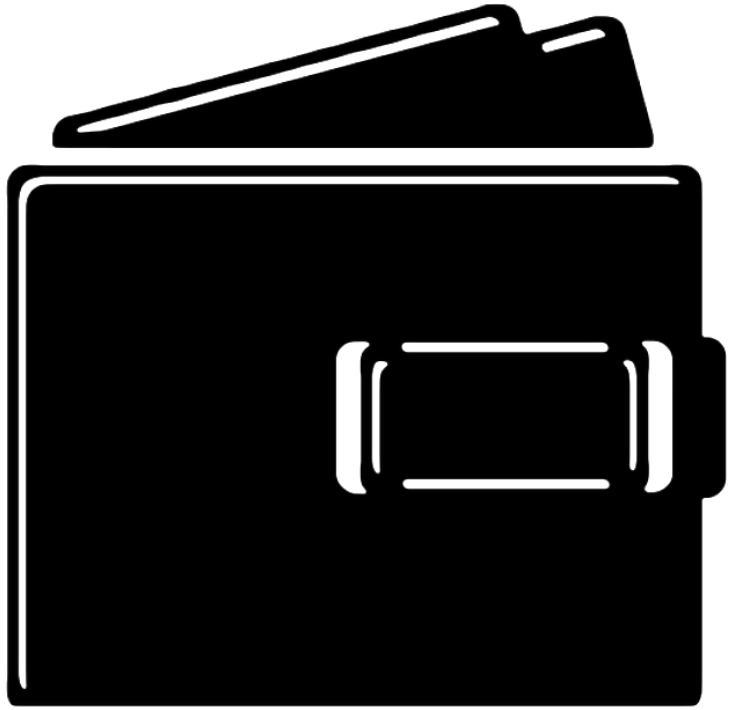
Case Study #4

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

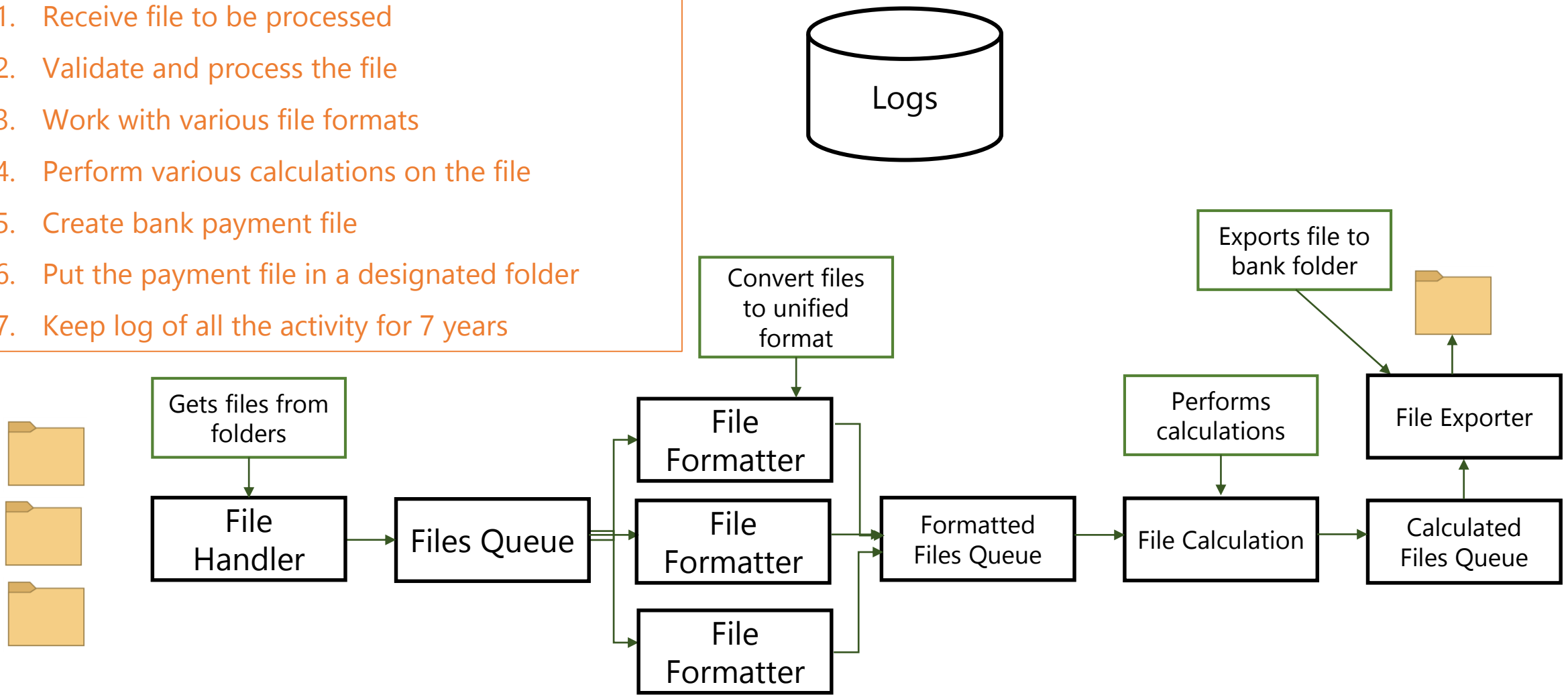
Non-Functional

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

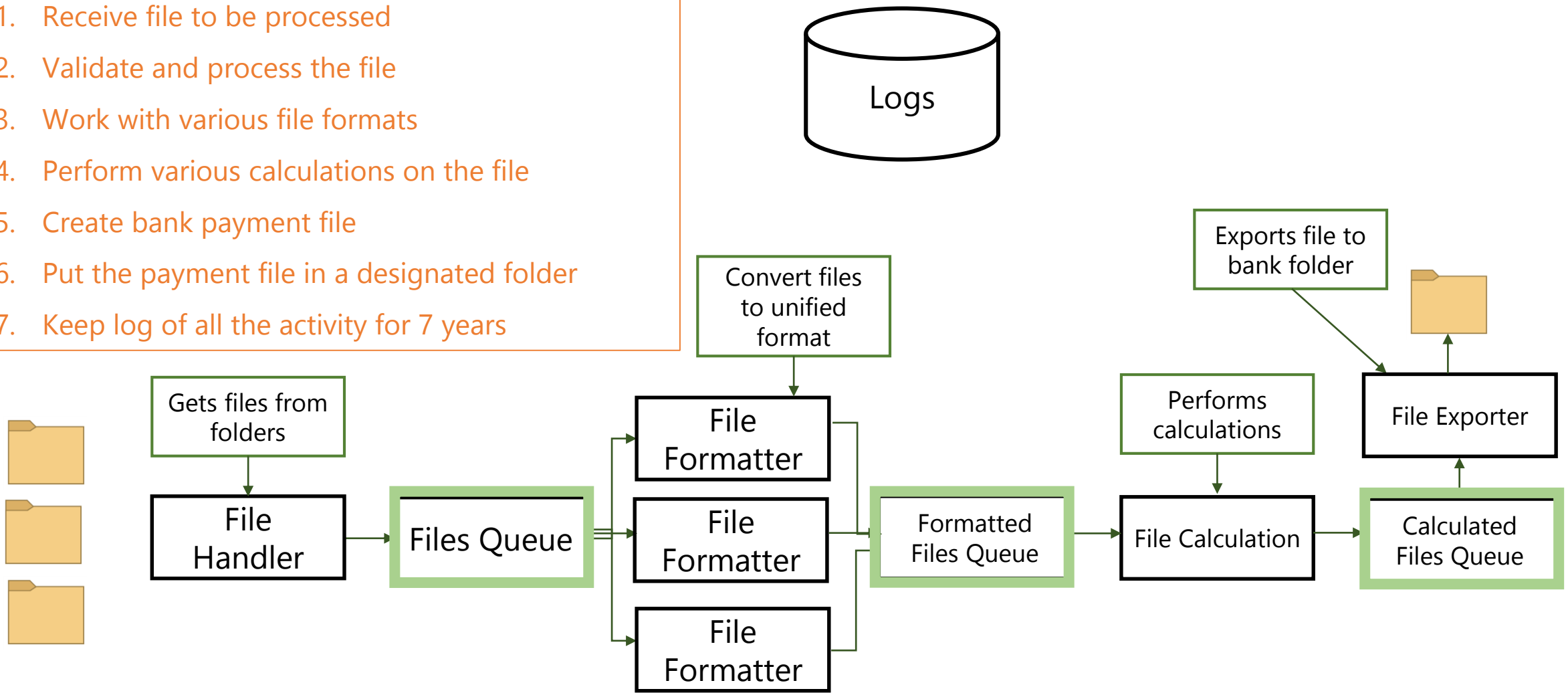
Components

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?

RabbitMQ

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

APACHE kafka®
A distributed streaming platform

- 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

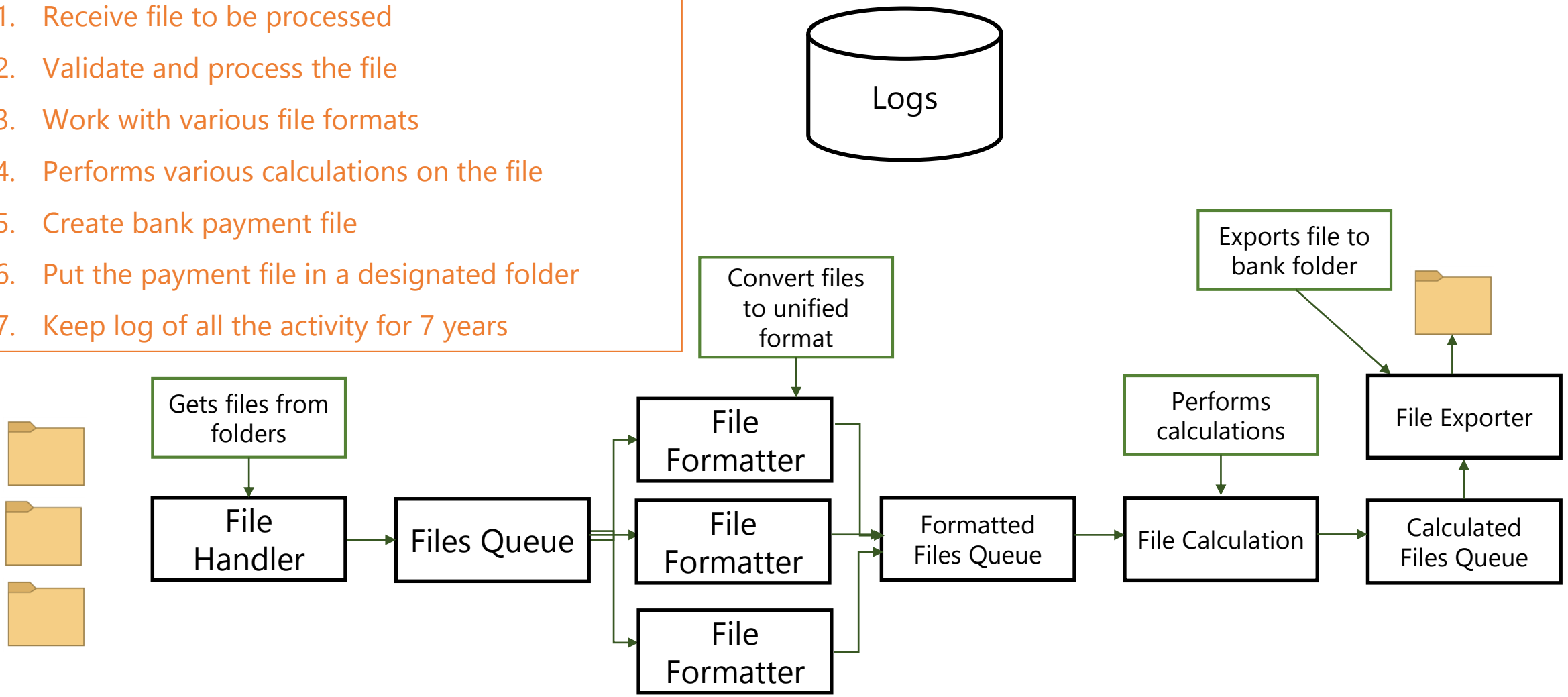
The Queue

- Queue is usually represented like this:

- So…

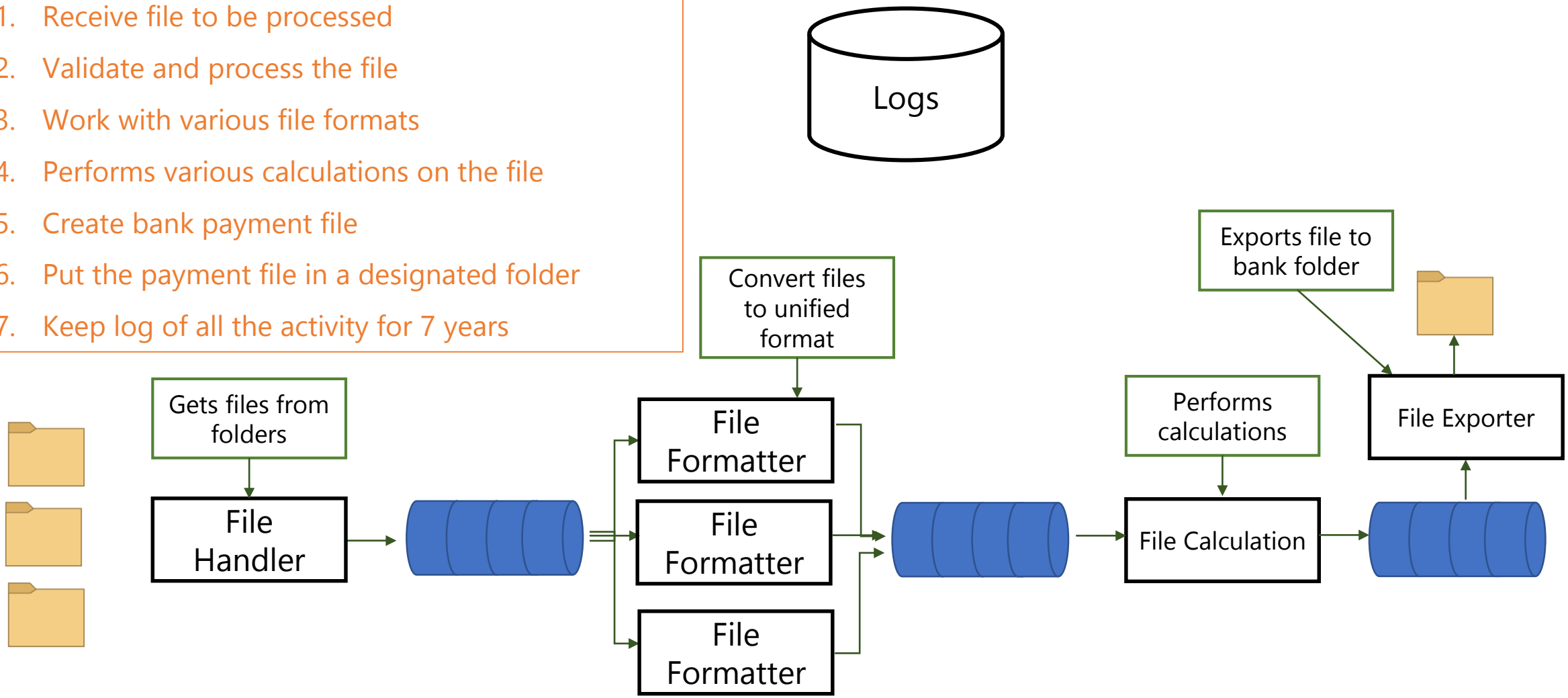
Components

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

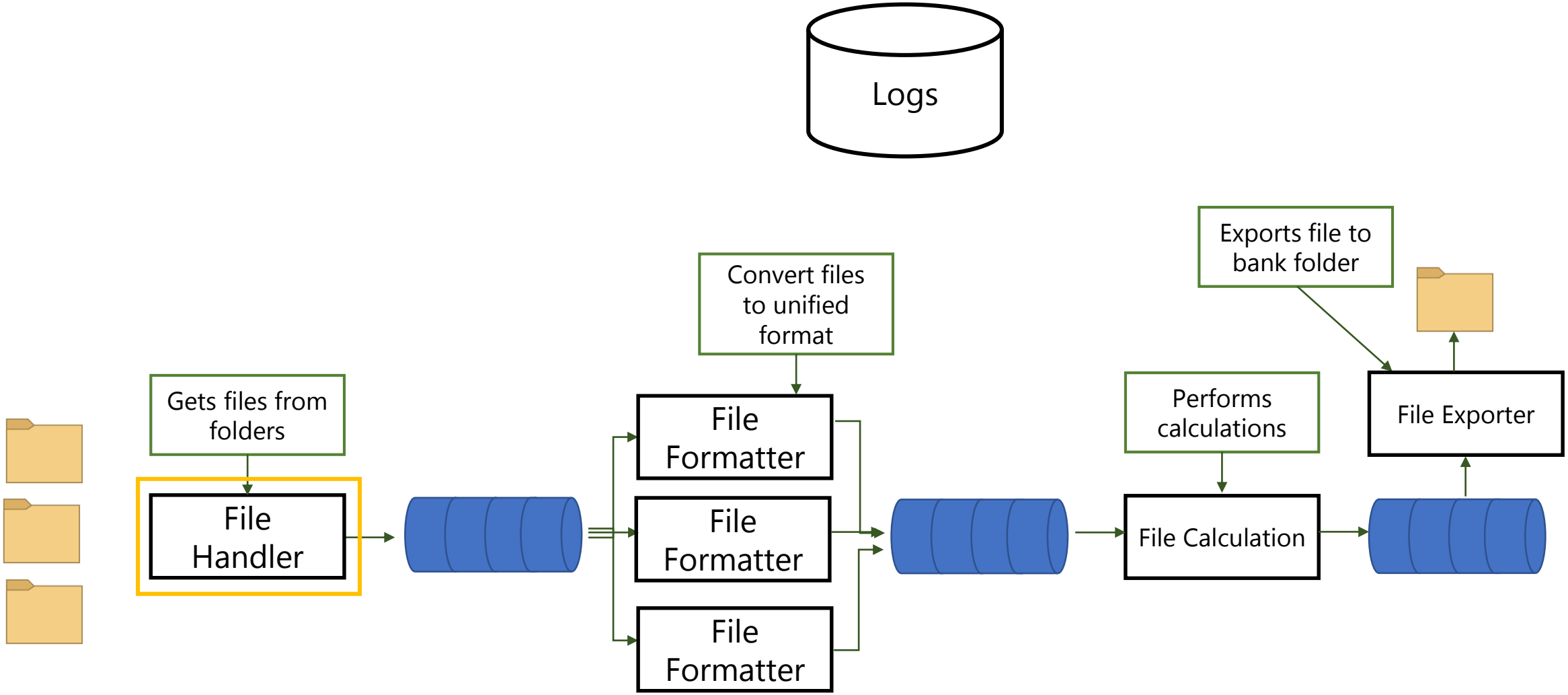


Components

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



Components


Logs
Exports file to bank folder
Convert files to unified format
Gets files from folders
Performs calculations
File Exporter
File Formatter
File Formatter
File Formatter
File Handler
File Calculation

File Handler

What it does:

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

Application Type

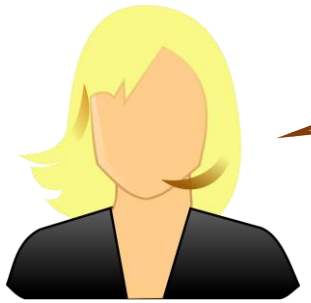
- Web App & Web API ✗
- Mobile App ✗
- Console ✓
- Service ✓
- Desktop App ✗

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:

Best fortunes responses per second, Dell R440 Xeon Gold + 10 GbE

Rnk	Framework	Best performance (higher is better)	
1	aspcore-ado-pg	300,613	100.0% (42.8%)
2	aspcore-rhtx-pg	294,096	97.8% (41.9%)
3	aspcore-mw-ado-pg	249,447	83.0% (35.5%)
4	aspcore-mw-dap-pg	217,793	72.4% (31.0%)
5	aspcore-ado-my	200,634	66.7% (28.6%)
6	aspcore-mw-ado-my	182,694	60.8% (26.0%)
7	aspcore-mw-ef-pg	173,958	57.9% (24.8%)
8	aspcore-mw-dap-my	171,681	57.1% (24.5%)
9	nodejs-mongodb-raw	125,743	41.8% (17.9%)
10	aspcore-mvc-ado-pg	120,731	40.2% (17.2%)
11	aspcore-mvc-dap-pg	115,062	38.3% (16.4%)
12	aspcore-mvc-ado-my	107,835	35.9% (15.4%)
13	aspcore-mvc-dap-my	102,905	34.2% (14.7%)
14	aspcore-mvc-ef-pg	95,922	31.9% (13.7%)
15	nodejs	91,086	30.3% (13.0%)
16	nodejs-chakra	91,058	30.3% (13.0%)
17	nodejs-postgres	63,532	21.1% (9.0%)
18	nodejs-mysql	57,995	19.3% (8.3%)
19	spring	30,891	10.3% (4.4%)

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

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

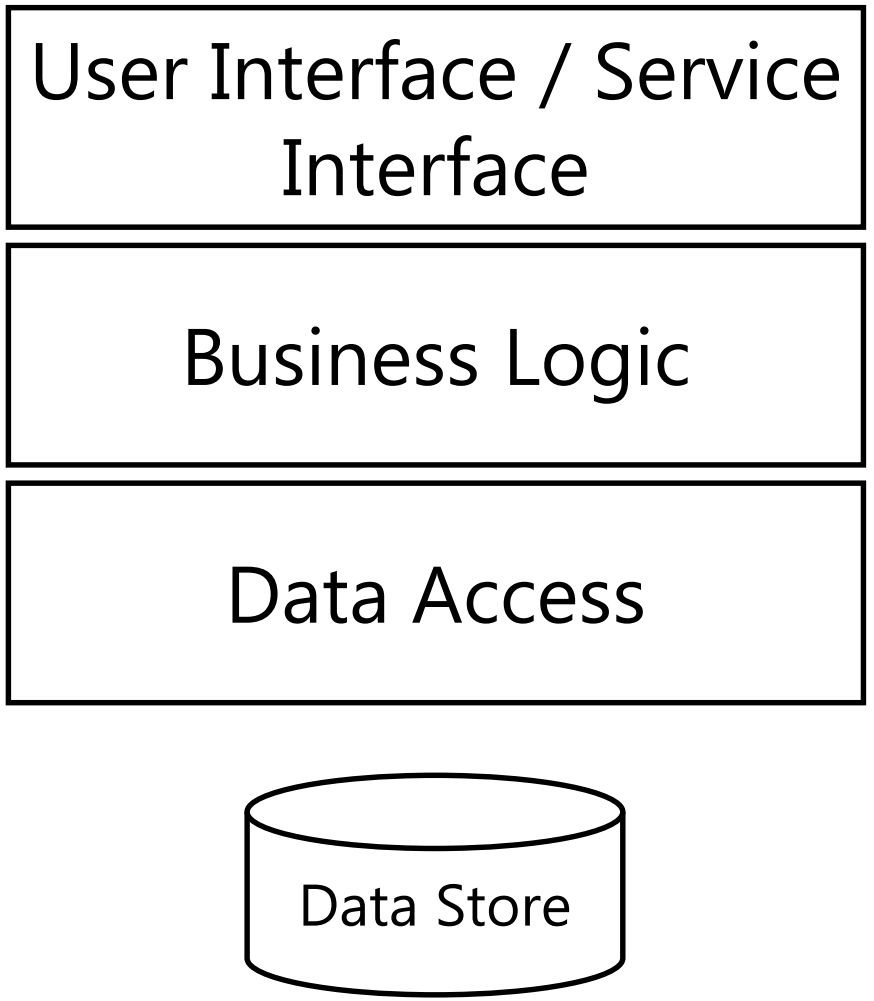
Technology Stack

Threading support:

Technology Stack - Decision

Architecture

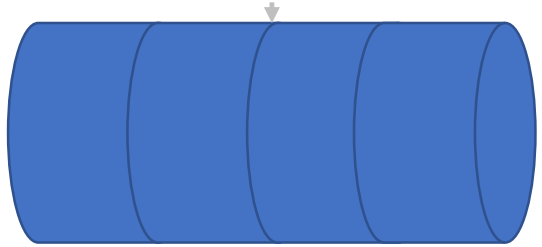
Traditional:



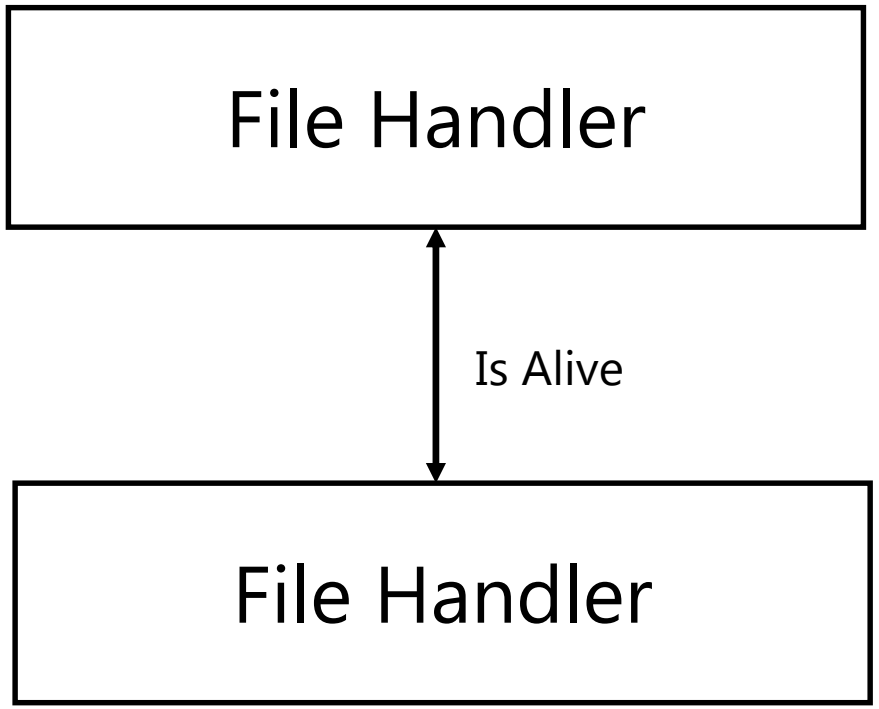
Architecture

File Watcher

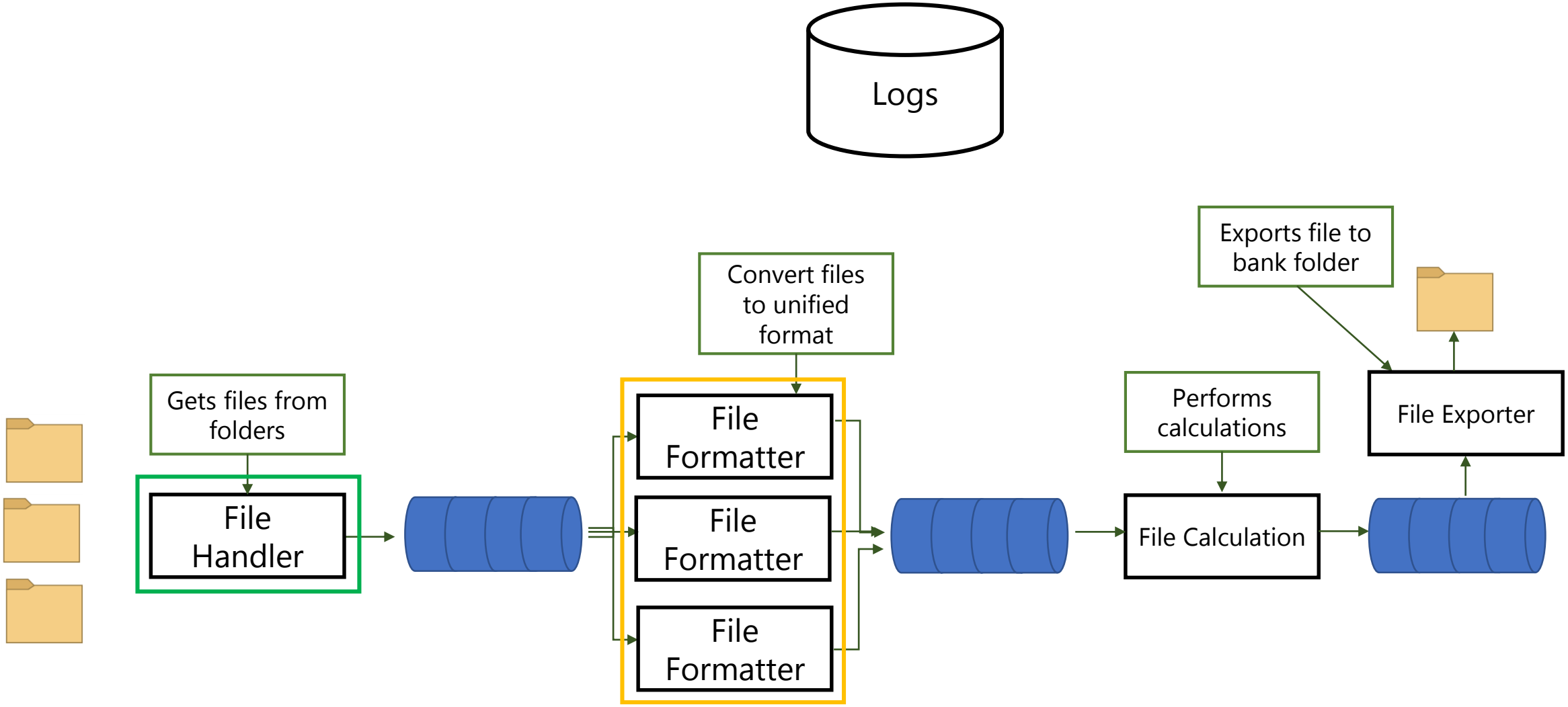
Topic is selected by file location



Redundancy



Components


Logs
Exports file to bank folder
Convert files to unified format
Gets files from folders
Performs calculations
File Exporter
File Formatter
File Formatter
File Formatter
File Handler
File Calculation

File Formatter

What it does:

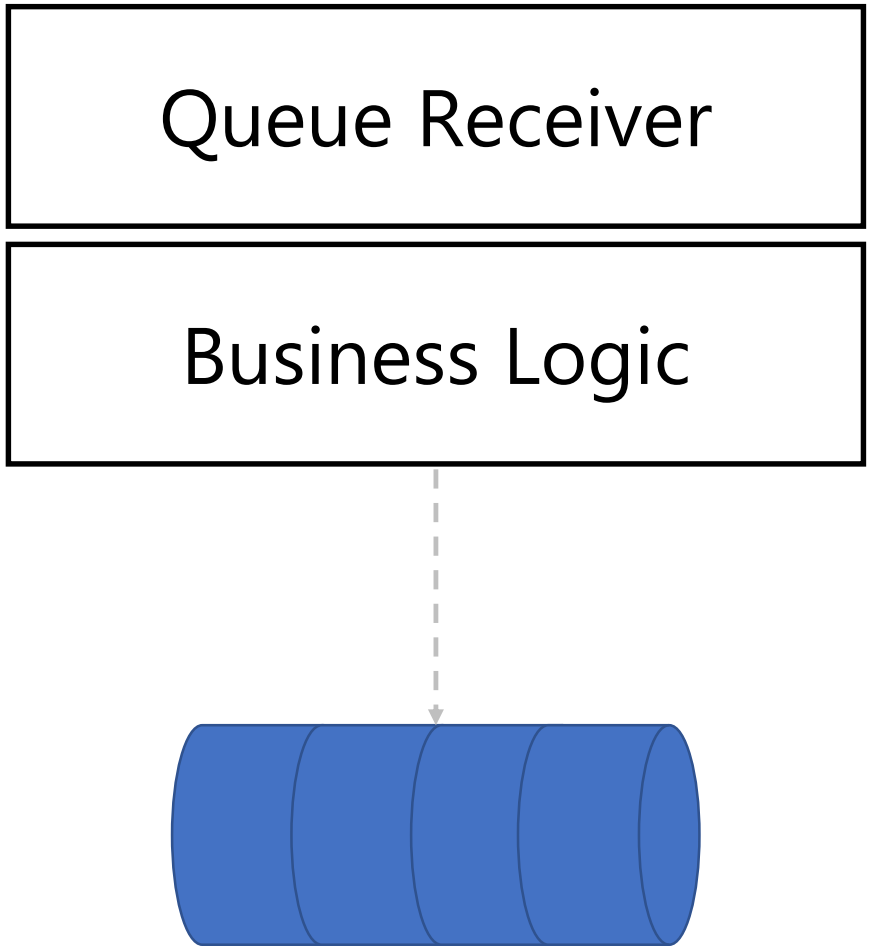
- Receives files from its specific topic
- Validates and formats the file to unified format
- Puts the new file in a queue
- New formatters will be developed for new file

Application Type

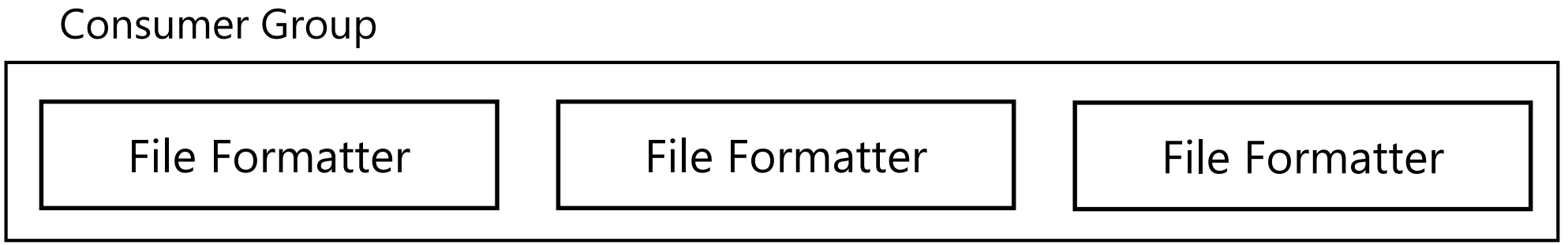
- Web App & Web API ✗
- Mobile App ✗
- Console ✓
- Service ✓
- Desktop App ✗

Technology Stack

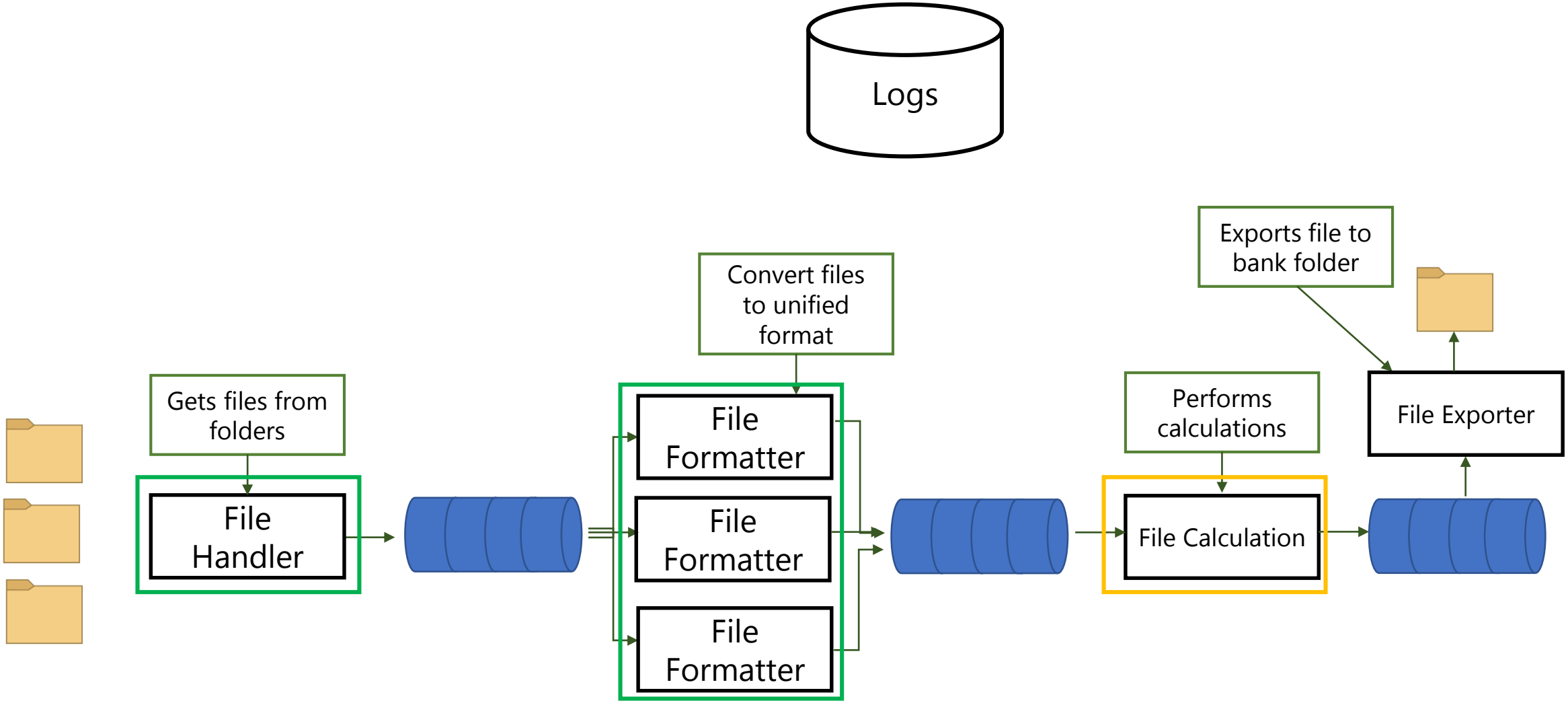
Architecture



File Formatter Redundancy



Components


Logs
Exports file to bank folder
Convert files to unified format
Gets files from folders
Performs calculations
File Exporter
File Formatter
File Formatter
File Formatter
File Handler
File Calculation

File Calculation

What it does:

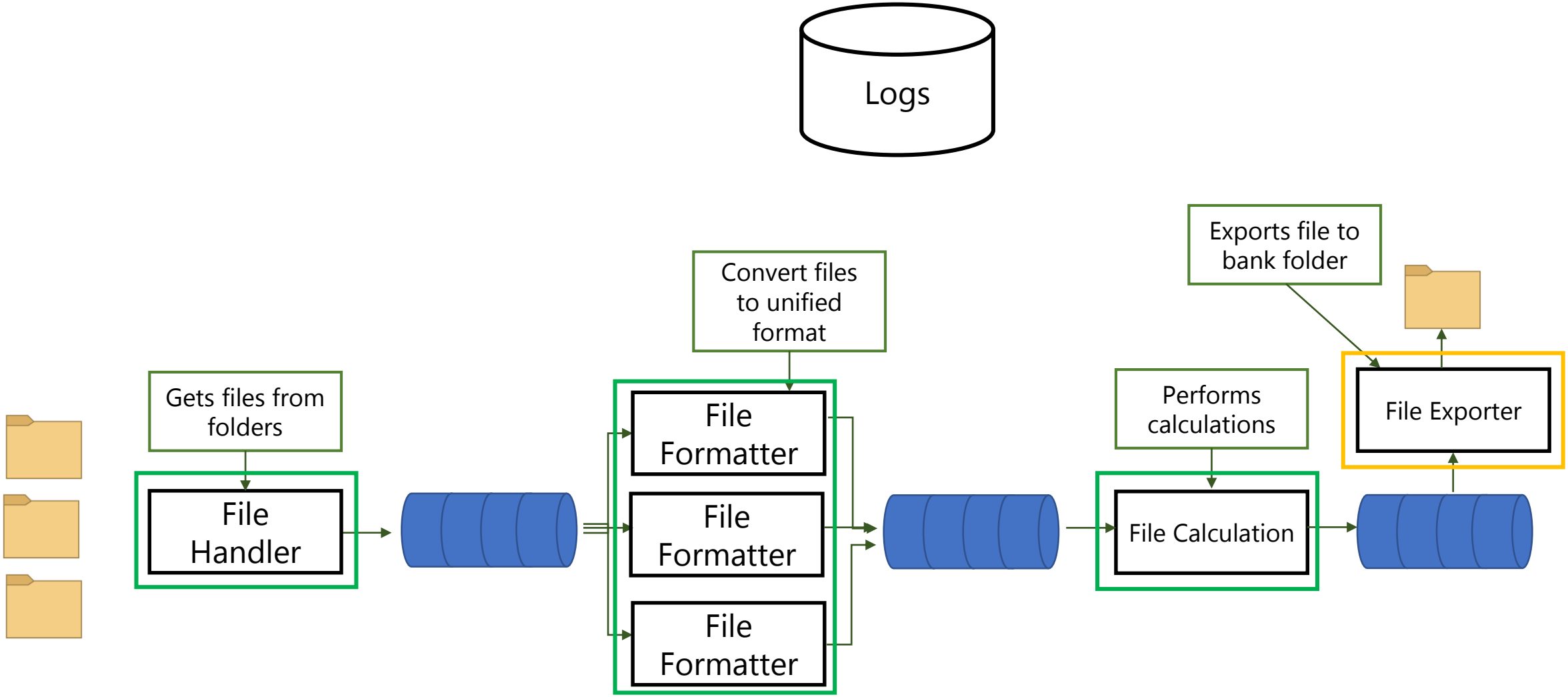
- Receives files from the queue
- Performs some calculations on the data
- Puts the new file in a queue

File Calculation

Quite similar to the file formatter, so:

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

Components


Logs
Exports file to bank folder
Convert files to unified format
Gets files from folders
Performs calculations
File Exporter
File Formatter
File Formatter
File Formatter
File Handler
File Calculation

File Exporter

What it does:

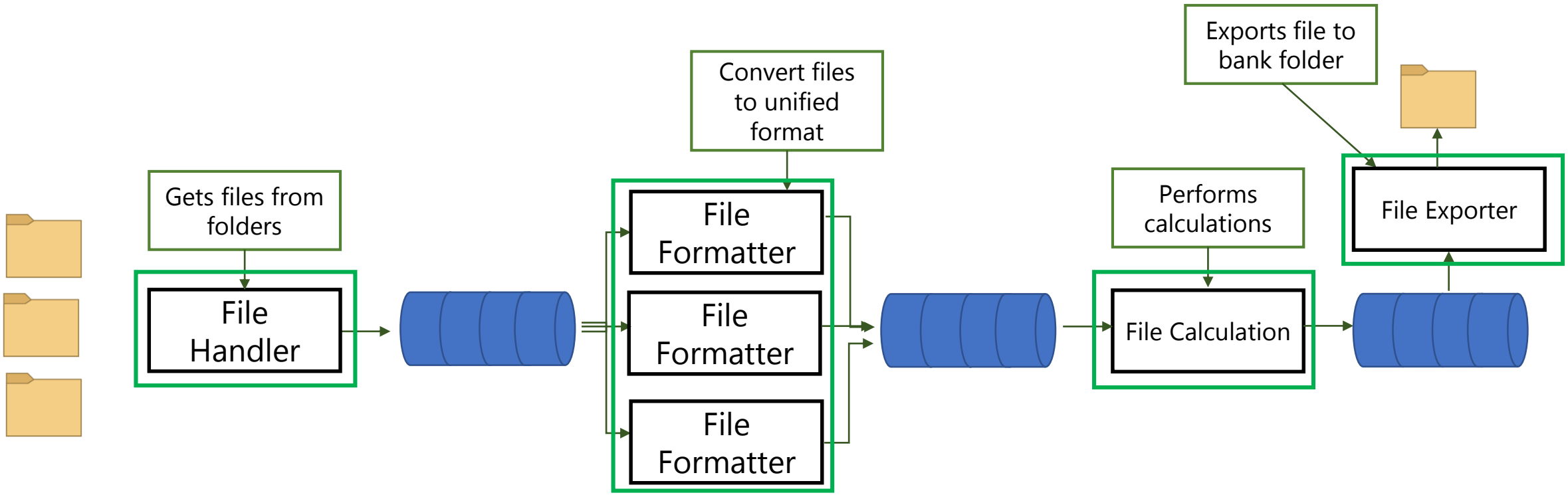
- Receives files from the queue
- Puts the file in the bank’s folder

File Exporter

Quite similar to the file calculation service, so:

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

Components


Logs
Exports file to bank folder
Convert files to unified format
Gets files from folders
Performs calculations
File Exporter
File Formatter
File Formatter
File Formatter
File Handler
File Calculation

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:

elastic

Logging

The Elastic Stack has 4 tools:

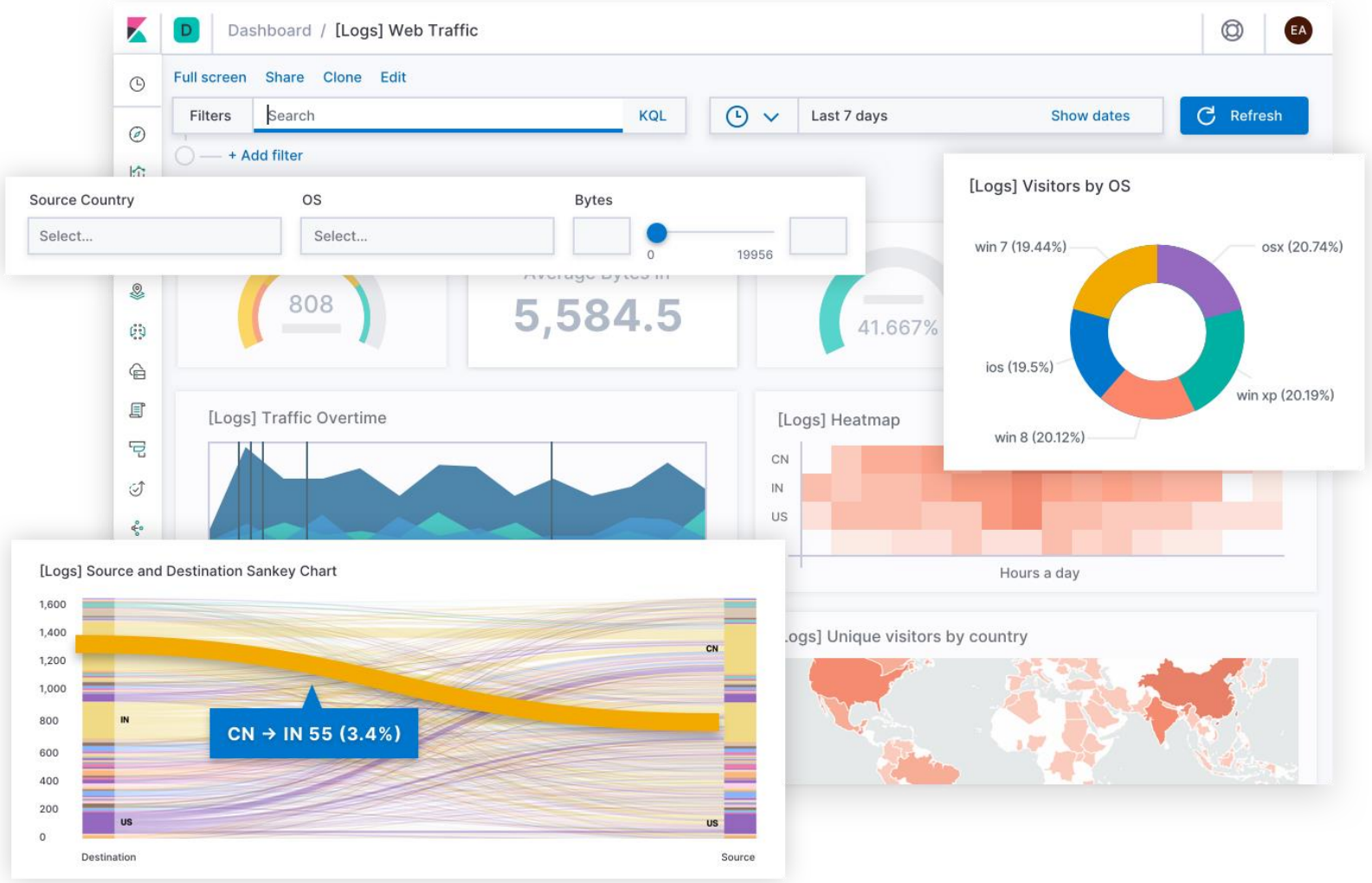
- Elastic Search: Search and analytics engine
- Kibana: Visualization engine
- Logstash: Data collection pipeline
- Beats: Lightweight log shippers

Logging

Or:

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

Logging


Dashboard / [Logs] Web Traffic
EA
Full screen
Share
Clone
Edit
Filters
Search
KQL
Last 7 days
Show dates
Refresh
+ Add filter
Source Country
Select...
OS
Select...
Bytes
0
19956
808
5,584.5
41.667%
[Logs] Visitors by OS
win 7 (19.44%)
osx (20.74%)
ios (19.5%)
win xp (20.19%)
win 8 (20.12%)
[Logs] Traffic Overtime
[Logs] Heatmap
CN
IN
US
Hours a day
[Logs] Source and Destination Sankey Chart
1,600
1,400
1,200
1,000
800
600
400
200
0
CN → IN 55 (3.4%)
Destination
Source
[Logs] Unique visitors by country

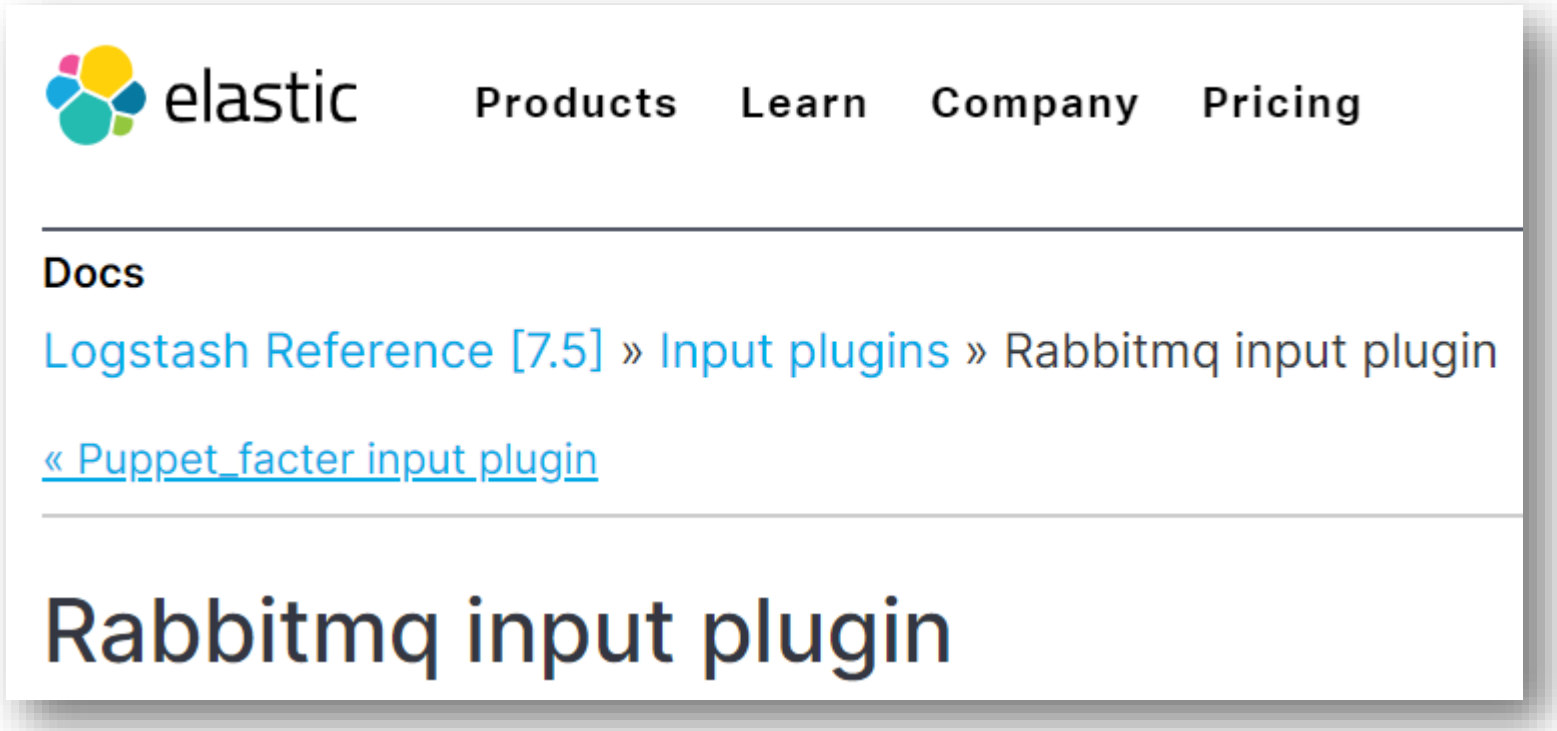
Logging

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

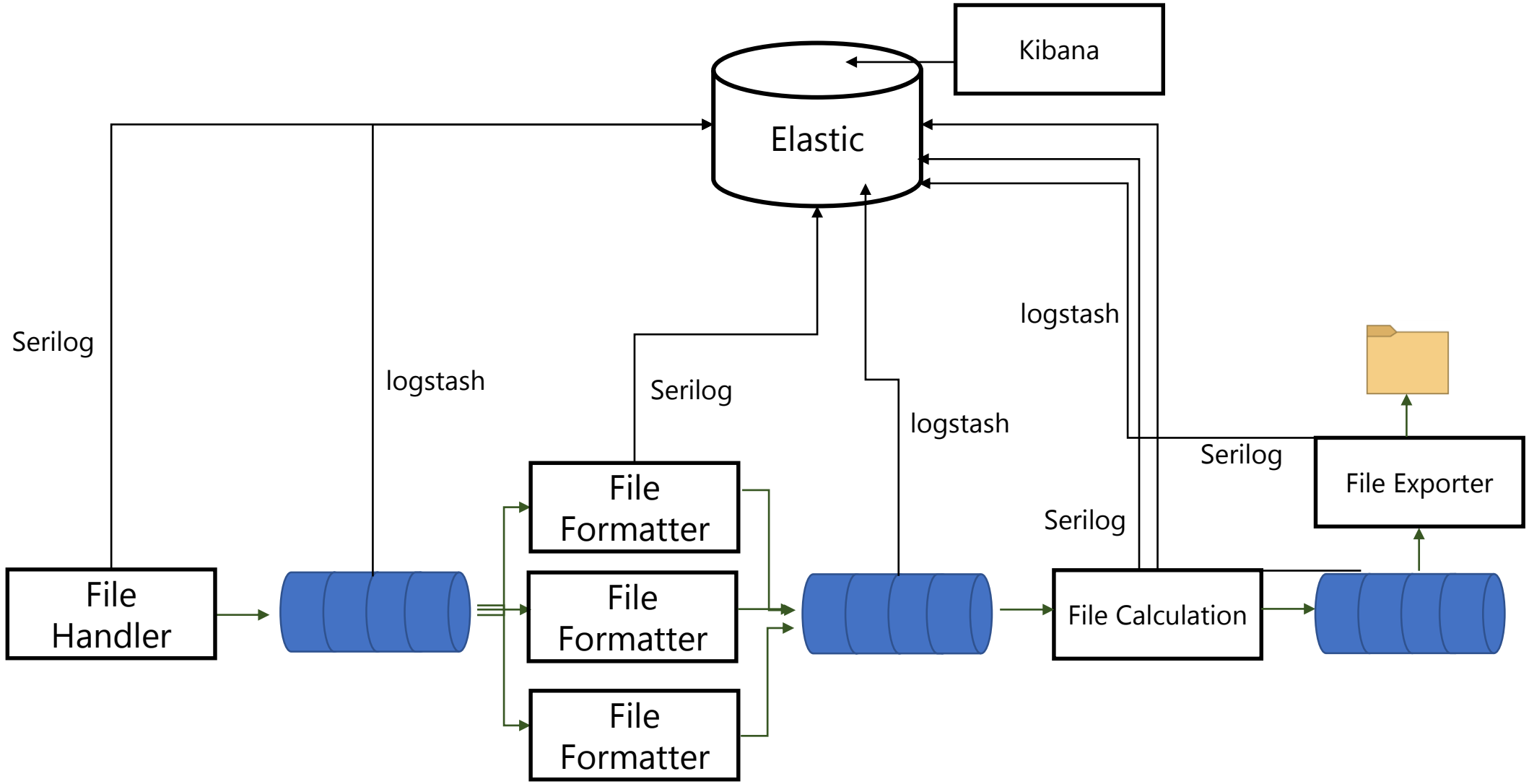
Serilog.Sinks.Elasticsearch  

Logging

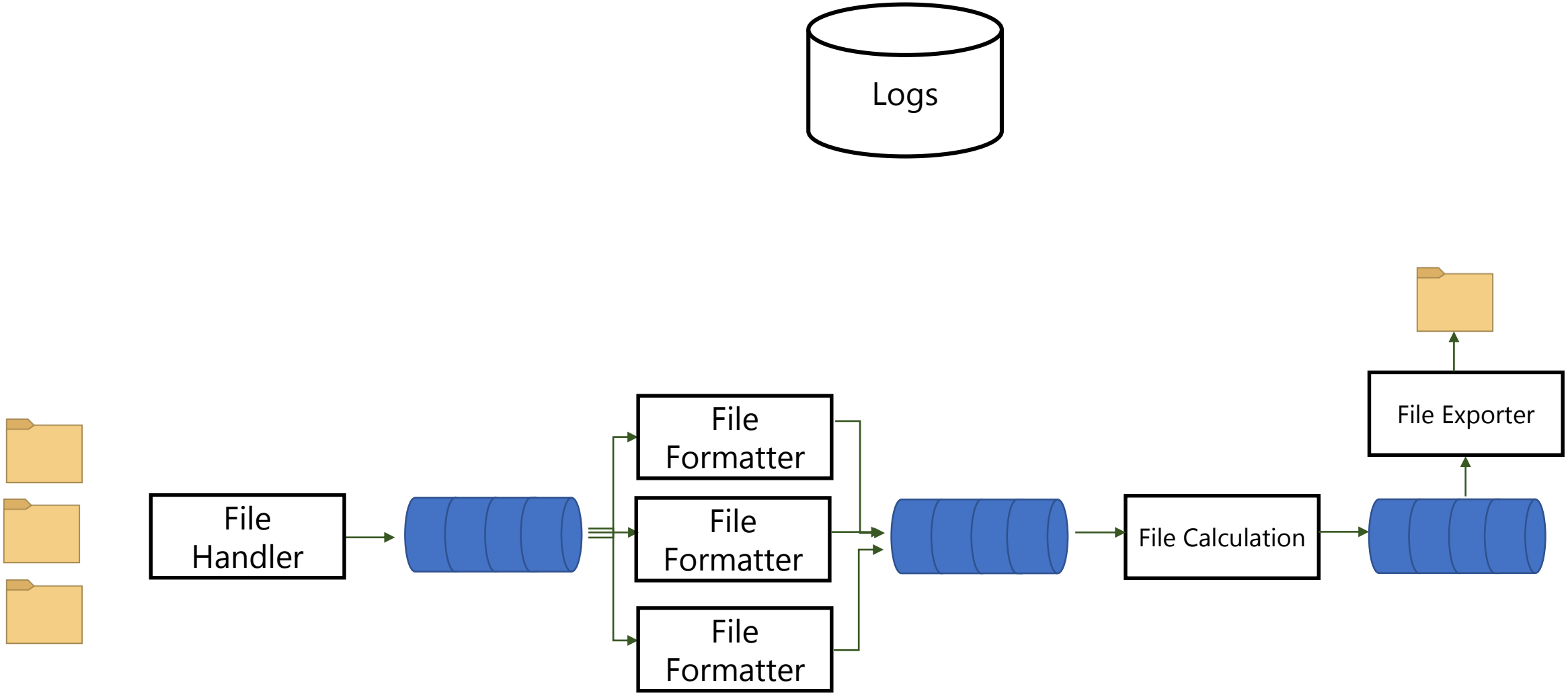
How do we ship logs from RabbitMQ to Elastic?



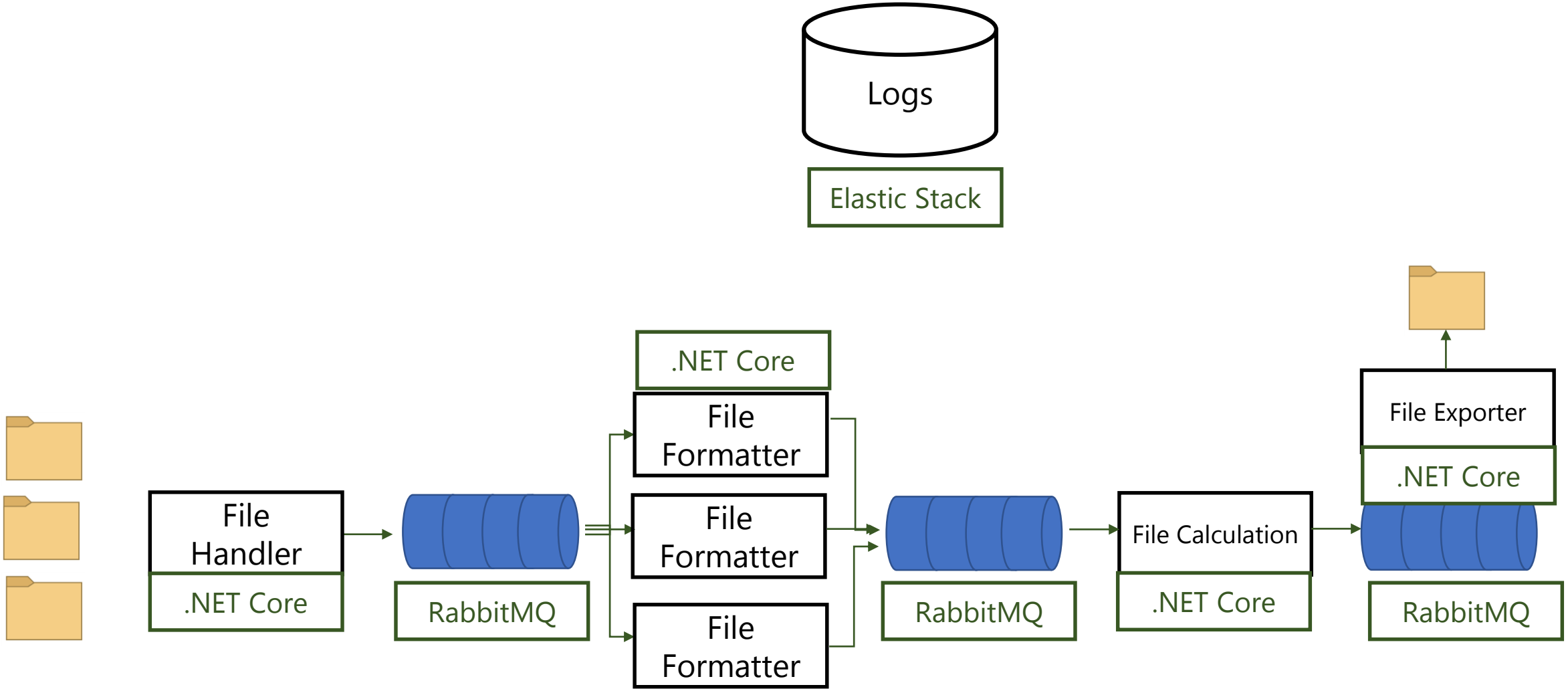
Components


Kibana
Elastic
Serilog
logstash
Serilog
logstash
logstash
Serilog
Serilog
File Exporter
File Handler
File Formatter
File Formatter
File Formatter
File Calculation

Logic Diagram



Technical Diagram



Physical Diagram