

# Building Log Analysis Workflows

---



**Cristian Pascariu**

Information Security Professional

[www.cybersomething.com](http://www.cybersomething.com)



# Overview



## **Go beyond single-purpose scripts**

- Design log analytics capabilities

## **Touch upon technical aspects**

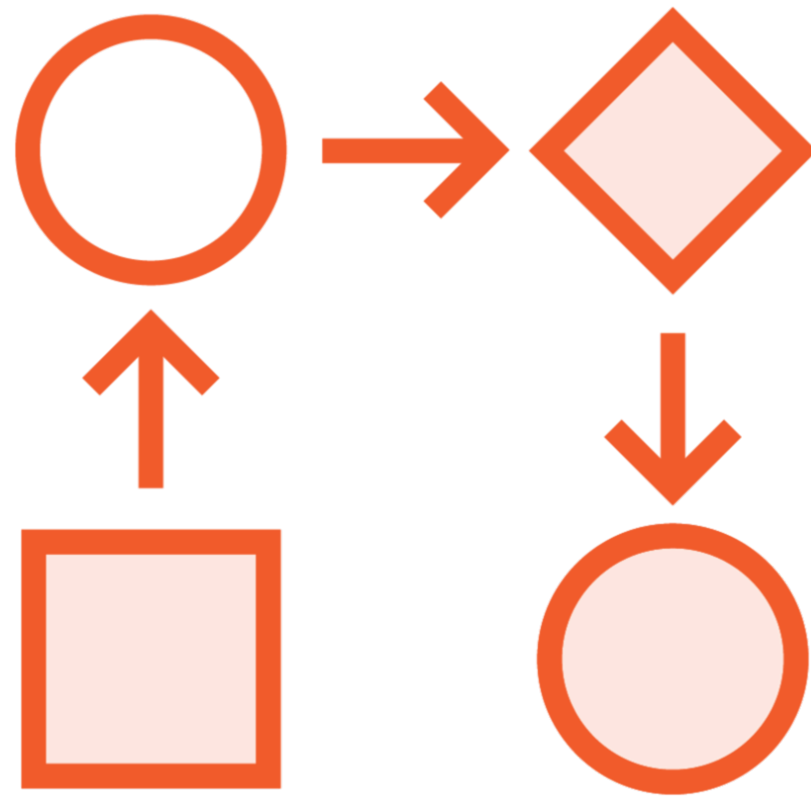
- Working with JSON
- Interacting with web services using built-in modules

## **Integrating with other services and technologies**

- Generate alerts and store them in a database
- Index enriched data to a SIEM



# Building Log Analysis Workflows



## **Leverage Python to gain insight from log files**

- Identify and extract IoC's
- Visualize trends

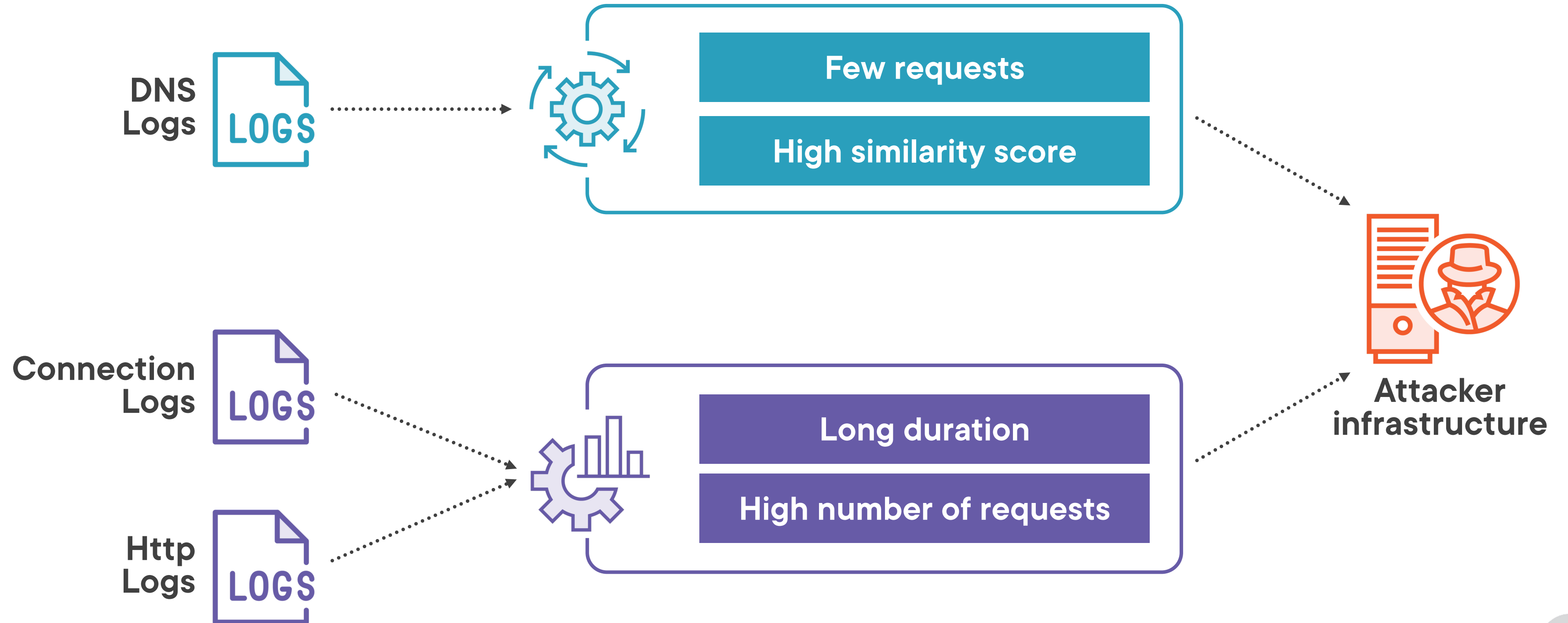
## **Automate manual analysis**

- Focus on next steps in the process

## **Opportunities to enhance existing workflows**



# Investigation Workflows



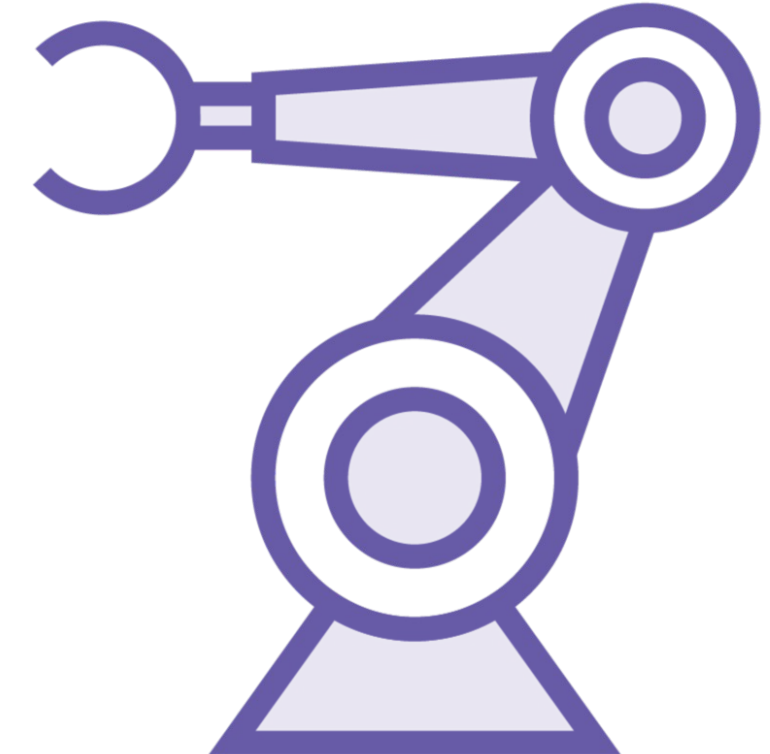
# Log Analysis Workflows in Practice



**Push results into  
centralized  
repositories**



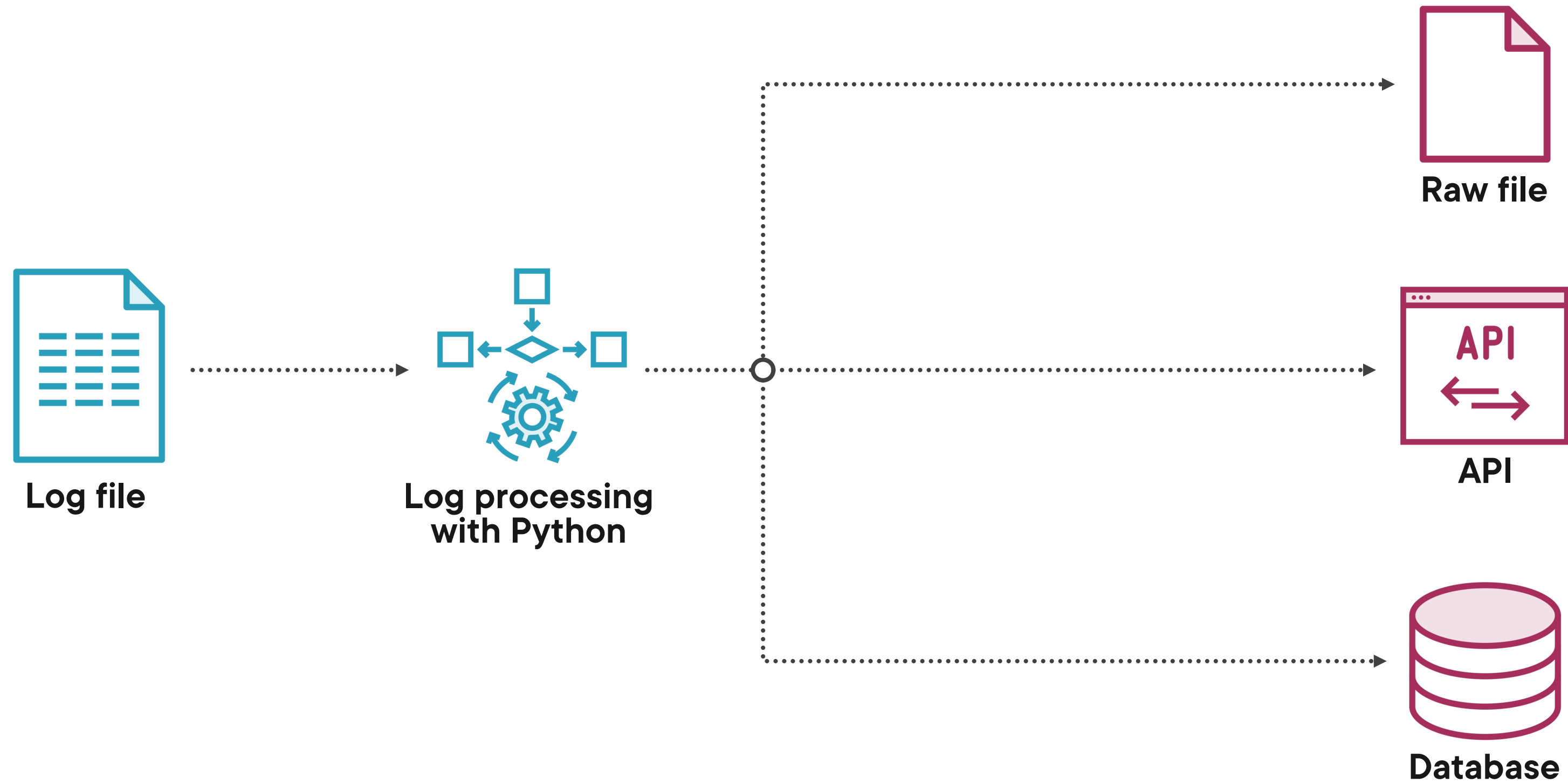
**Correlate with data  
from other services**



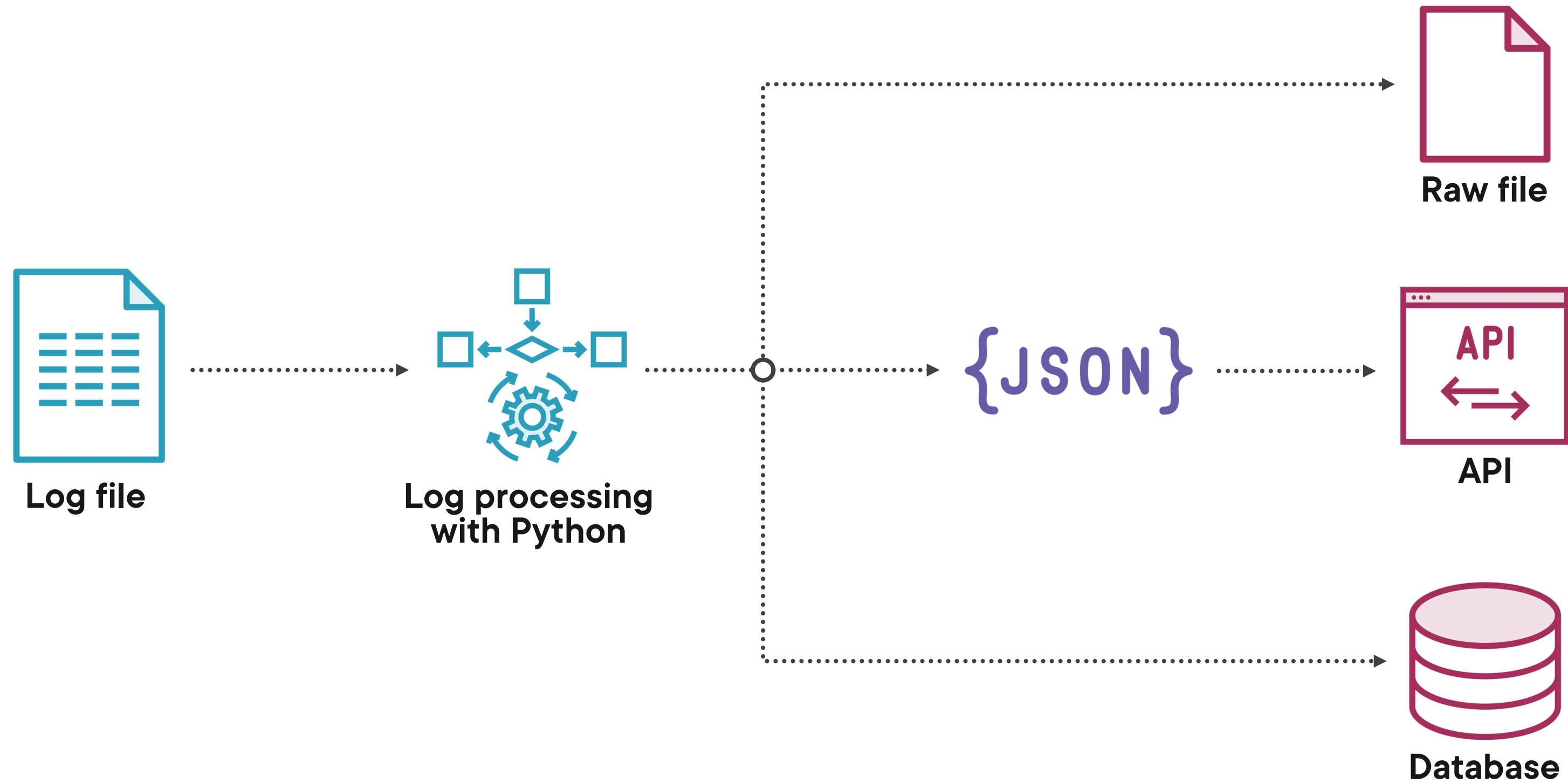
**Build stand-alone  
solutions**



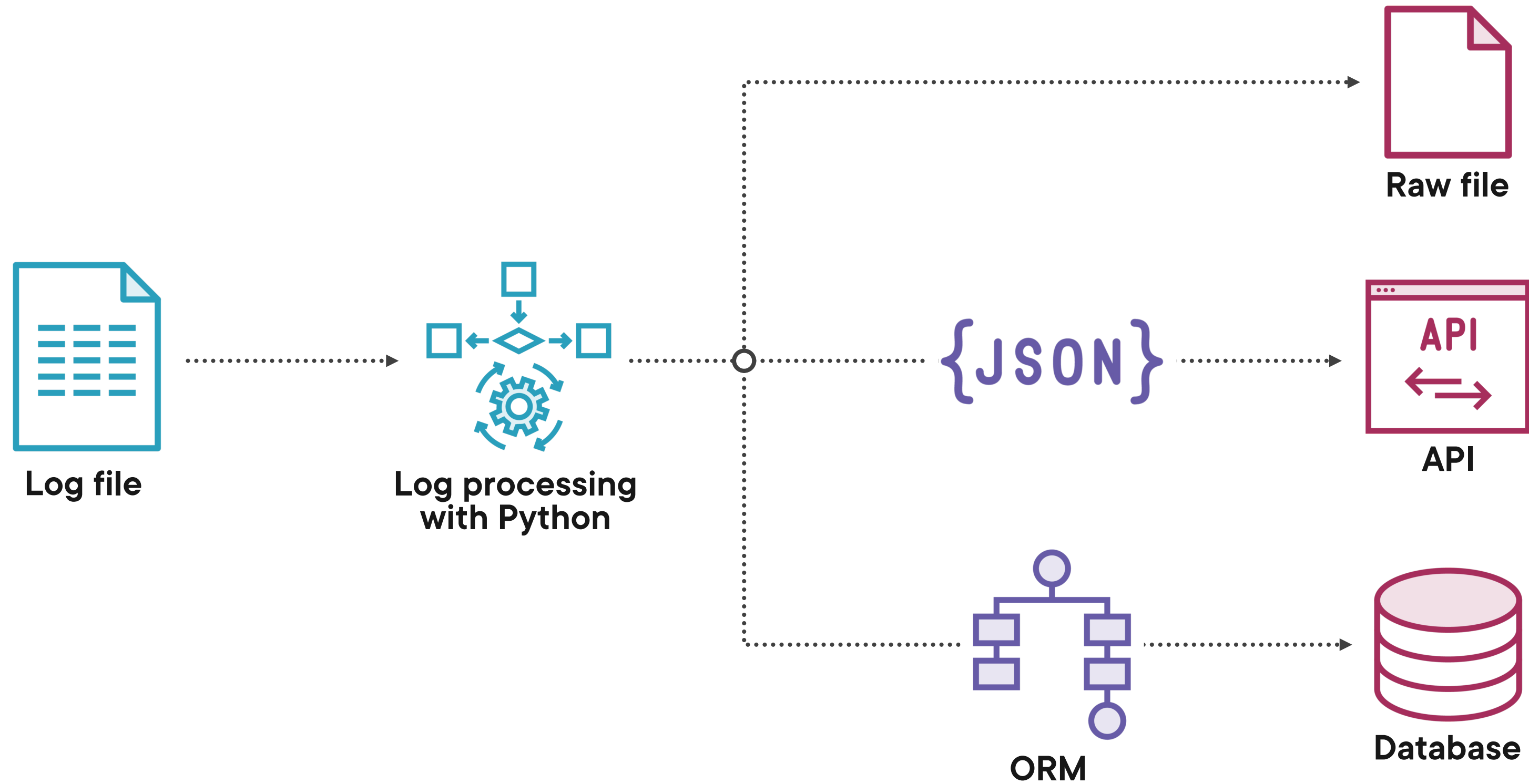
# Integration Workflows



# Integration Workflows

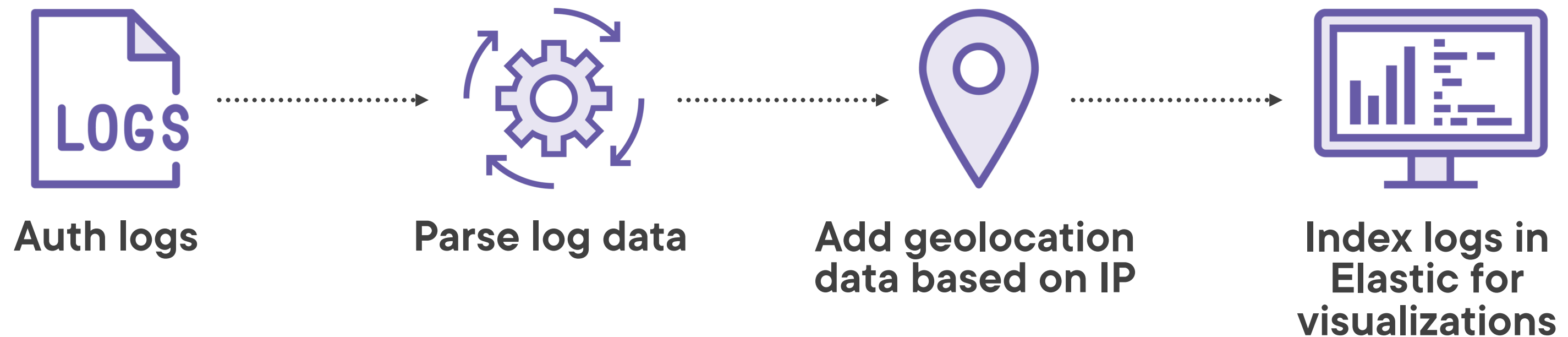
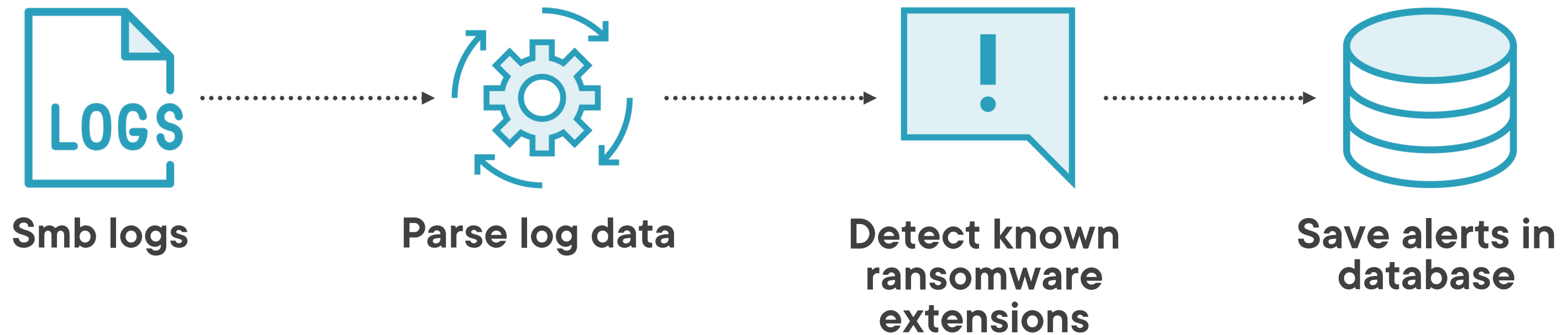


# Integration Workflows

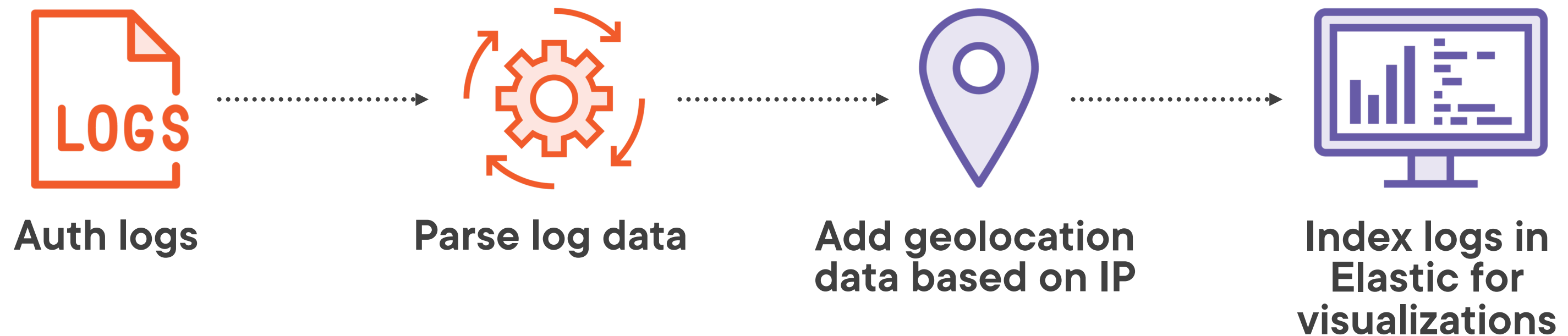
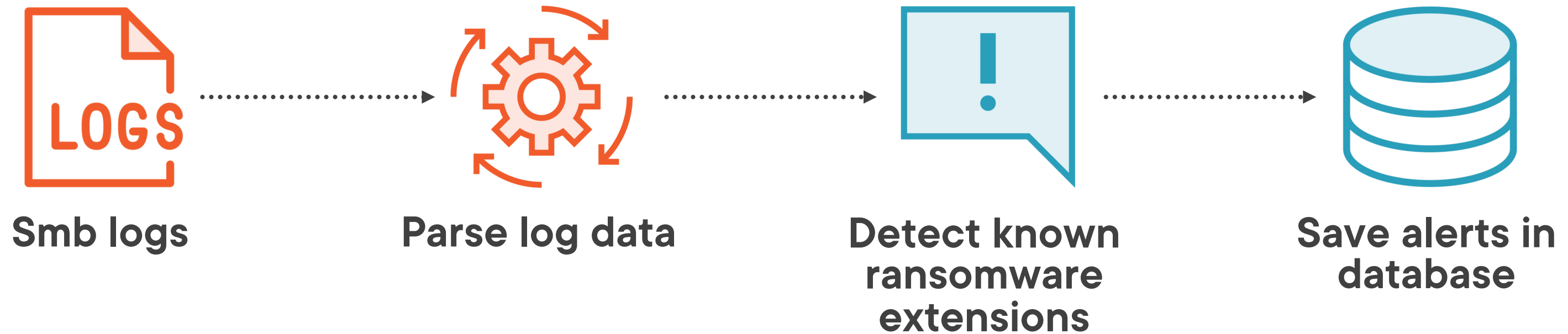




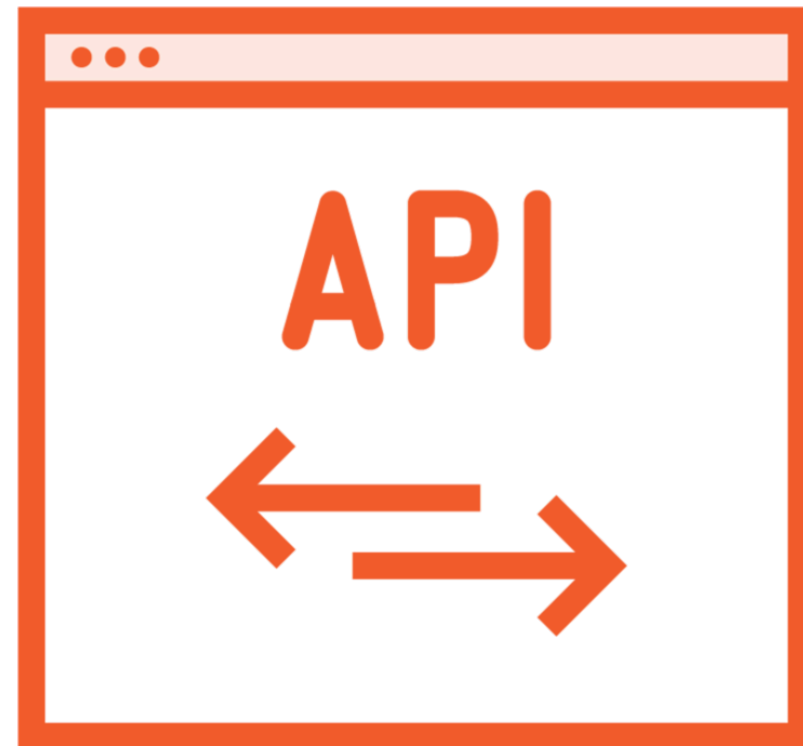
# Investigation Workflows



# Investigation Workflows



# Interacting with REST APIs



## Interacting with other services via a REST API

- Retrieve data that can be used for enriching log data
- Send results to another service for reporting or further analysis

## Two important aspects

- Format of the data being sent and received
- Module for initiating connections



# Working with JSON

{JSON}

**REST APIs requires serialized data**

- JSON is a widely used format

**Python has built-in support via JSON module**

- Easy conversion for lists and dictionaries



```
# import Json module
import json

# convert a dictionary to a Json encoded string
json_data = json.loads(<dict>)

# convert a Json string to an object
raw_data = json.dumps(json_data)
```

## Working with JSON

# Python and REST APIs



## **Implement with standard libraries**

- Requires attention around authentication details

## **Two popular modules available**

- Urllib
- Requests

```
# import requests module
import requests

# make a get request and save response
r = requests.get('url')

# response in json format
r.json()
```

## Leveraging Python Requests

# Working with a Database



## **Leverage a database**

- Store newly processed data
- Extract information from DB to further enrich current log analysis

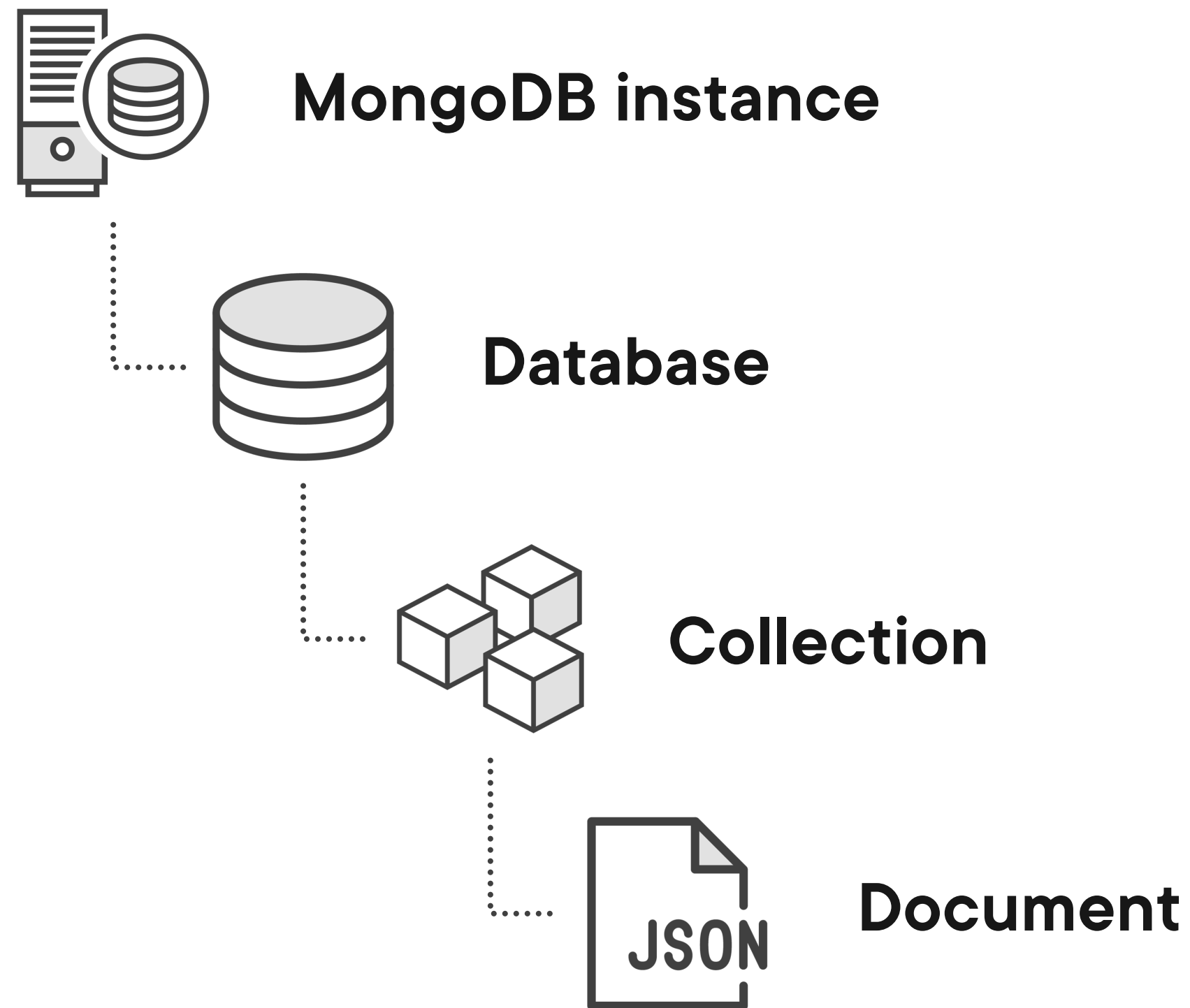
## **Interact with relational databases using an Object Relational Mapper**

- SQLAlchemy for SQL databases

## **MongoDB provides native client**



# MongoDB Hierarchy



# MongoDB Hierarchy



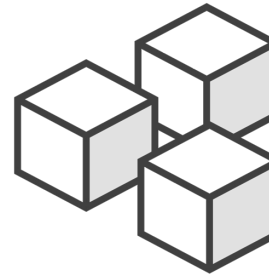
**MongoDB instance**

Log DB



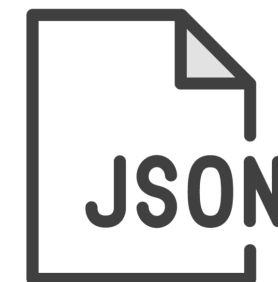
**Database**

Log type



**Collection**

Individual log  
entries



**Document**



```
# import pymongo module
from pymongo import MongoClient

# connect to the MongoDB instance
client = MongoClient()

# select database
db = client['test-database']

# select a collection
collection = db['test-collection']
```

## Getting Started with Pymongo

```
# Get the collection
logs = db['logs']

# Insert a document into the collection
logs.insert_one({...})

# Retrieve the first document that matches the query
logs.find_one({...})
```

## Getting Started with Pymongo

# Demo



**Set up a Mongo database using Docker**

- Connect from Python using Pymongo

**Generate alerts based on common ransomware extensions identified in SMB logs**



# [Demo 4.1] Script



# Working with Elasticsearch



**The elastic stack is a very popular framework**

- Initially designed for distributed search
- Adopted for log analytics

**Python client available to interact with a cluster directly from a script**

- Elasticsearch

```
from elasticsearch import Elasticsearch

# Establish the connection with the Elastic cluster
es = Elasticsearch()

# Add a document to an index
es.index(index="auth", document={})

# Search an index
es.search(index="auth", query={"match_all": {}})
```

## Elasticsearch and Python



# Demo



**Set up a single-node Elasticsearch instance using docker**

**Send enriched log data**

**Build a map visualization**



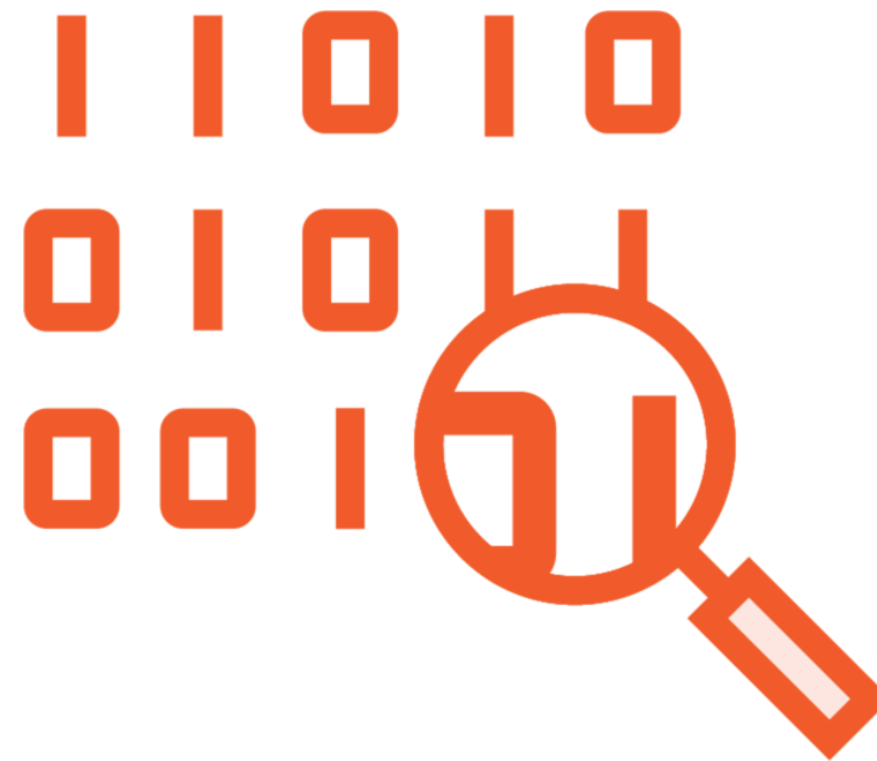
# [Demo 4.2] Script



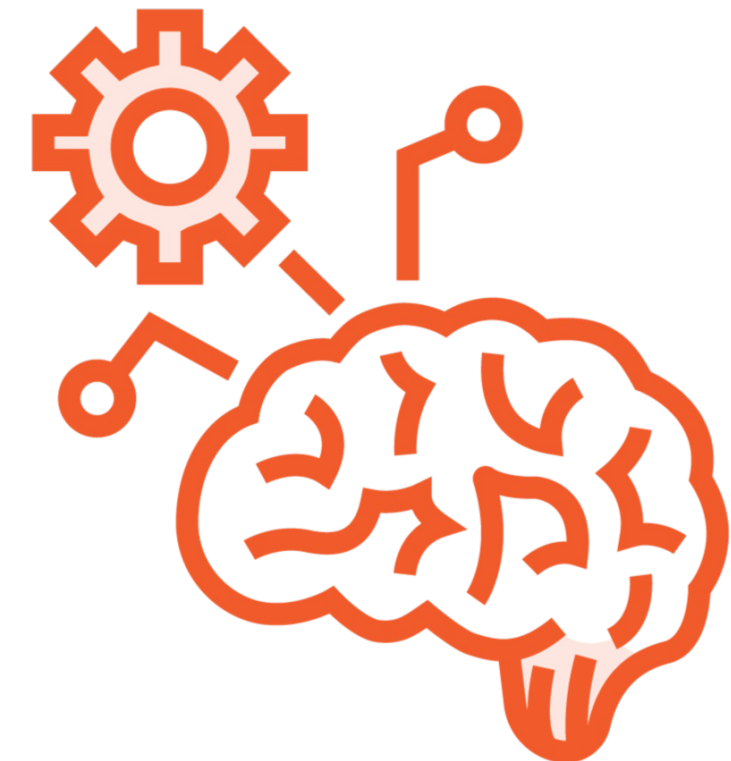
# Develop Yourself Further



**Integrations with  
other tools and  
services**



**Data analysis and  
statistics**



**Machine learning and  
artificial intelligence**



## Summary



**Go beyond single-use scripts to develop log analysis workflows**

**Generated alerts and stored them in a MongoDB for future use**

**Indexed enriched data into Elastic to build visualizations**

**Thank you!**

