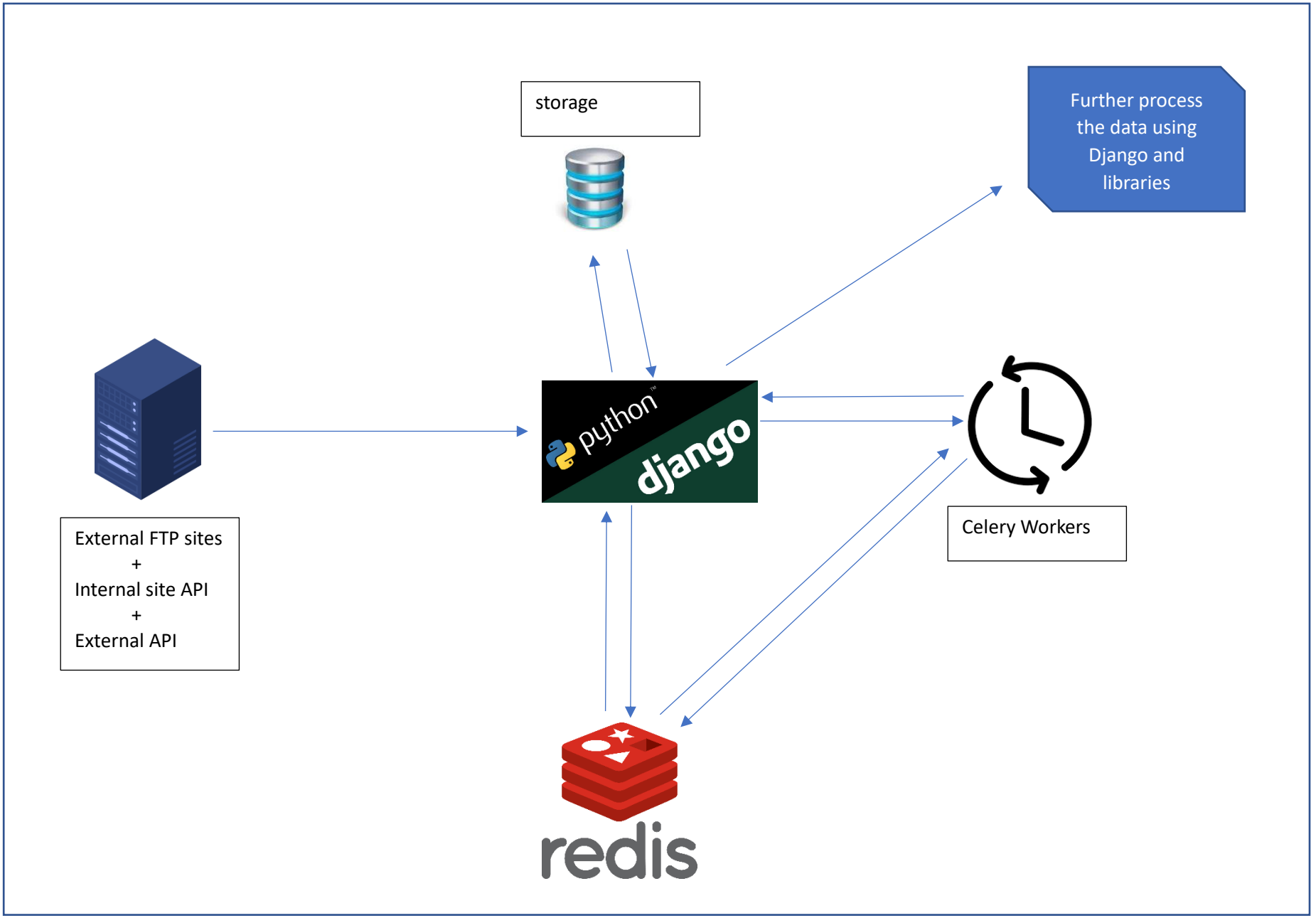
storage
Further process the data using Django and libraries
python
django
External FTP sites
+
Internal site API
+
External API
Celery Workers
redis

Django: Django is a high-level web framework written in Python that encourages rapid development and clean, pragmatic design. I suggest using this framework as it will make the process fast, configuration will be easy. We will be easily managing the different type of connections to different APIs and servers. Connecting to redis, scheduling tasks etc. With the help of this framework we will be able to use different python libraries effectively as compared to using core python.

Django Celery is a powerful distributed task queue system that allows to execute asynchronous tasks in Django application. It's commonly used for handling time-consuming or background tasks, such as sending emails, processing data, or interacting with external APIs. Here are some common use cases and features of Django Celery:

1. **Asynchronous Task Execution:**
    - Offload time-consuming tasks from the request-response cycle to improve the responsiveness of Django web application.
    - Execute tasks asynchronously in the background, allowing application to continue handling requests while the tasks are processed.
2. **Periodic Tasks:**
    - Schedule periodic tasks to run at specified intervals, such as cron-like functionality.
    - Useful for tasks like data cleanup, report generation, or other recurring background processes.
3. **Task Retry and Error Handling:**
    - Celery provides mechanisms for retrying failed tasks automatically with configurable retry policies.
    - Handle errors gracefully and implement custom error-handling logic for tasks.
4. **Distributed Task Queue:**
    - Scales application by distributing tasks across multiple worker processes or even multiple servers.
    - Celery supports multiple message brokers, such as RabbitMQ, Redis, and others, to facilitate communication between Django and Celery workers.
5. **Concurrency Control:**
    - Control the concurrency of tasks by specifying the number of worker processes and threads.
    - Adjust concurrency settings based on the nature of tasks and system resources.
6. **Chaining and Grouping Tasks:**
    - Chain multiple tasks together to create a sequence of steps.
    - Group tasks to execute them in parallel and aggregate the results.
7. **Task Result Storage:**

- Store task results in a backend (such as a database or cache) for retrieval by the application.
- Retrieve task results asynchronously or synchronously.

8. **Monitoring and Logging:**
    - Monitor the status and progress of tasks using tools like Flower (a real-time web-based monitoring tool for Celery).
    - Celery integrates with Django logging for better visibility into task execution.

Redis: Redis is primarily an in-memory database, it offers options for persistence. We can configure Redis to periodically write data to the storage system, ensuring that data is not lost in the event of a system restart. We will use the redis for running celery for scheduled tasks to fetch data from external / internal servers/API's.

Conversion from XML to JSON in Python: We can use libraries xmltodict or xmljson to make the process easier. These tools simplify the conversion process and handle various complexities associated with XML and JSON data structures.