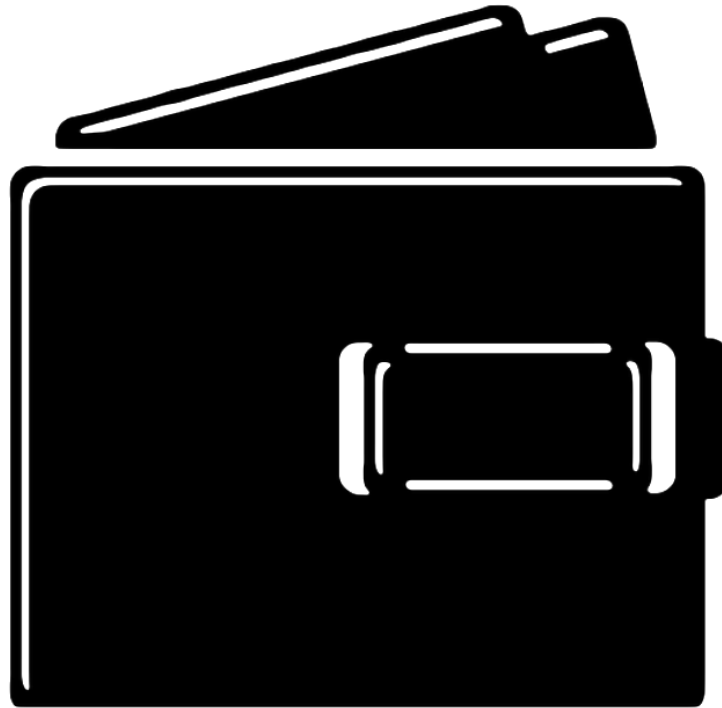


Case Study #4

Memi Lavi
www.memilavi.com





PayRawl

PayRawl

- Payment processing system
- Receives files from various sources
- Validates and processes the files
- Sends instruction files to banks
- Fully automatic, no UI





Requirements

Functional

What the system should do

1. Receive file to be processed
2. Validate and process the file
3. Work with various file formats
4. Perform various calculations on the file
5. Create bank payment file
6. Put the payment file in a designated folder
7. Keep log of all the activity for 7 years

Non-Functional

What the system should deal with



NFR - What We Ask

1. *"How many files per day?"* 500
2. *"How long should the process take?"* 1 min
3. *"What is the average size of a file?"* 1MB
4. *"Can we tolerate data loss?"* Absolutely Not!



Data Volume - Files

- 1 File = 1MB
- 500 files / day = 500MB / day
 - => ~182GB / year
 - => ~1.3TB / 7 years



Data Volume - Log

- Assuming each processing generates 500KB log data
- 500 files / day = 250MB log data / day
 - => ~91GB log data / year
 - => ~638GB log data / 7 years



Requirements

Functional

What the system should do

1. Receive file to be processed
2. Validate and process the file
3. Work with various file formats
4. Performs various calculations on the file
5. Create bank payment file
6. Put the payment file in a designated folder
7. Keep log of all the activity for 7 years

Non-Functional

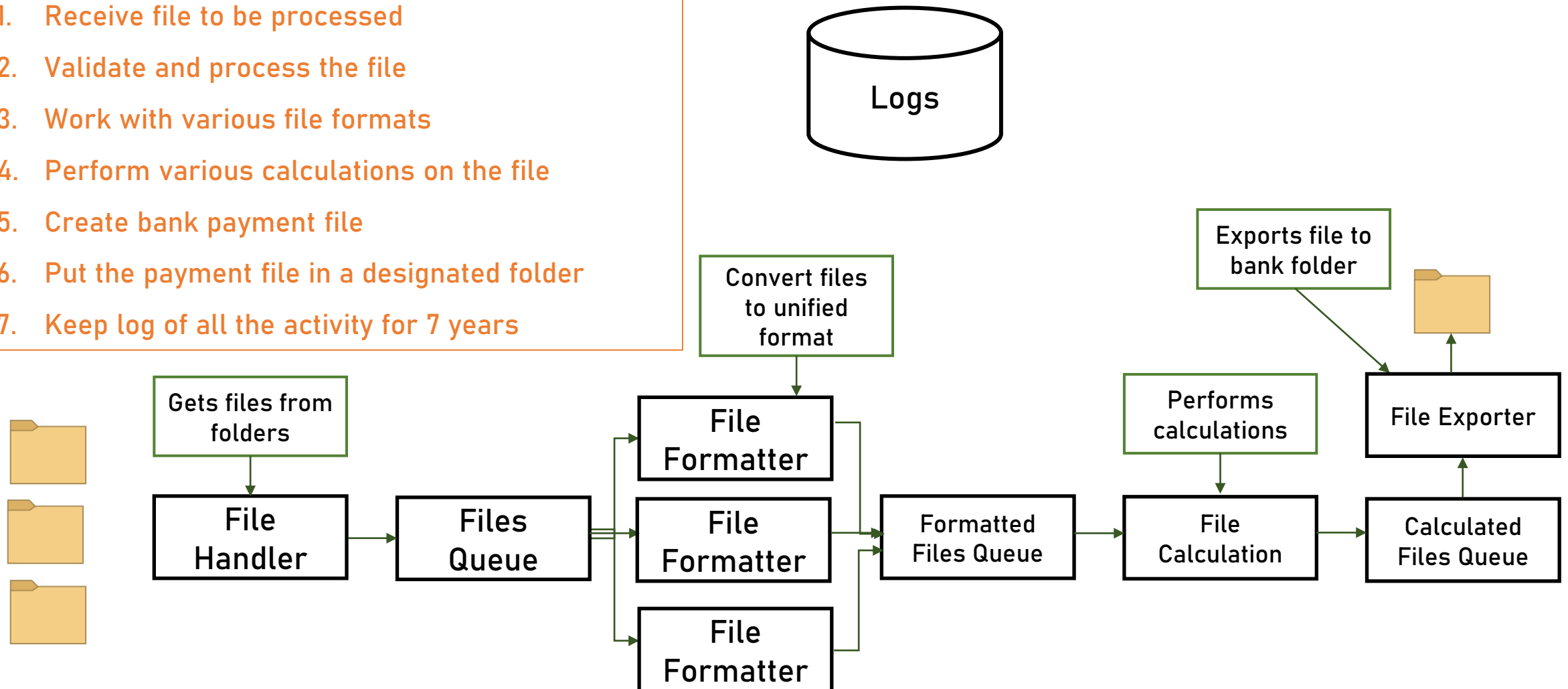
What the system should deal with

1. 500 files / day
2. No data loss
3. 1 min processing time
4. Activity log for 7 years
5. ~2TB / 7 years



Components

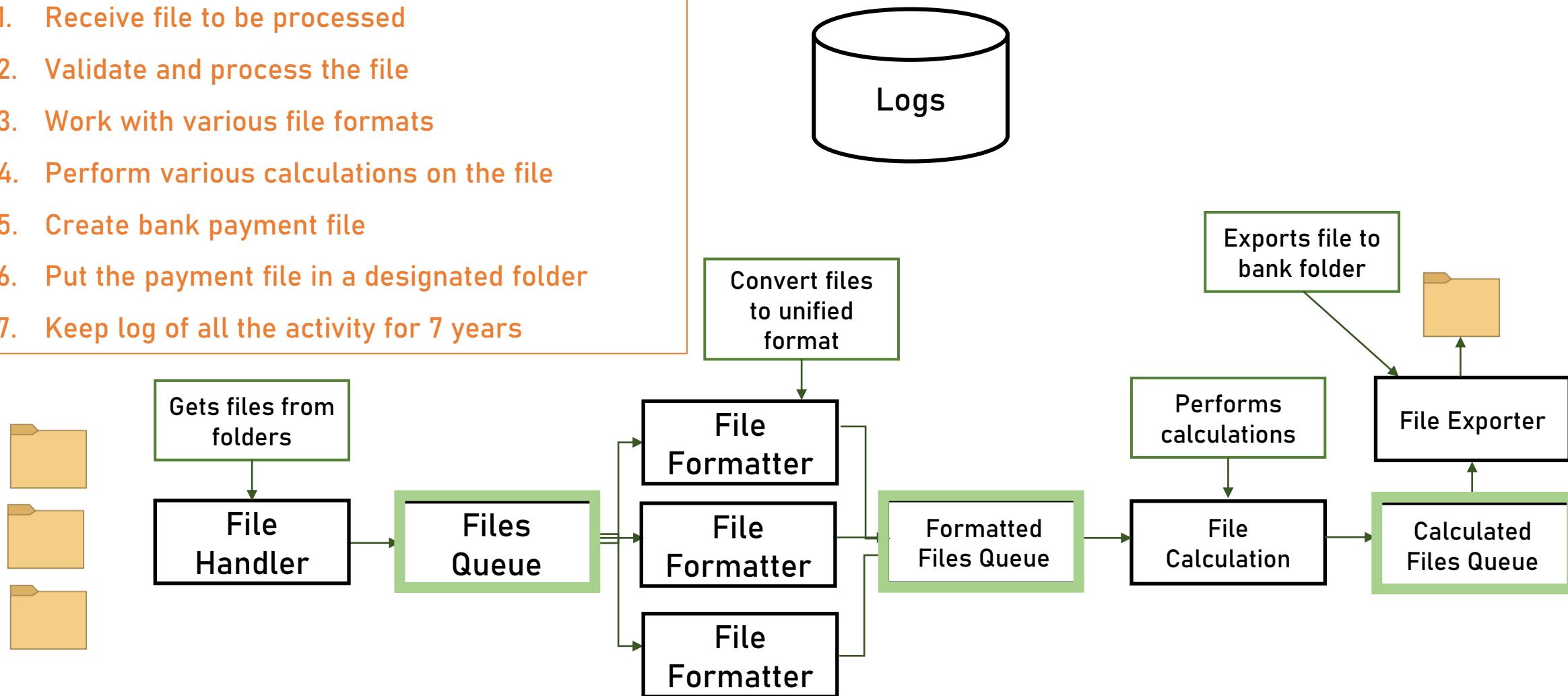
1. Receive file to be processed
2. Validate and process the file
3. Work with various file formats
4. Perform various calculations on the file
5. Create bank payment file
6. Put the payment file in a designated folder
7. Keep log of all the activity for 7 years





The Queue

1. Receive file to be processed
2. Validate and process the file
3. Work with various file formats
4. Perform various calculations on the file
5. Create bank payment file
6. Put the payment file in a designated folder
7. Keep log of all the activity for 7 years





The Queue

- Passes payloads from logic unit to another
- Balances load
- Persists messages (Durability!)



The Queue

- Asynchronous
 - Which is good since we don't have UI



The Queue

Which queue?



- General purpose
- Easy to setup
- Not suitable for streaming scenarios



- Great for streaming scenarios and high-load systems
- Complex to setup



The Queue

Which queue?

 RabbitMQ

- General purpose
- Easy to setup
- Not suitable for streaming scenarios



The Queue

- Queue is usually represented like this:

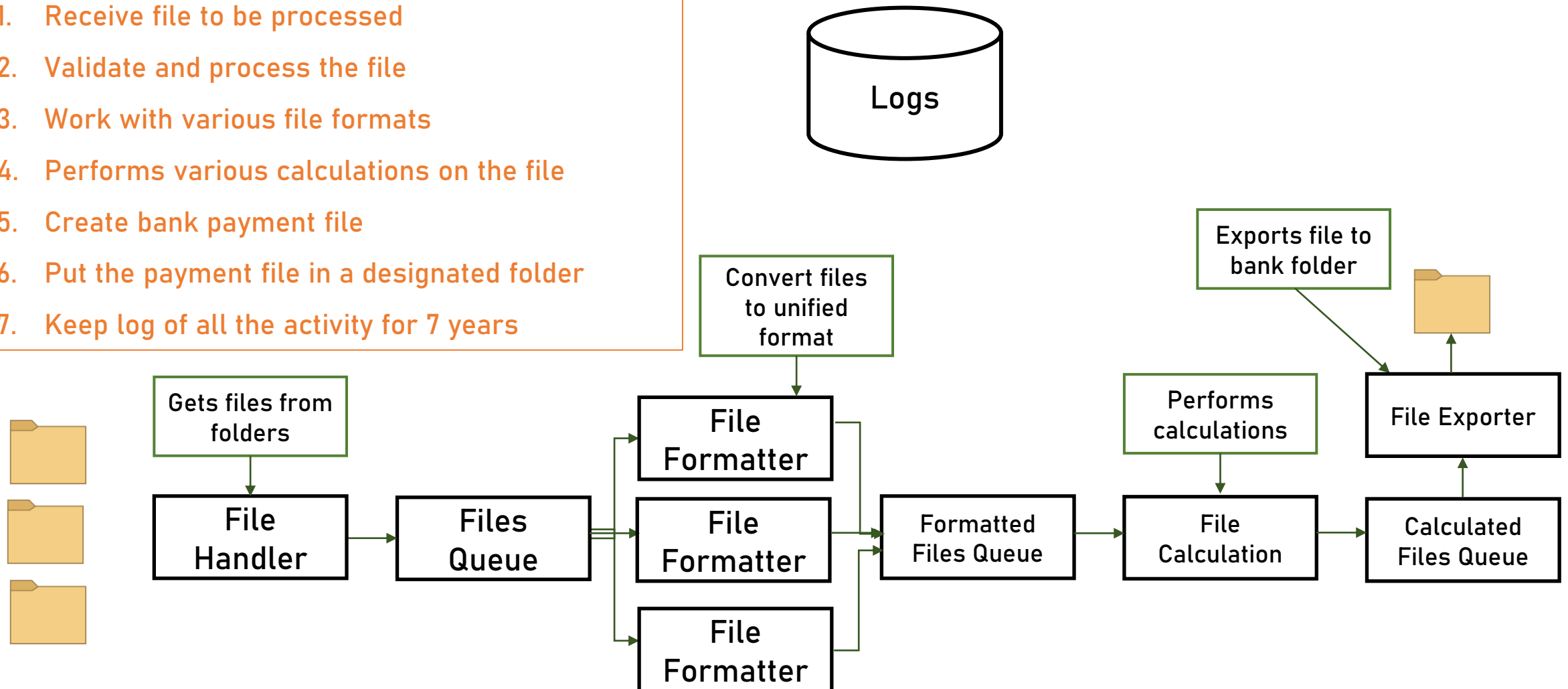


- So...




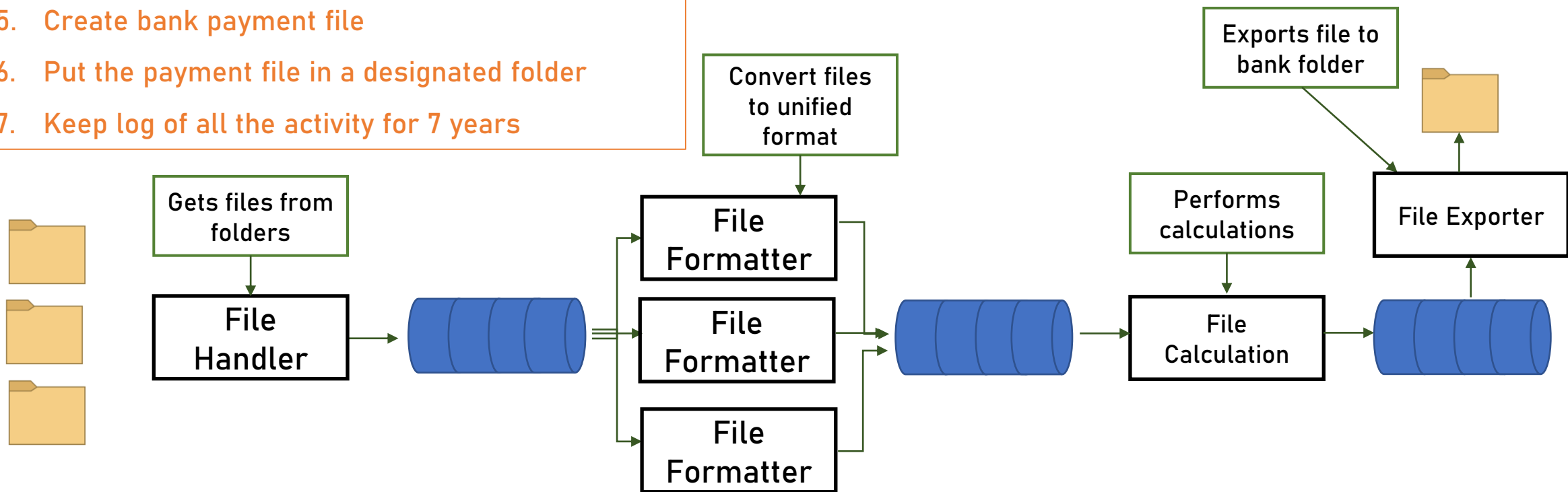
Components

1. Receive file to be processed
2. Validate and process the file
3. Work with various file formats
4. Performs various calculations on the file
5. Create bank payment file
6. Put the payment file in a designated folder
7. Keep log of all the activity for 7 years



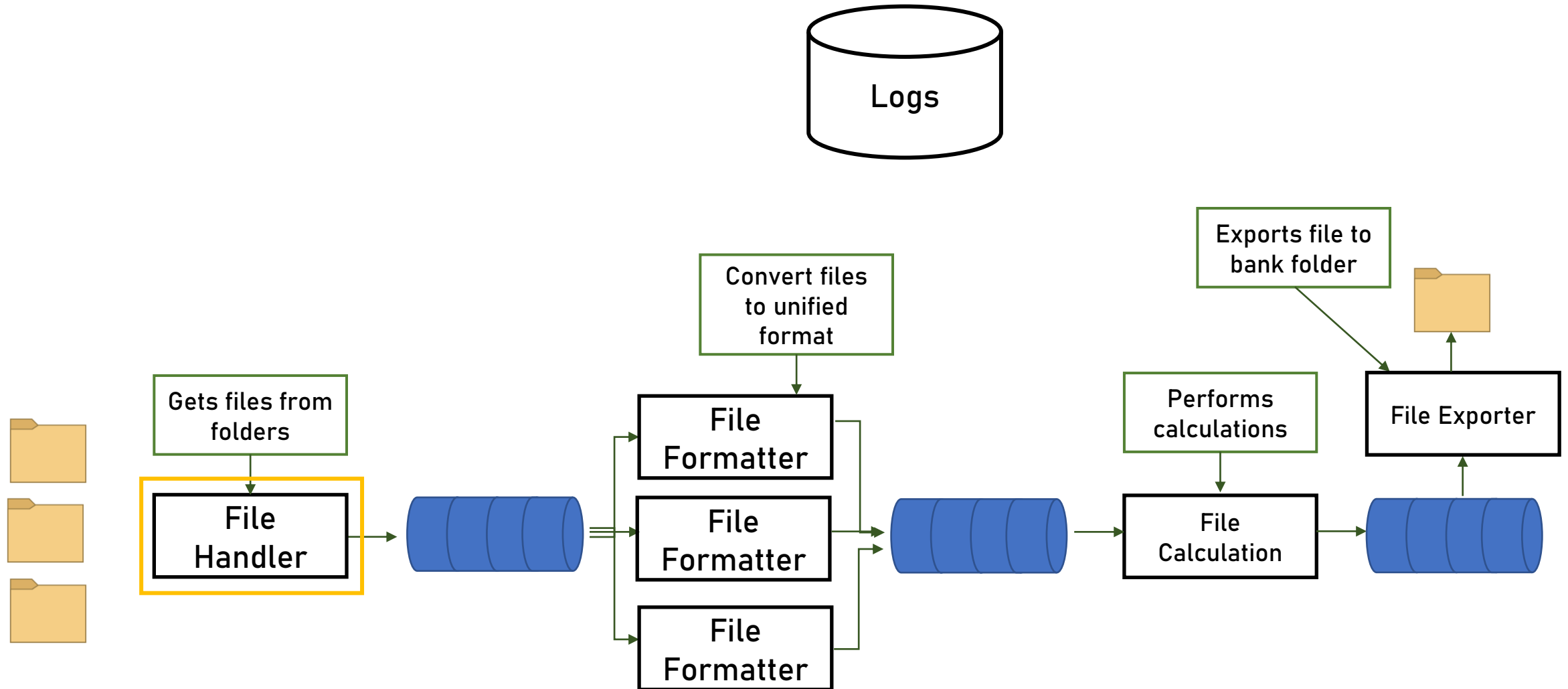


- 
- A cylinder labeled "Logs" representing a log database.





Components





File Handler

What it does:

- Pulls payment files from folders
- Put the files in the queue



Application Type

- Web App & Web API ✗
- Mobile App ✗
- Console ✓
- Service ✓
- Desktop App ✗



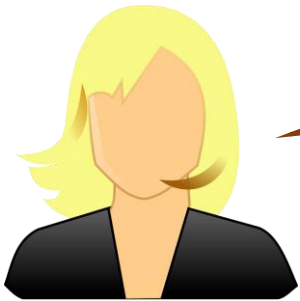
Technology Stack

Considerations:

- Should be able to pull files from folders
- Should be able to connect to queue
- Not much else...



Technology Stack



This is a brand new company, we don't have existing knowledge. What would you recommend?





Technology Stack

What we're looking for:

- Performance
- Community
- Cross Platform
- Easy to learn and use
- Evolving
- Great threading support



Technology Stack

Our candidates:





Technology Stack

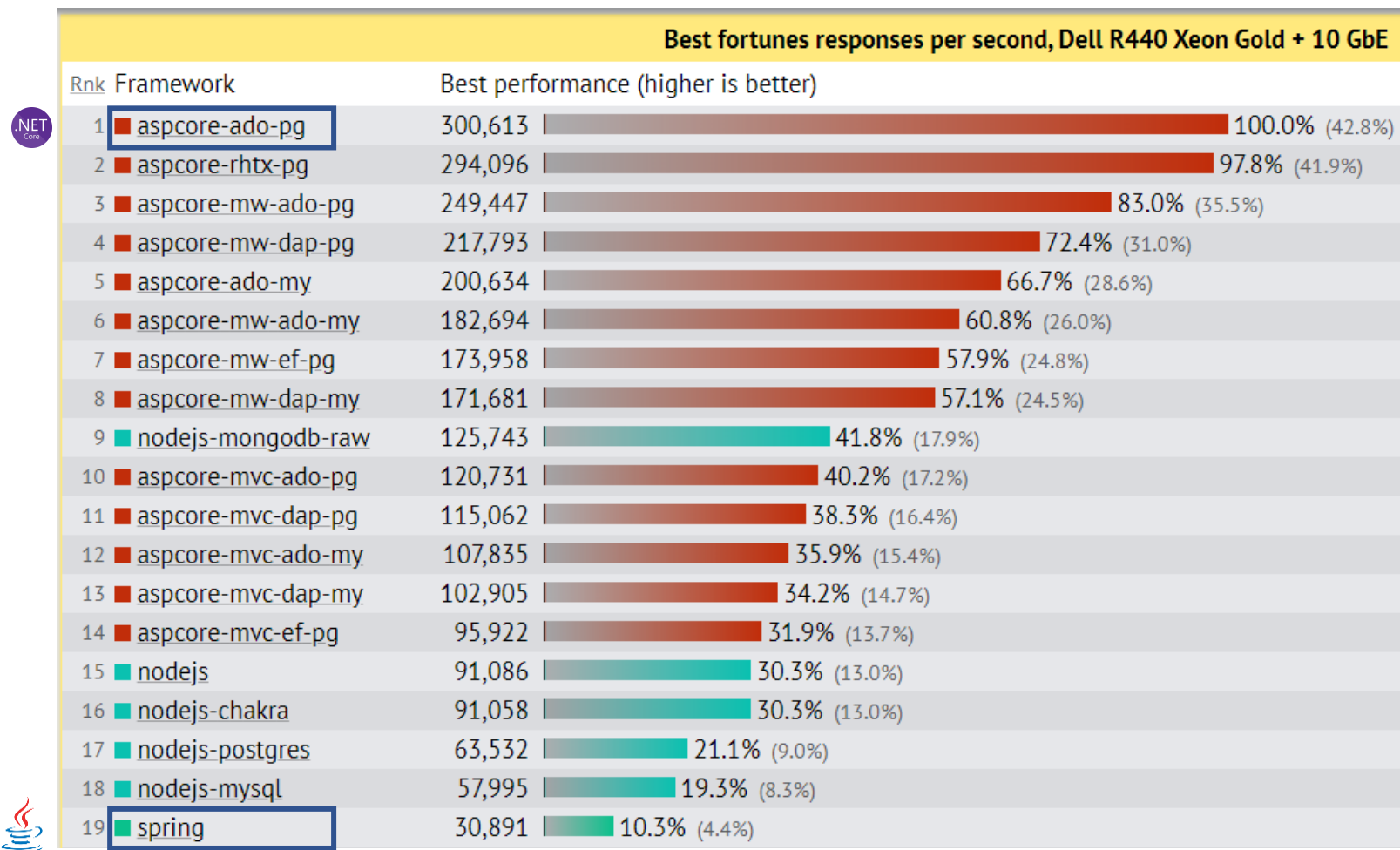
Node is mainly for web apps, our component is a service, so...





Technology Stack

Performance:



Source: <https://www.techempower.com/benchmarks/#section=data-r18&hw=ph&test=fortune&l=zik0ot-f&p=zik0zj-zijocf-zijocf-4atpfi>



Technology Stack

Community:

Jan 2020	Jan 2019	Change	Programming Language	Ratings	Change
1	1		Java	16.896%	-0.01%
2	2		C	15.773%	+2.44%
3	3		Python	9.704%	+1.41%
4	4		C++	5.574%	-2.58%
5	7	▲	C#	5.349%	+2.07%
6	5	▼	Visual Basic .NET	5.287%	-1.17%
7	6	▼	JavaScript	2.451%	-0.85%
8	8		PHP	2.405%	-0.28%

Source: <https://www.tiobe.com/tiobe-index/>



Technology Stack

Cross Platform:





Technology Stack

Ease to learn and use:





Technology Stack

Evolving?



Next versions planned until 2021



Roadmap announced until 2023





Technology Stack

Threading support:





Technology Stack - Decision





Architecture

Traditional:

User Interface /
Service Interface

Business Logic

Data Access

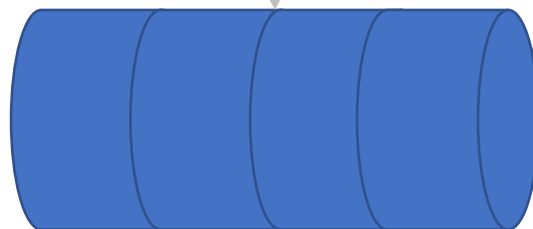
Data Store



Architecture

File Watcher

Topic is selected by file location





Redundancy

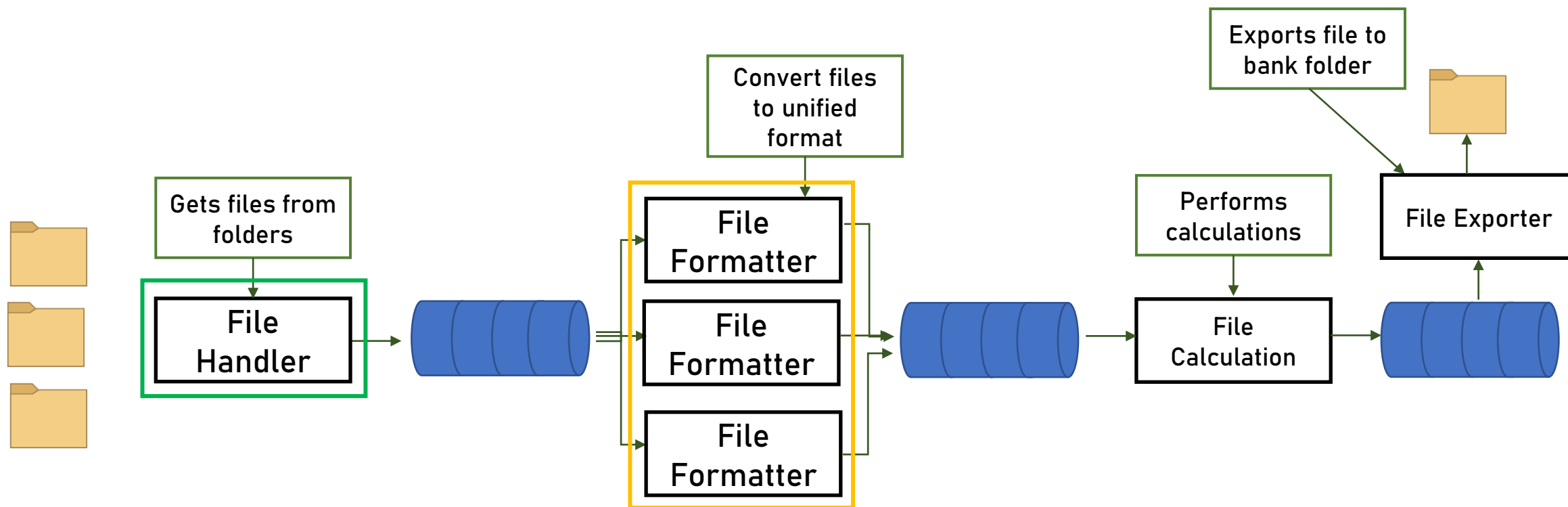
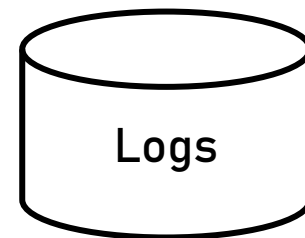
File Handler

Is Alive

File Handler



Components





File Formatter

What it does:

- Receives files from its specific topic
- Validates and formats the file to unified format
- Puts the new file in a queue
- New formatters will be developed for new file



Application Type

- Web App & Web API ✗
- Mobile App ✗
- Console ✓
- Service ✓
- Desktop App ✗



Technology Stack





Architecture

Queue Receiver

Business Logic





File Formatter Redundancy

Consumer Group

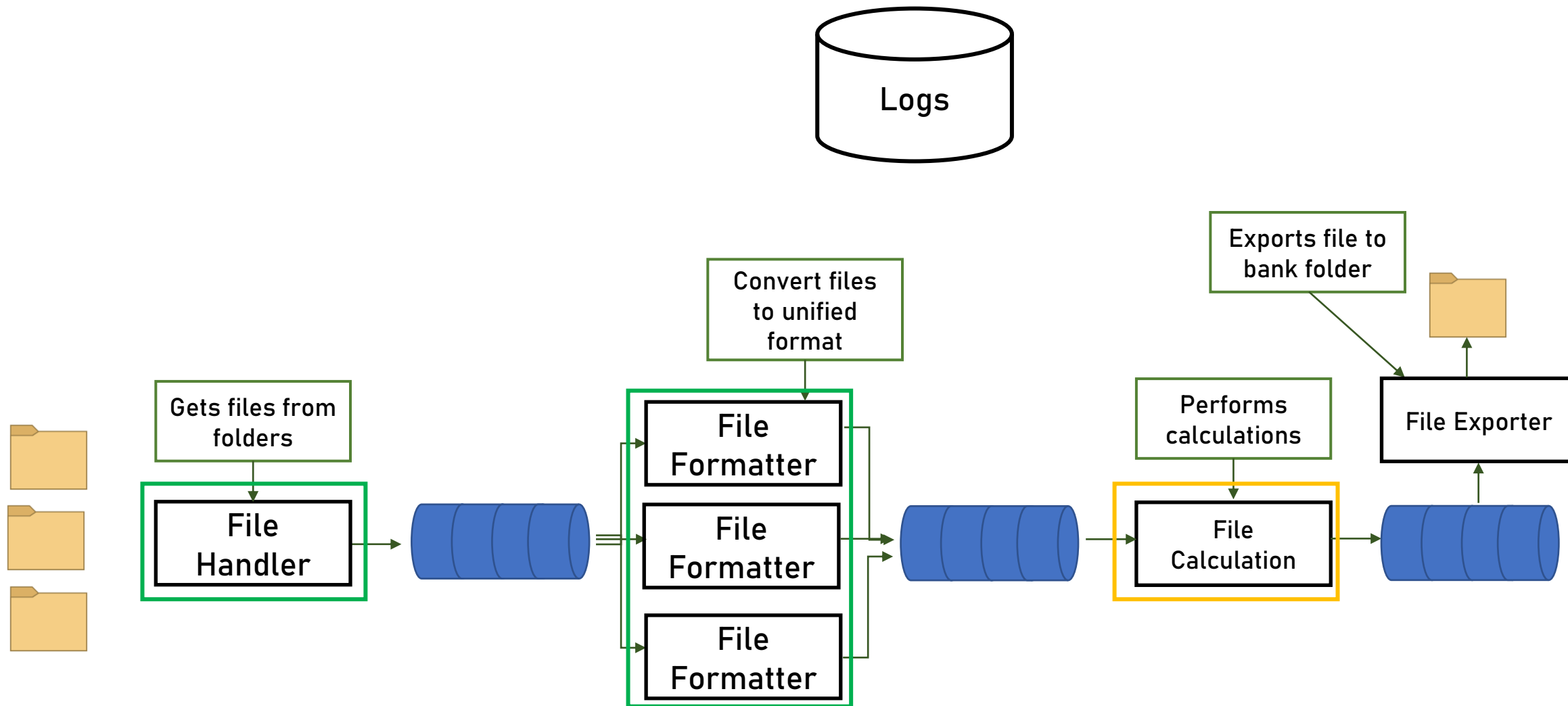
File Formatter

File Formatter

File Formatter



Components





File Calculation

What it does:

- Receives files from the queue
- Performs some calculations on the data
- Puts the new file in a queue



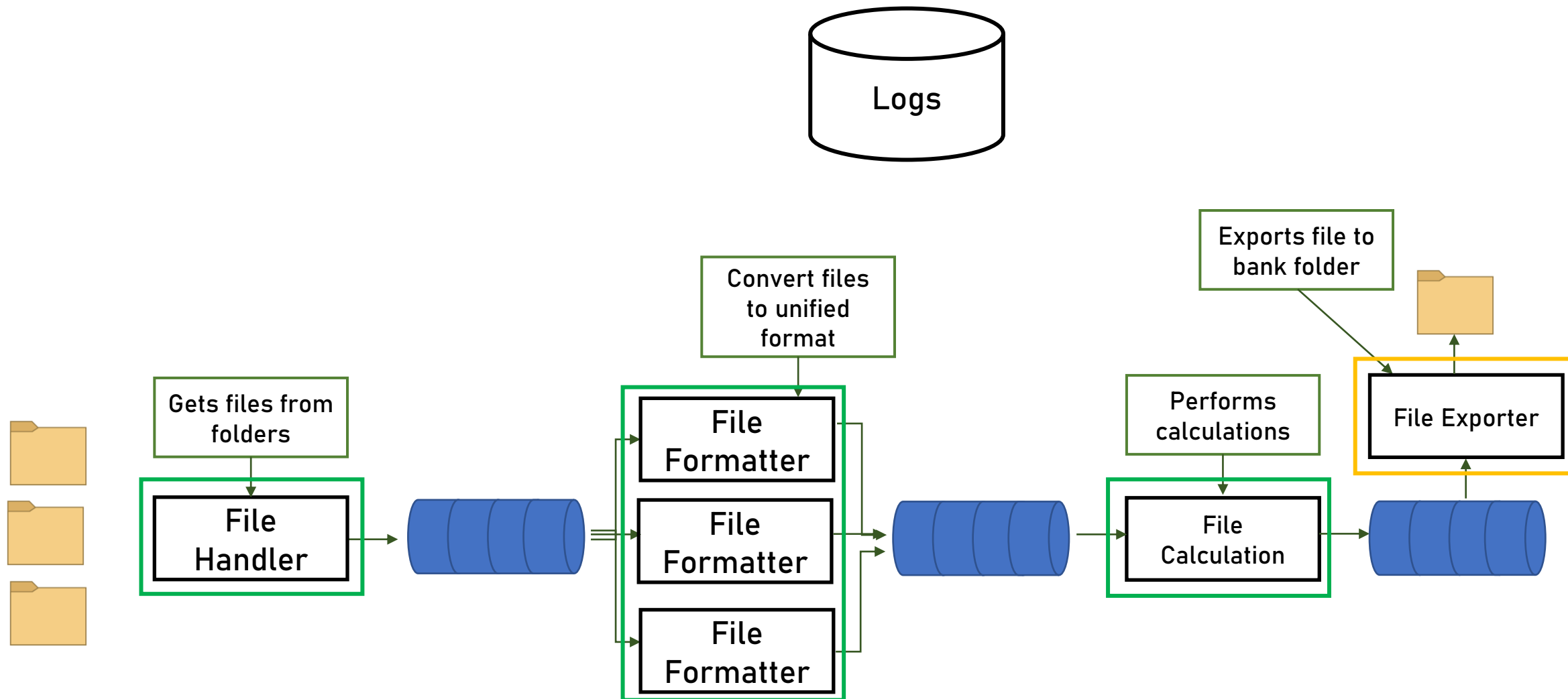
File Calculation

Quite similar to the file formatter, so:

- Tech Stack: .NET Core
- 2 layers architecture
- Redundancy using Consumer Group



Components





File Exporter

What it does:

- Receives files from the queue
- Puts the file in the bank's folder



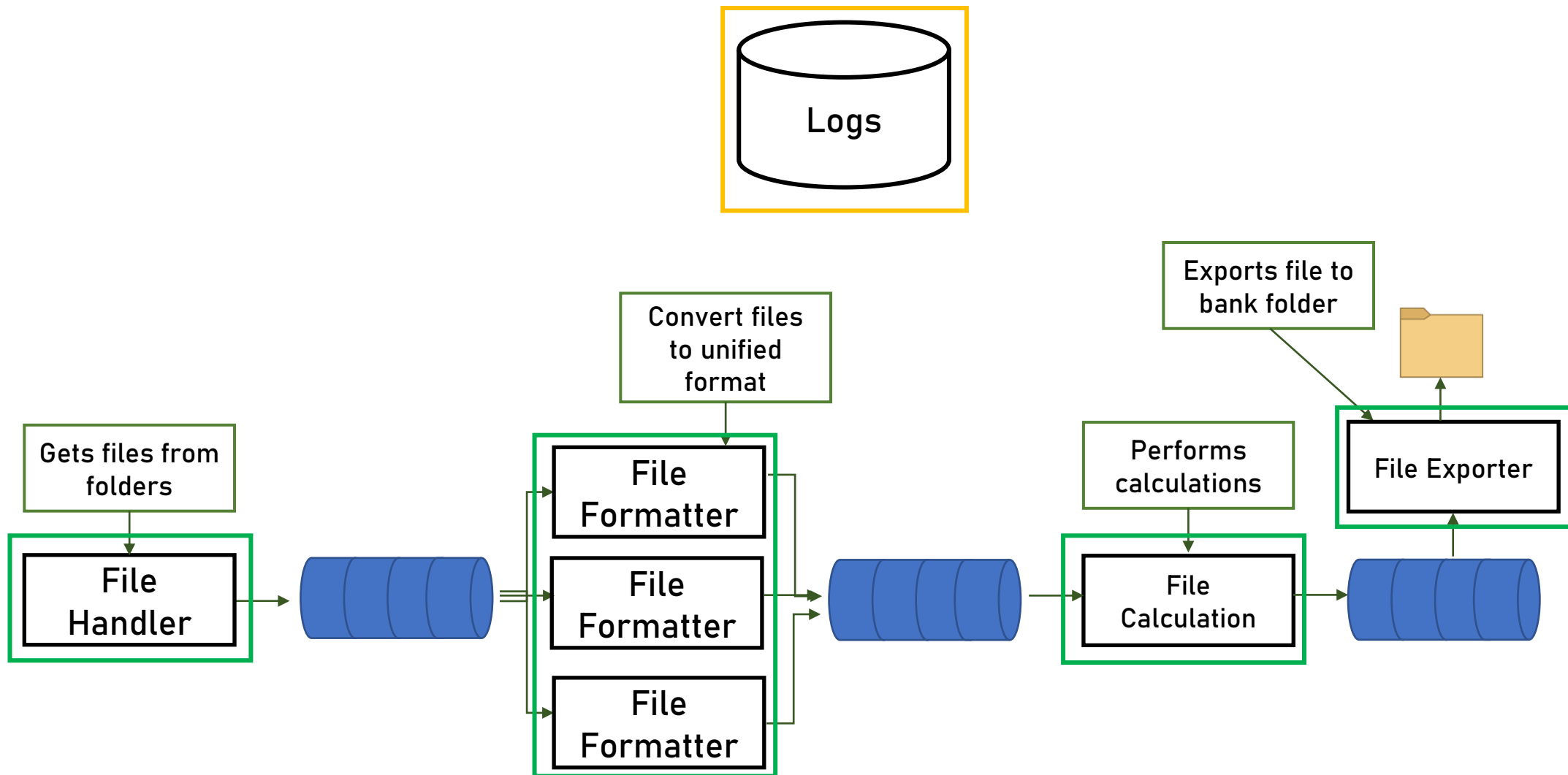
File Exporter

Quite similar to the file calculation service, so:

- Tech Stack: .NET Core
- 2 layers architecture
- Redundancy using Consumer Group



Components





Requirements

Functional

What the system should do

1. Receive file to be processed
2. Validate and process the file
3. Work with various file formats
4. Performs various calculations on the file
5. Create bank payment file
6. Put the payment file in a designated folder
7. Keep log of all the activity for 7 years

Non-Functional

What the system should deal with

1. 500 files / day
2. No data loss
3. 1 min processing time
4. Activity log for 7 years
5. ~2TB / 7 years



Logging

What we need:

- Write a lot of log records
- Allow easy visualizations and analytics
- Preferably – based on existing platform



Logging

Most popular:





Logging

The Elastic Stack has 4 tools:

- Elastic Search: Search and analytics engine
- Kibana: Visualization engine
- Logstash: Data collection pipeline
- Beats: Lightweight log shippers



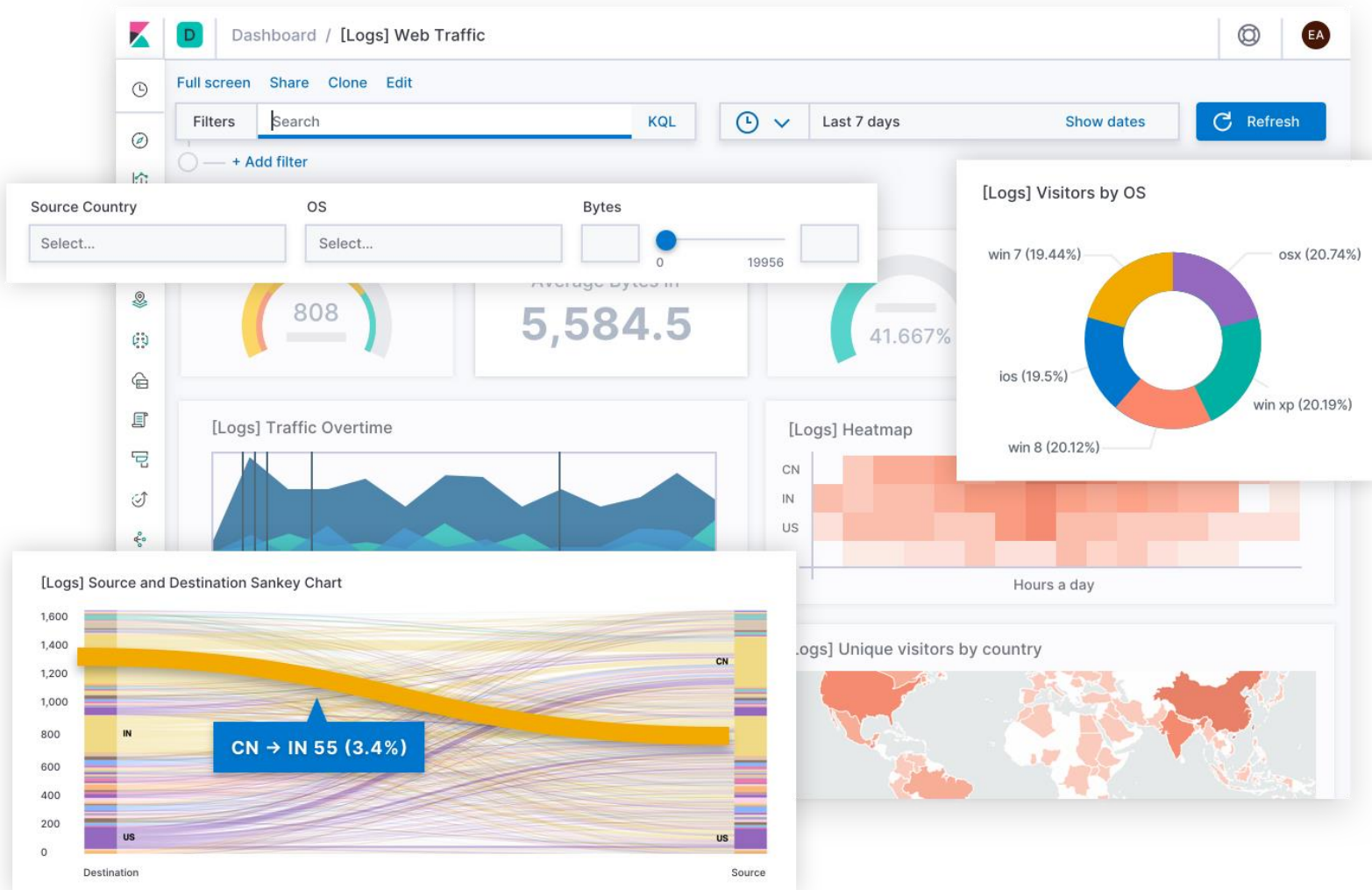
Logging

Or:

- Elastic Search stores your log
- Kibana displays the logs
- Logstash and Beats bring your logs to Elastic



Logging





Logging

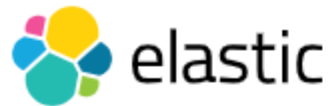
How do we ship logs from .NET Core to Elastic?

Serilog.Sinks.Elasticsearch  build passing  nuget v8.0.1



Logging

How do we ship logs from RabbitMQ to Elastic?



Products

Learn

Company

Pricing

Docs

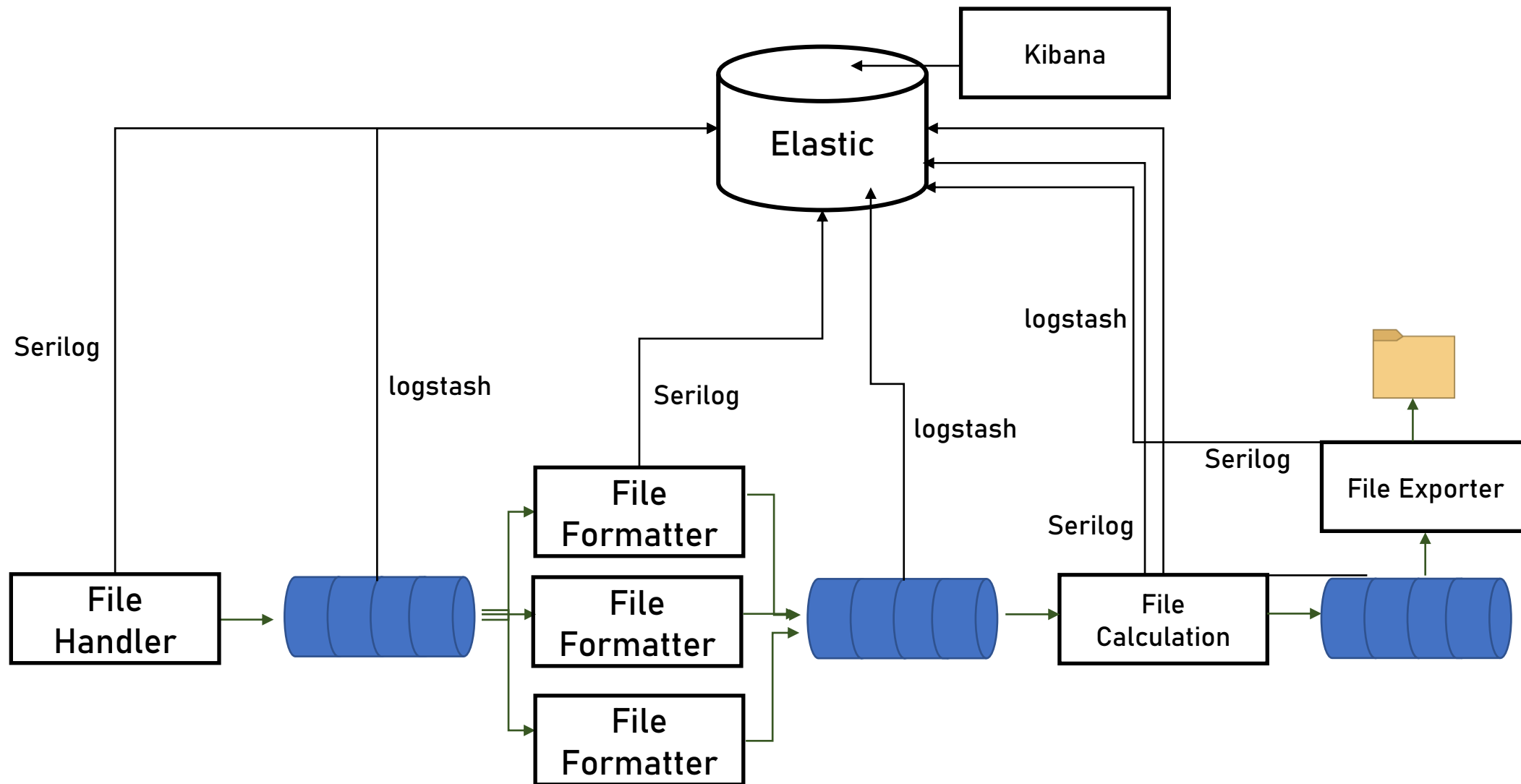
[Logstash Reference \[7.5\]](#) » [Input plugins](#) » Rabbitmq input plugin

« [Puppet_facter input plugin](#)

Rabbitmq input plugin

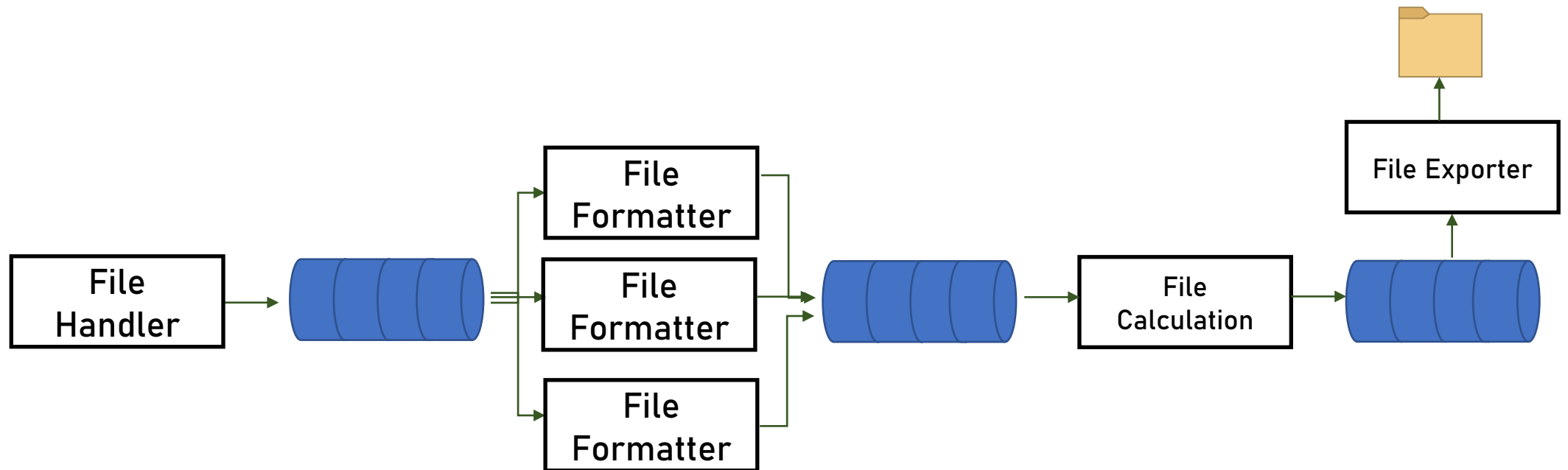
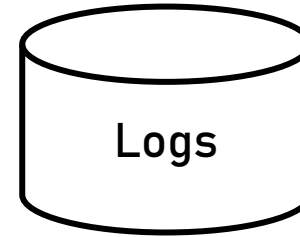


Components



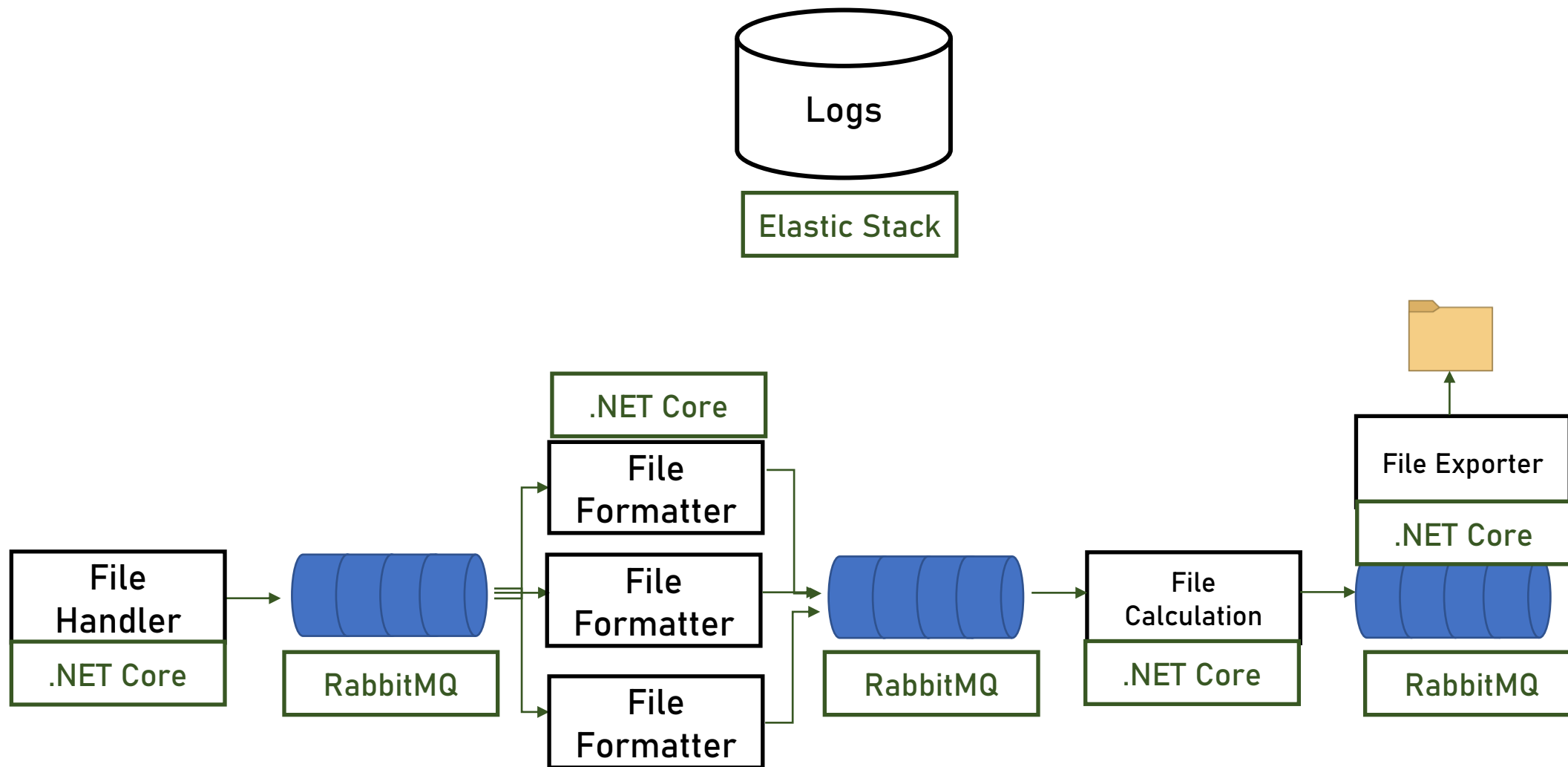


Logic Diagram





Technical Diagram





Physical Diagram

