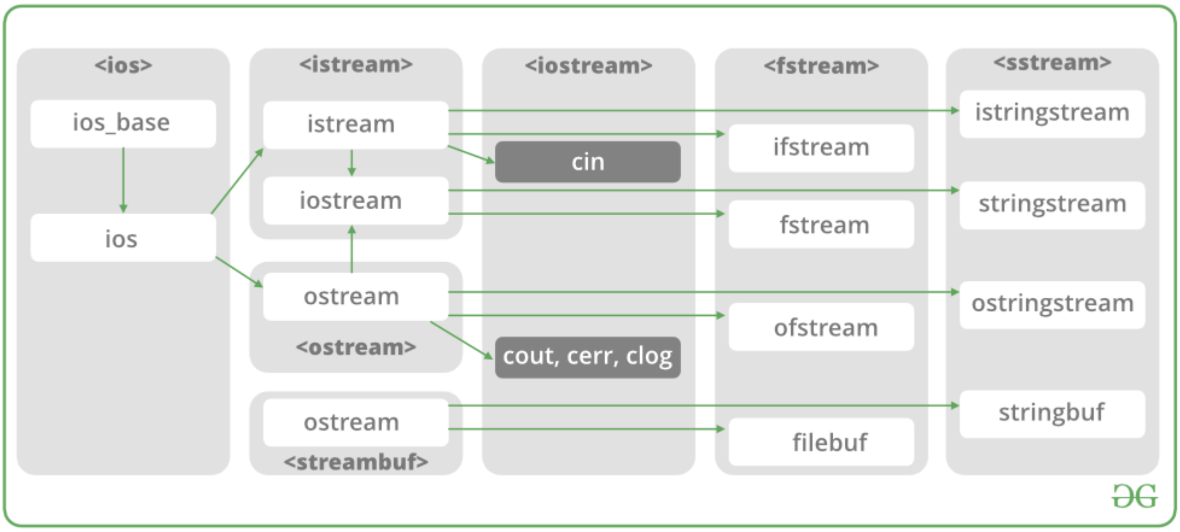
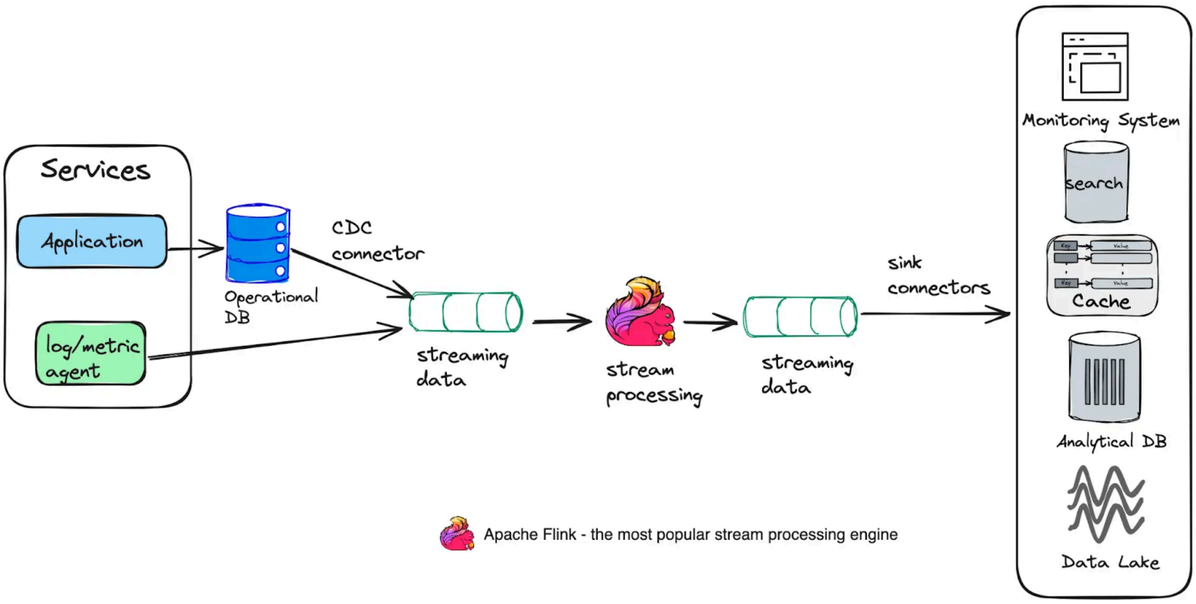
🎮

Streams and File Handling



✅ 1. Introduction to Streams in C++

👉 What is a Stream?



Apache Flink - the most popular stream processing engine

- A **stream** is a flow of data (like water flowing through a pipe).

- In C++, streams are used to perform **input** (reading) and **output** (writing).

Stream Type	Description	Example
Input Stream	Data flows **into** the program	Reading from a file or keyboard
Output Stream	Data flows **out of** the program	Writing to console or file

✅ 2. Files and Streams in C++

C++ provides file handling support via **fstream** library:

```
#include <fstream>
```

Three classes are provided:

Class	Description
ifstream	Input file stream (read from files)
ofstream	Output file stream (write to files)
fstream	Input + Output (read and write)

✅ 3. How Do Streams Work?

- Streams use **buffers**: temporary storage between your program and device (keyboard, file, screen).
- Data is read/written from/to these buffers efficiently.

Example of Stream Operation:

```
ofstream fout("data.txt");   // Creates a stream 'fout' linked to file
fout << "Hello, World!";     // Writes data to file
fout.close();                // Closes the stream
```

✅ 4. Input/Output Stream Classes

📌 Hierarchy of IOStream classes

```
         ios
        /  \
   istream     ostream
      |           |
  ifstream     ofstream
      \         /
       fstream
```

📌 Common Functions:

Function	Description
`open()`	Opens a file
`close()`	Closes a file
`eof()`	Returns true if end of file is reached
`fail()`	Returns true if a non-fatal error occurs
`bad()`	Returns true if a serious error occurs
`good()`	Returns true if no error

✅ 5. Error Handling with IO Streams

Function	Usage Example	Meaning
`.eof()`	`while(!fin.eof())`	End of file reached
`.fail()`	`if(fin.fail())`	Logical error
`.bad()`	`if(fin.bad())`	Physical error
`.good()`	`if(fin.good())`	Everything is good

Example:

```
ifstream fin("file.txt");
if(!fin) {
    cout << "File could not be opened!";
}
```

✅ 6. Lower-Level Streams and Memory Buffers

- Streams interact with **memory buffers** (temporary storage).
- Buffer stores data temporarily before writing to or reading from files.
- Example: `fstream` uses a buffer to collect data before writing the whole block to the disk for efficiency.

✅ 7. Reading and Writing to Files (Basic Examples)

📌 Writing to a file:

```
ofstream fout("example.txt");
fout << "Hello File!";
fout.close();
```

📌 Reading from a file:

```
ifstream fin("example.txt");
string word;
fin >> word;
cout << word;
fin.close();
```

📌 Reading full lines:

```
string line;
while (getline(fin, line)) {
    cout << line << endl;
}
```

✅ 8. File Modes in C++

Mode	Description
ios::in	Open for reading
ios::out	Open for writing
ios::app	Append data

ios::binary	Open in binary mode
ios::ate	Move to the end of file on open

Example:

```
ofstream fout("data.txt", ios::app);
```

✅ 9. Parsing File Data (String Manipulation)

When reading structured data from a text file (CSV style):

```
101,John Doe,CS
102,Alice,Math
```

You can use **getline() + string manipulation**:

```
getline(fin, line);
size_t pos1 = line.find(",");
size_t pos2 = line.rfind(",");
```

✅ Use **istringstream** for easy splitting:

```
istringstream iss(line);
string token;
while(getline(iss, token, ',')) {
    cout << token << endl;
}
```

✅ 10. Object-Oriented File Handling (Practical Project)

🎯 *Final Project*: Student Registration System

Features:

- Add student details
- Search student by Roll No

- View all students
- Store data in **students.txt**

Sample Stored File Format:

```
101,John Doe,Computer Science
102,Alice,Mathematics
```

Solution:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
using namespace std;

class Student {
private:
    int roll;
    string name;
    string course;

public:
    Student() {}
    Student(int r, string n, string c) : roll(r), name(n), course(c) {}

    int getRoll() const { return roll; }
    string getName() const { return name; }
    string getCourse() const { return course; }

    void inputStudent() {
        cout << "Enter Roll No: ";
        cin >> roll;
        cin.ignore();
        cout << "Enter Name: ";
        getline(cin, name);
        cout << "Enter Course: ";
        getline(cin, course);
    }

    void display() const {
        cout << "Roll No: " << roll << " | Name: " << name << " | Course: " << co
    }
};

// File handling class
class StudentDatabase {
private:
    string filename;

public:
    StudentDatabase(string fname) : filename(fname) {}

    // Add Student (write in text format)
    void addStudent(const Student& s) {
        ofstream fout(filename, ios::app);  // Text mode
        if (fout.is_open()) {
            fout << s.getRoll() << "," << s.getName() << "," << s.getCourse() << "\
            fout.close();
            cout << "Student added successfully!\n";
        } else {
            cout << "Error opening file.\n";
        }
    }

    // Get Student by Roll No
    bool getStudentByRoll(int roll) {
        ifstream fin(filename);
        string line;
        bool found = false;
        while (getline(fin, line)) {
            int r;
            string n, c;
            size_t pos1 = line.find(",");
            size_t pos2 = line.rfind(",");
            r = stoi(line.substr(0, pos1));
            n = line.substr(pos1 + 1, pos2 - pos1 - 1);
            c = line.substr(pos2 + 1);

            if (r == roll) {
                cout << "Roll No: " << r << " | Name: " << n << " | Course: " << c <<
                found = true;
                break;
            }
        }
        fin.close();
        if (!found) cout << "Student with Roll No " << roll << " not found.\n";
        return found;
    }

    // Get all students
    void getAllStudents() {
        ifstream fin(filename);
        string line;
        cout << "\n----- Student List -----\n";
        while (getline(fin, line)) {
            int r;
            string n, c;
            size_t pos1 = line.find(",");
            size_t pos2 = line.rfind(",");

            r = stoi(line.substr(0, pos1));
            n = line.substr(pos1 + 1, pos2 - pos1 - 1);
            c = line.substr(pos2 + 1);

            cout << "Roll No: " << r << " | Name: " << n << " | Course: " << c << e
        }
        fin.close();
    }
};

// Menu driven program
int main() {
    StudentDatabase db("students.txt");
    int choice;
    do {
        cout << "\n===== Student Registration System =====\n";
        cout << "1. Add Student\n";
        cout << "2. Get Student by Roll No\n";
        cout << "3. Display All Students\n";
        cout << "4. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;
        switch (choice) {
            case 1: {
                Student s;
                s.inputStudent();
                db.addStudent(s);
                break;
            }
            case 2: {
                int roll;
                cout << "Enter Roll No to search: ";
                cin >> roll;
                db.getStudentByRoll(roll);
                break;
            }
            case 3:
                db.getAllStudents();
                break;
            case 4:
                cout << "Exiting program...\n";
                break;
            default:
                cout << "Invalid choice. Try again.\n";
        }
    } while (choice != 4);

    return 0;
}
```