64 / aarch64	arm (32-bit)	s390x
RHEL ⁸				✓
SLES ⁹				✓
Ubuntu ¹⁰	✓	✓	✓	✓
W¹¹Indows	✓			

The following table represents the minimum requirements for a Production Server, note that it is recommended that the Database be in a separate server.

Component	Specifications
CPU Count	Minimum: 2; Recommended 4+. CPU Architecture should be chosen according to the table above.
Hard disk	1. 3 x 100MB or more SSD or SAS SSD with data striping through RAID Xa
Memory	From 16 GB up to 128 GB DIMM/LRDIMM/RDIMM
NIC	1 or more network card(s) supported by the network installed with capacity of 100/1000 Mbps for best performance
Computer in general	If the server is a stand-alone computer, it could be a tower model or suitable for placement in a rack. Both, with a scalable internal capacity and flexibility to adapt to changing workload conditions. Local technical support is strongly recommended.

Additional Storage

Some additional storage might be required in order to store persistent data, such as attachments. This storage will be mapped using an external Volume to Docker and must be on a shared

⁸ <https://docs.docker.com/engine/install/rhel/>

⁹ <https://docs.docker.com/engine/install/sles/>

¹⁰ <https://docs.docker.com/engine/install/ubuntu/>

¹¹ <https://docs.docker.com/engine/install/binaries/>

High Availability

In order to achieve high availability using the Docker installation Docker Swarn must be installed and running on at least two servers, this would increase the required number of server to at least 2, it is possible to use the Database Server as part of the Docker Swarn but this will require an additional 8GB of RAM and an additional vCPU on the Database server in order to accommodate the new workload.

Software Requirements

The following table lists the required software that must be installed in order to be able to complete the Application installation. Source

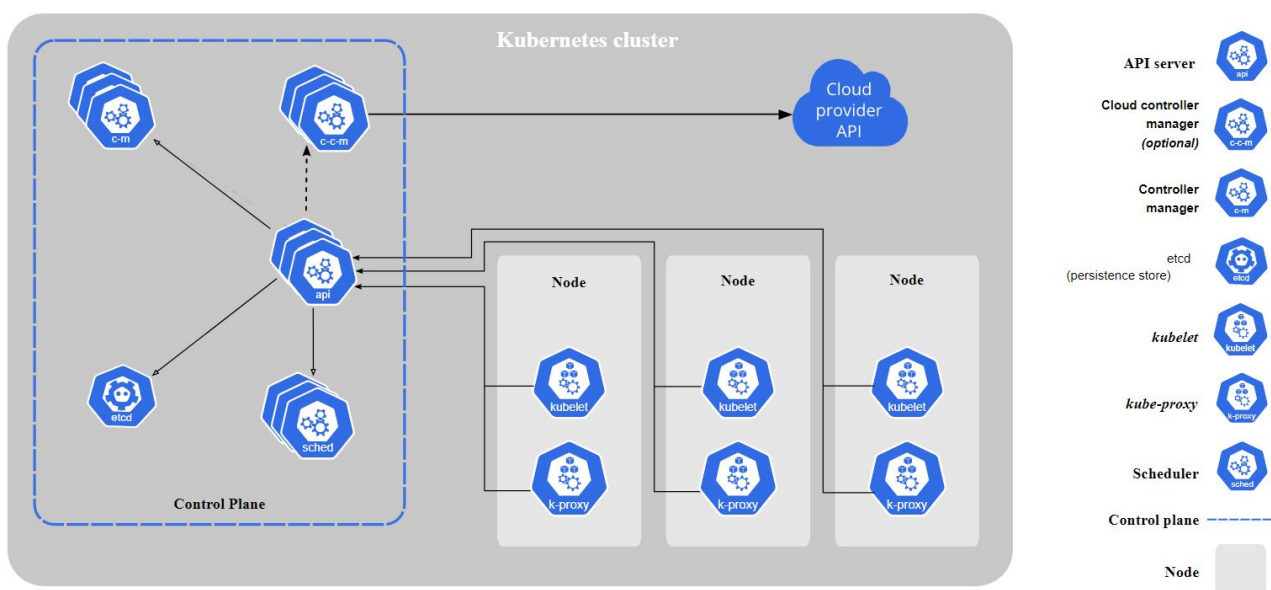
Software	Version	Description	
Docker	>= 20.10	Docker engine	https://docs.docker.com/engine/install/
git	latest		https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

3.2.1.2 Kubernetes

For bigger installations it is recommended that Kubernetes be used to deploy all the services, [Kubernetes](#)¹² is an open-source container orchestration platform for managing, automating, and scaling containerized applications, Kubernetes is the de facto standard for container orchestration because of its greater flexibility and scaling capabilities

The following diagram depicts the basic architecture of a Kubernetes cluster.

¹² <https://www.dynatrace.com/news/blog/what-is-kubernetes-2/>



source: <https://kubernetes.io/docs/concepts/overview/components/>

Each cluster is formed by one or more servers that hold the Control Plane Components (api server / etcd / scheduler / ...) and one or more servers that hold the running pods (Node Servers).

Each node server can hold a given number of pods, this number is determined by the amount of available Memory and Cores of each individual server. There is also some hard limits to the amount of pods that a server can run imposed by the Kubernetes architecture, but these numbers are high enough (no more than 110 pods per node, no more than 5000 nodes per cluster, etc.), so that they will not impact on the architecture of DMFAS 7.

If high availability is a requirement there should be at least two Control Plane servers and at least two Nodes for running pods in different zones, this ensures that on failure the pods from the other nodes will take the load.

Control Plane

Component	Specifications
CPU Count	Minimum: 4+. CPU Architecture should be chosen according to the table above.
Hard disk	1. 3 x 200 MB or more SSD or SAS SSD with data striping through RAID Xa
Memory	From 16 GB up to 128 GB DIMM/LRDIMM/RDIMM
NIC	1 or more network card(s) supported by the network installed with capacity of 100/1000 Mbps for best performance
Computer in general	If the server is a stand-alone computer, it could be a tower model or suitable for placement in a rack. Both, with a scalable internal capacity and flexibility to adapt to changing workload conditions. Local technical support is strongly recommended.

Node Server

Component	Specifications
CPU Count	Minimum: 4+. CPU Architecture should be chosen according to the table above.
Hard disk	3 x 200 MB or more SSD or SAS SSD with data striping through RAID Xa.
Memory	From 16 GB up to 128 GB DIMM/LRDIMM/RDIMM
NIC	1 or more network card(s) supported by the network installed with capacity of 100/1000 Mbps for best performance
Computer in general	If the server is a stand-alone computer, it could be a tower model or suitable for placement in a rack. Both, with a scalable internal capacity and flexibility to adapt to changing workload conditions. Local technical support is strongly recommended.

Single Instance Server

This is used in small instances where the application is installed on a single server

Component	Specifications
CPU Count	Minimum: 4+. CPU Architecture should be chosen according to the table above.
Hard disk	3 x 300 MB or more SSD or SAS SSD with data striping through RAID Xa.
Memory	From 24 GB up to 128 GB DIMM/LRDIMM/RDIMM
NIC	1 or more network card(s) supported by the network installed with capacity of 100/1000 Mbps for best performance
Computer in general	If the server is a stand-alone computer, it could be a tower model or suitable for placement in a rack. Both, with a scalable internal capacity and flexibility to adapt to changing workload conditions. Local technical support is strongly recommended.

Additional Storage

There is a number of options in terms of provisioning Storage Classes supported by Kubernetes, the selection of the provisioner is based on local expertise and preference, [here](https://kubernetes.io/docs/concepts/storage/storage-classes/#nfs)¹³ (<https://kubernetes.io/docs/concepts/storage/storage-classes/#nfs>) is a list of supported adapters.

¹³ <https://kubernetes.io/docs/concepts/storage/storage-classes/#nfs>

Software

The following table lists the required software that must be installed in order to be able to complete the Application installation.

Software	Version	Description	
Docker	>= 20.10	Docker Engine	https://docs.docker.com/engine/install/
kubectl		Kubernetes administrator	https://kubernetes.io/releases/download/
Kubernetes	>= 1.23	At least one master and one worker nodes should be installed and configured	
helm	~3.8.1	Tool used to deploy the application	https://helm.sh/docs/intro/install/
git	latest		https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
cert-manager	latest	Kubernetes Certificate manager used to manage and update digital certificates. This should be installed in the kubernetes container	https://cert-manager.io/docs/installation/
helm	>= 3.9.x	Helm charts are used to install all DMFAS components to the Kubernetes cluster.	https://helm.sh