Team

- Git repository for each project, each with its own Dockerfile;
- Docker compose. **BONUS:** common repository with git submodules pointing to each separate project.

Services

- Each service has a database. At least 1 with SQL DB and 1 with NOSQL DB (https://www.xplenty.com/blog/the-sql-vs-nosql-difference). **BONUS:** Add an adapter with a unified interface that will build DB calls based on DB type;
- Tasks are distributed, across multiple requests. Example:

```
- POST /calculations { "type": "addition", "init": 39 }  => OK { "id": "123", "status": "building" }

- PUT /calculations/123 { "term": 12 } => OK { "id": "123", "status": "building" }

- PUT /calculations/123/finalize => OK { "id": "123", "status": "processing" }

- GET /calculations/123 => OK { "id": "123", "status": "done", "result": 51 }
```

- **Status endpoint** to show how many tasks are currently processing;
- **Limit the number of tasks** that can be processed concurrently. Return errors for new tasks if no resources are available;
- **Service discovery:** upon start, services will register themselves with the gateway;
- **BONUS:** add a **priority system**. Some resources should be saved for high priority tasks;
- **BONUS:** add **timeouts** for tasks. Kill a task once it has been processing for too long;
- **BONUS:** use **RPC** (https://www.smashingmagazine.com/2016/09/understanding-rest-and-rpc-for-http-apis/) for internal calls instead of REST;
- **BONUS:** unit testing for each endpoint/function of the service.

Gateway

- **Round Robin** load balancing. **BONUS:** load balancing based on service load;
- **Service discovery:** add a service registry (use cache for storage). Load balancer will pick from registered services when making a decision;

- **Circuit Breaker:** wrap service calls with a circuit breaker. If a call to the service fails or times out, the breaker should be tripped. Details: https://martinfowler.com/bliki/CircuitBreaker.html. **BONUS:** Remove service from service cache once a threshold is reached.
- Outbound API should be **REST**. **BONUS:** use **RPC** for internal calls instead of REST.

Cache

OPTIONAL: Use Redis if team has 3 members

- **In memory storage;**
- Communication should be done directly through a keep-alive **socket**;
- Support multiple simultaneous connections;
- Implement query language (ref https://gist.github.com/LeCoupa/1596b8f359ad8812c7271b5322c30946):
  - SET
  - SETNX
  - GET
  - MGET
  - DEL
  - **BONUS:** EXPIRE
  - **BONUS:** TTL