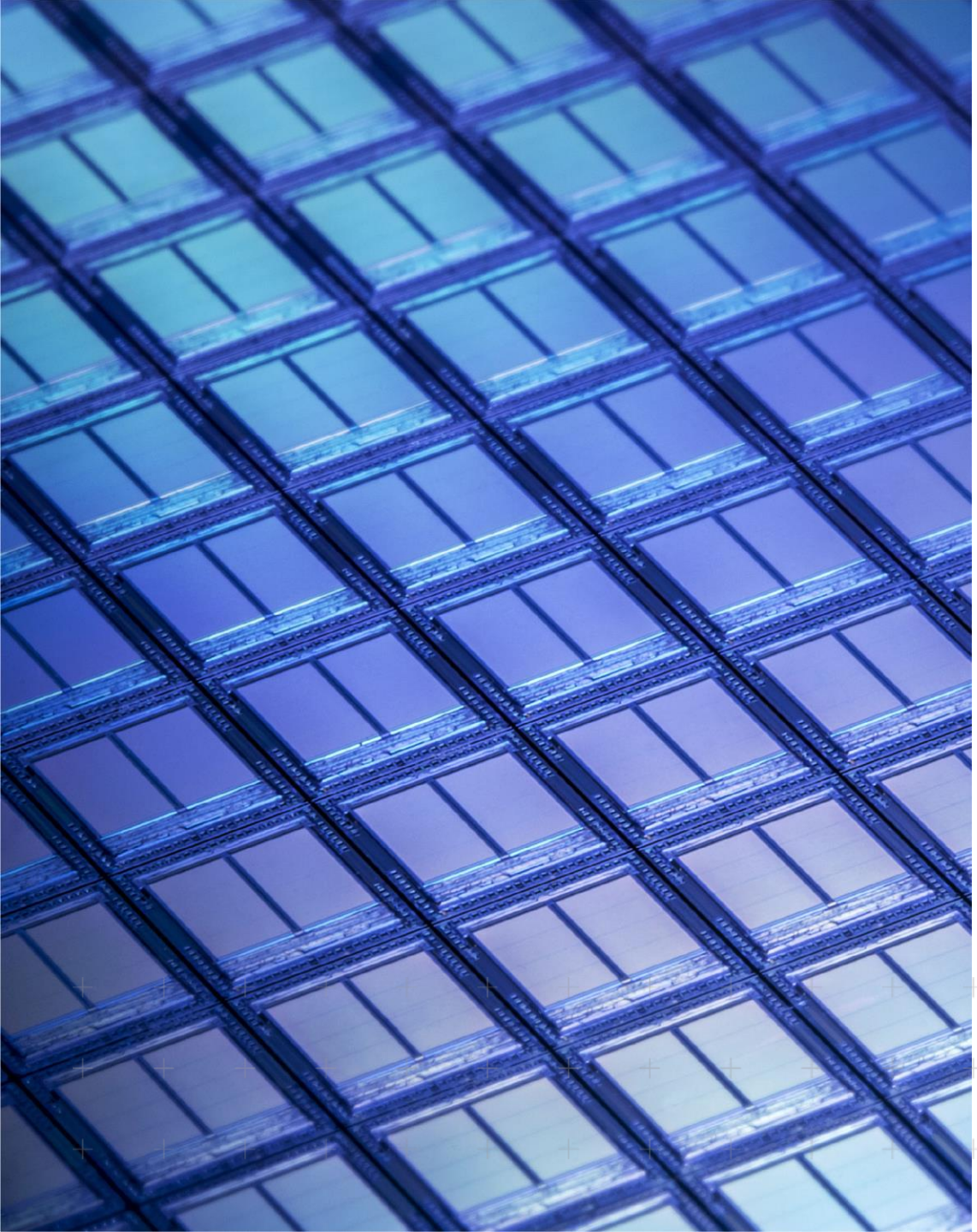


Cloud Object storage

Dec 2019

Problem Statement:

Design and Implement a distributed cloud storage.

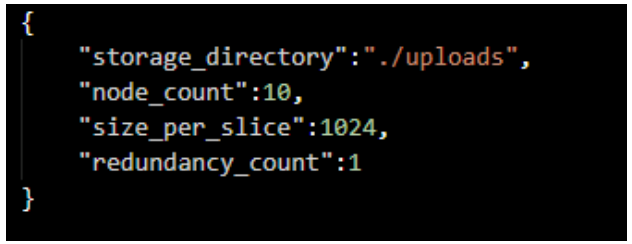
Similar to Google drive, where user can store and retrieve any Objects.

What is an Object?

- Any BLOB - like txt, jpeg, mp4 etc…
- Which could be stored and retrieved on demand for anywhere

Basic Functionalities

- One web server
- REST API
- Config file:



```
{
    "storage_directory":"./uploads",
    "node_count":10,
    "size_per_slice":1024,
    "redundancy_count":1
}
```

- PUT
  - Store an incoming file and return the ID of the resource
- GET
  - Download the file for a given <ID>
- LIST
  - List the files stored in the server
- Delete
  - Delete the file for a given <ID>

API documentation

For Detailed documentation: https://praveenkumars.docs.apiary.io/

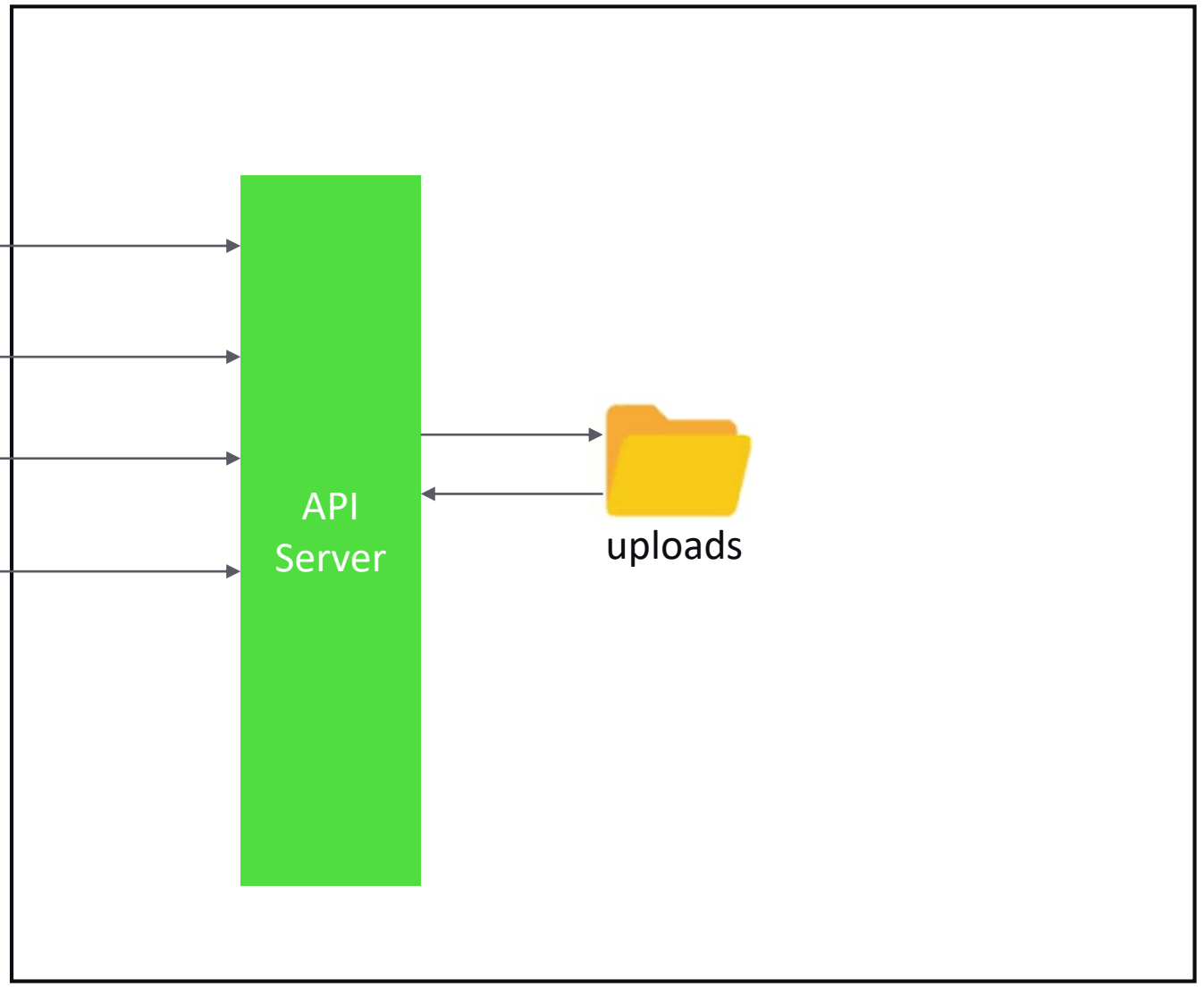
Description	Request	Response
Upload a File	PUT: http://localhost:5000/files ----WebKitFormBoundary7MA4YWxkTrZu0gW Content-Disposition: form-data; name="file"; filename="/D:/Test/TestData/test_file_1.txt" Content-Type: text/plain	Content-Type: text/plain 5e1b3c4c-27f0-11ea-9a34-a4c3f0b4a7fe <id> of the resource is returned
Download a File	GET: http://localhost:5000/files/{id}	The File gets downloaded
Delete a File	DELETE: http://localhost:5000/files/{id}	Content-Type: text/plain object 5e1b3c4c-27f0-11ea-9a34-a4c3f0b4a7fe deleted successfully
List all Files	GET: http://localhost:5000/files/list	Content-Type: application/json [{ "file_name": "test_file_1.txt", "id": "ef5d2258-27cd-11ea-8e88-a4c3f0b4a7fe" }, { "file_name": "test_file_2.txt", "id": "ef6531d4-27cd-11ea-a4b7-a4c3f0b4a7fe" }

Constraints

- Client server architecture
  - Could be out of proc – stream/queue - REST API
  - Client sends data to be stored on server
- No public database allowed :
  - Expectation is to build their own customized data store
  - Sql , sqlite, redis, caching libraries are not allowed
- Can borrow code snippet / libraries from internet to build your own solution
- Follow schema and URL schemes as per the API documentation
- Follow folder naming conventions as per the documentation
- Postman collection import link:
  https://www.getpostman.com/collections/76e732538fe48439bfc2

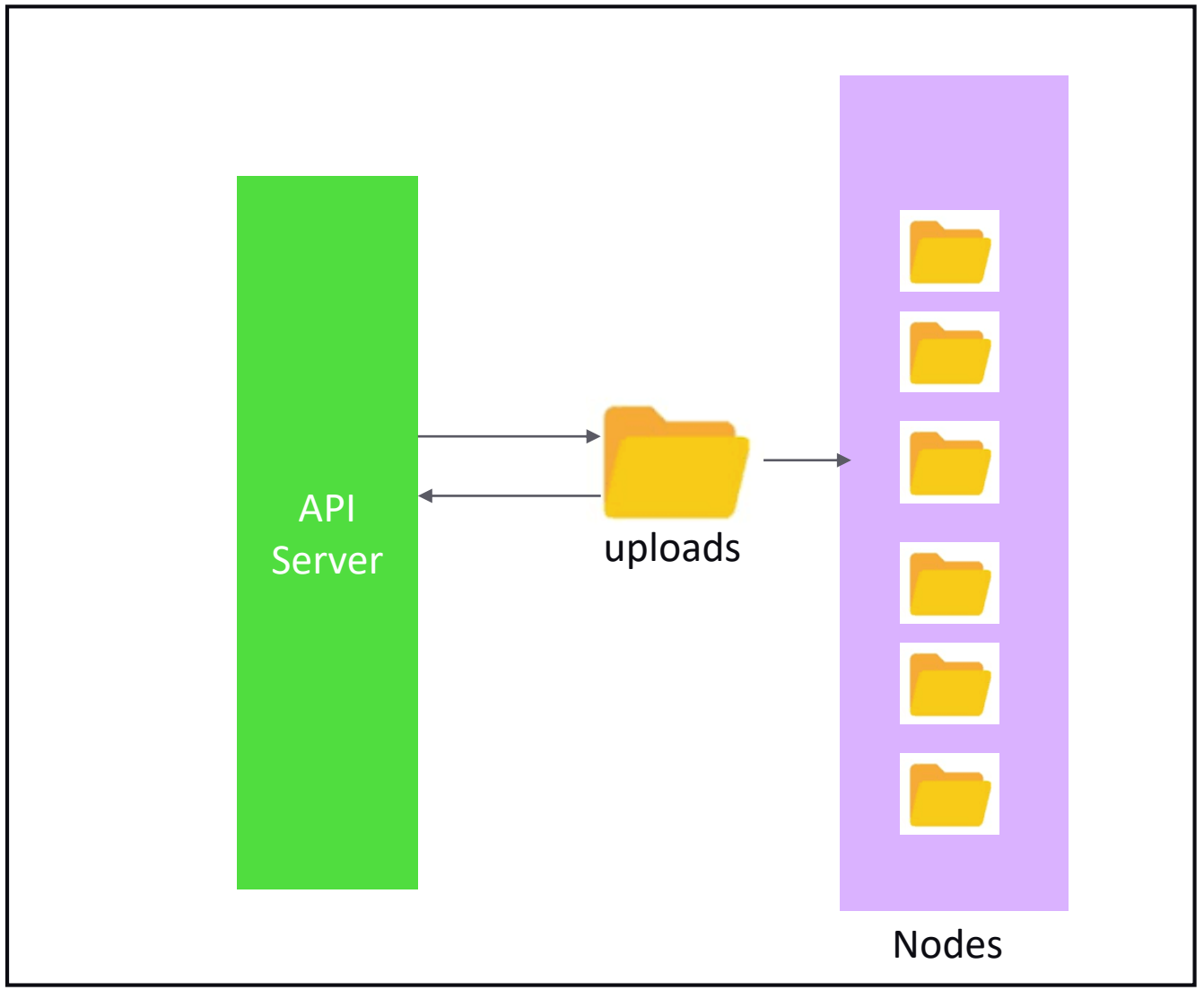
Deliverables - Milestone #1

- Goal: Basic API Implementation
  - Ability to Upload a file to the server
  - Download the file
  - List all the files on the server
  - Delete a file



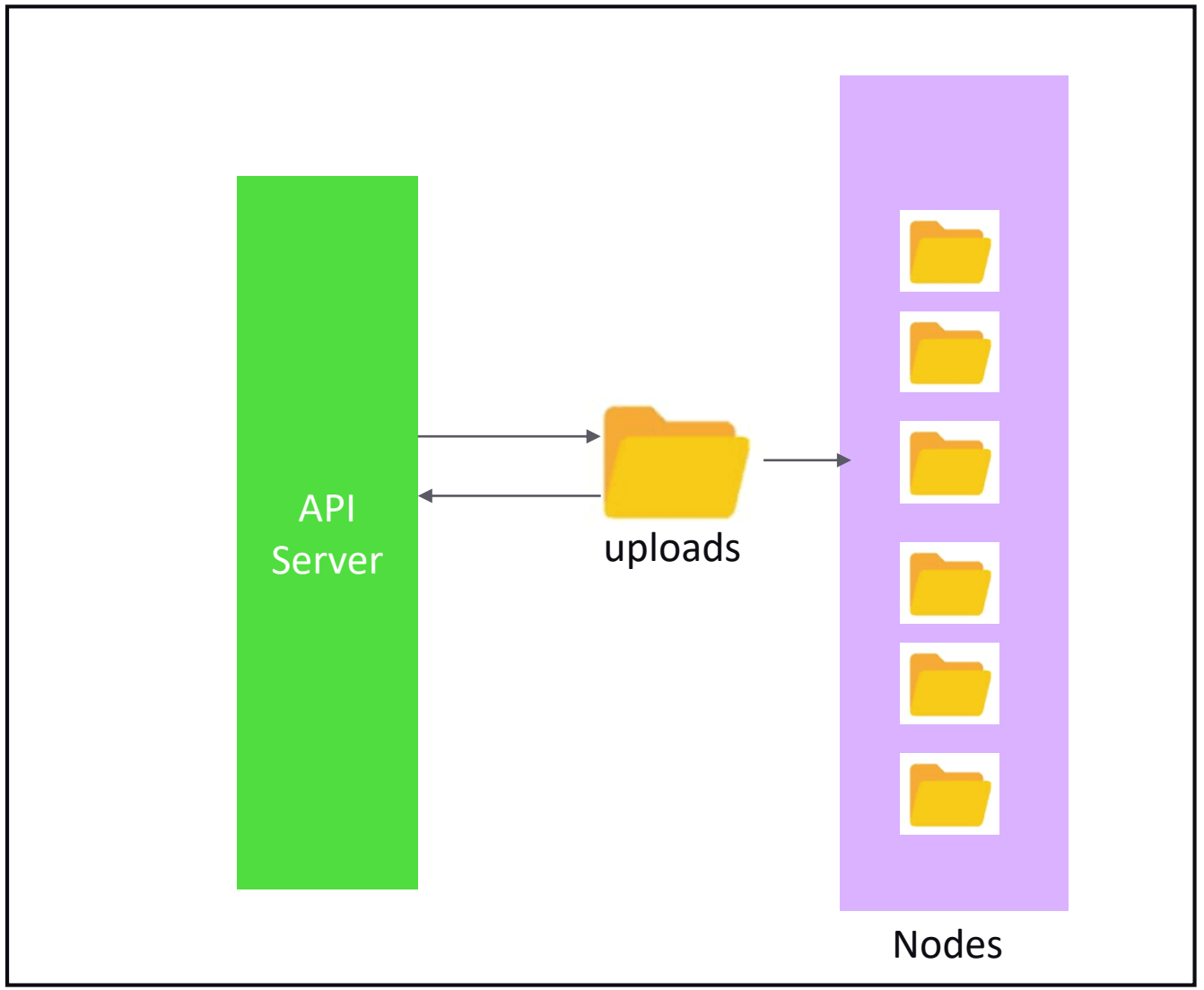
Deliverables - Milestone #2

- Goal: Milestone#1 + Load Balancing
  - Uploaded file must be broken down in to chunks as mentioned on the config
  - Nodes (folders) must be created as per the node count on config
  - Nodes must be named as node_<number> Eg. node_1
  - File chunks must be moved to the nodes with load balancing. Node with the least number of files must be filled first
  - Metadata file(s) must be created to save information on the file chunks and their location



Deliverables - Milestone #3

- Goal: Milestone#2 + Redundancy
  - When one or more nodes go down, the file must still be retrievable
  - Redundancy_count = 1 value in config means, file must be retrievable when 1 node goes down.
  - Redundancy level must increase as per the value specified on the config



Instructions

- **Open Book format, welcome to use internet resources**
- **Prize will be awarded based on the following**
  - **Interactive review of your design approach**
  - **Result Validation**
  - **Solution completeness and performance**
  - **Original code contribution (vs. Library used)**
- **Participants are encouraged to work individually and refrain from helping other participants**
- **Participants actively engaged and present through out workshop will be awarded certificates of participation**

Instructions II

- **Practice the Primer Program to understand REST API implementation basics**
- **Make sure that the API server resets on a restart. Such that all files and nodes are deleted and the API server becomes new during a restart operation**
- **Make sure you use the attached Postman collection for testing your API**
- **Run tests using test validator with your roll number**
- **Do not run the validator unless you have some change to test**
- **Create a storage directory as mentioned in the config**
- **Nodes must have node_<number> as name. Eg. node_1, node_2**
- **Make sure to update the test config with the storage locations before running milestone-3 test**

Thank you