DigitalSkola

LEARNING PROGRESS REVIEW "WEEK 9"

BY : DREAM (KELOMPOK 1)

Data Realm Engineers And Maestros

Anggota Kelompok

01

Afroh Fauziah

02

Althaf Nawadir Taqiyyah

03

Andi Rosilala

04

Andrew Bintang Pratama

05

Andrew Fortino Mahardika Suadnya

JAVA

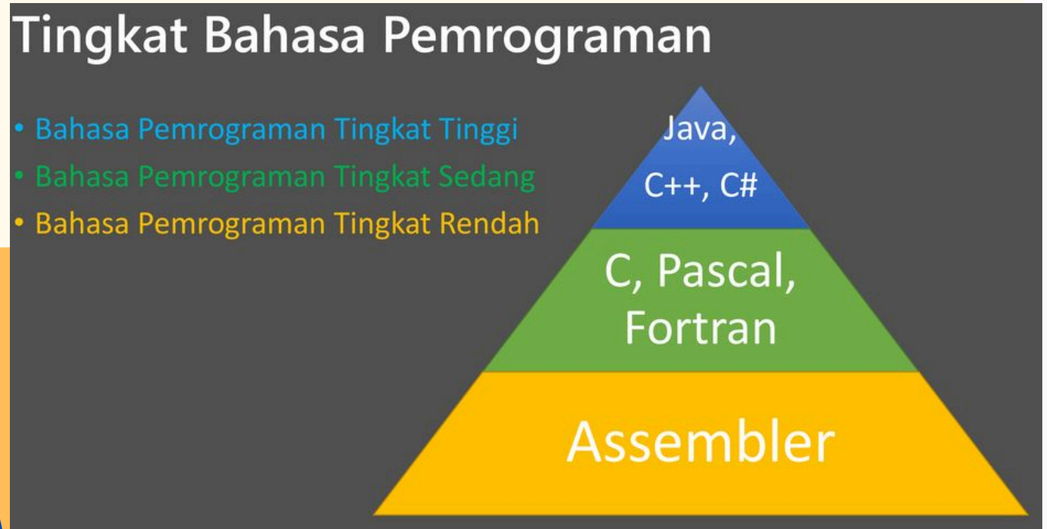
Introduction & History

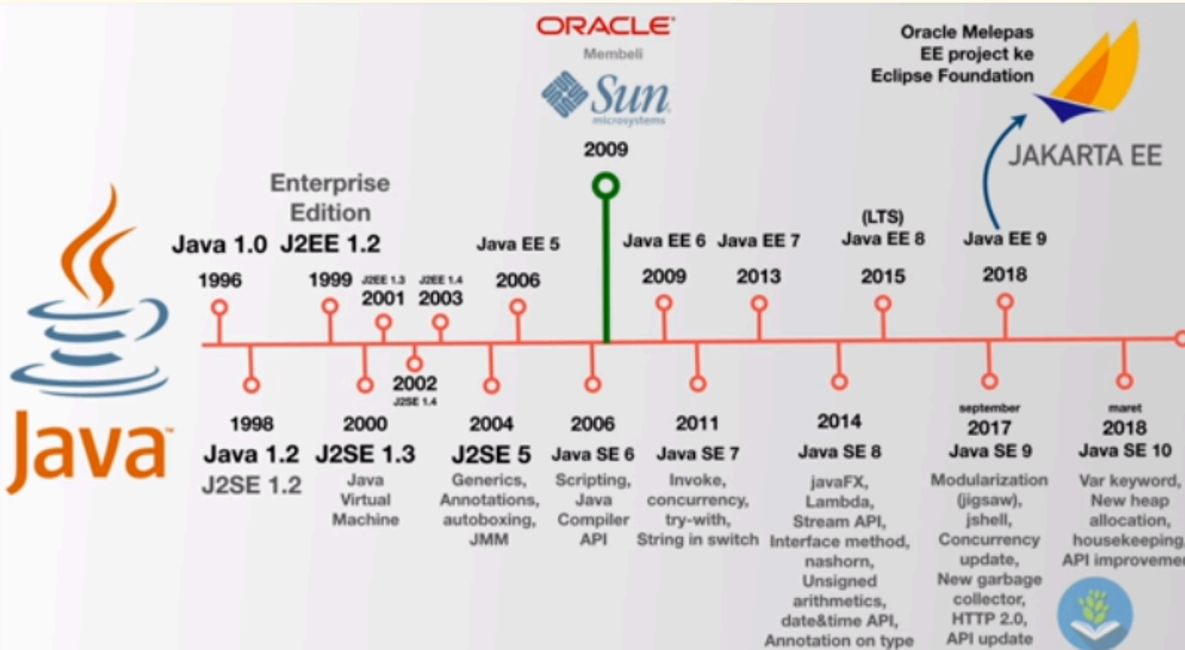
JAVA dikembangkan oleh James Gosling di Sun Microsystems tahun 90an, kemudian diakuisisi oleh Oracle Corporation.

CIRI :

- bahasa pemrograman berorientasi objek
- memiliki ekosistem yang luas
- berbasis kelas
- sintaksis sederhana
- dirancang memiliki ketergantungan implementasi sesedikit mungkin.

karena suka kopi





Java Terminology

Java Virtual Machine(JVM)

- Ada tiga fase eksekusi program: written, compile dan running program.
- Bertanggung jawab sebagai input, sedangkan outputnya menjalankan bytecode Java oleh kompiler dan menerjemahkannya menjadi instruksi yang dapat dijalankan oleh sistem operasi yang mendasarinya.

Bytecode in the Development Process

- Kompiler Javac dari JDK mengkompilasi kode sumber java menjadi bytecode.
- Representasi intermediate dari program Java yang dihasilkan oleh kompilator Java.
- Dieksekusi / dijalankan oleh JVM.
- Disimpan sebagai file .class oleh compiler.

Java Development Kit (JDK)

- Sekumpulan alat untuk mengembangkan perangkat lunak dalam bahasa pemrograman Java mencakup kompilator Java, JRE, debugger java, java docs, dan alat lainnya.
- Perlu diinstall di komputer untuk membuat, mengkompilasi dan menjalankan program java.

Java Runtime Environment (JRE)

- paket perangkat lunak yang menyediakan lingkungan untuk menjalankan aplikasi Java.
- Instalasi JRE di komputer memungkinkan program java berjalan, namun tidak dapat mengompilasinya.
- Mencakup browser, JVM, dukungan applet, dan plugin.

Garbage Collector

- Mengelola memori secara otomatis dengan menghapus objek yang tidak lagi digunakan oleh program.
- Mengingat kembali benda-benda yang tidak dirujuk.
- Menangani manajemen memori.

ClassPath

- Variabel lingkungan yang digunakan oleh JVM untuk menemukan kelas yang diperlukan oleh program Java saat runtime. Berisi daftar direktori dan file JAR yang berisi file kelas.
- Jika ingin menyertakan library eksternal pada JDK, maka harus ditambahkan ke classpath.

Main Features of Java

1. Platform Independent
2. Object-Oriented Programming Language
3. Simple
4. Robust
5. Secure

Basic Terminologies

1. Object
2. Class
3. Method
4. Instance variables

Comments in Java

- Single line Comment

```
// System.out.println("Test!");
```

- Multi-line Comment

```
/* System.out.println("Test!");*/
```

- Documentation Comment. Also called a doc comment.

```
/** documentation */
```

Basic Syntax

Program File Name

Harus sama persis nama kelas dengan ekstensi.java. Contoh :

- Hello.java // valid syntax
- hello.java // invalid syntax

Case Sensitivity

Pengidentifikasi AB, Ab, aB, dan ab harus benar. Contoh :

- System.out.println("Alice"); // valid syntax
- system.out.println("Alice"); // invalid syntax

Class Names

Huruf pertama dari kelas harus dalam huruf besar. Contoh :

- class MyJavaProgram // valid syntax
- class myJavaProgram // invalid syntax

Method Names

Semua method harus di mulai dari Lower Case letter. Contoh :

- public void employeeRecords () // valid syntax
- public void EmployeeRecords () // valid syntax

Identifiers in Java

Semuda Identifiers dimulai dari abjad or an underscore _.

- Legal : MinNumber, total, ak74, hello_world, $amount, _under_value
- Illegal : 74ak, -amount

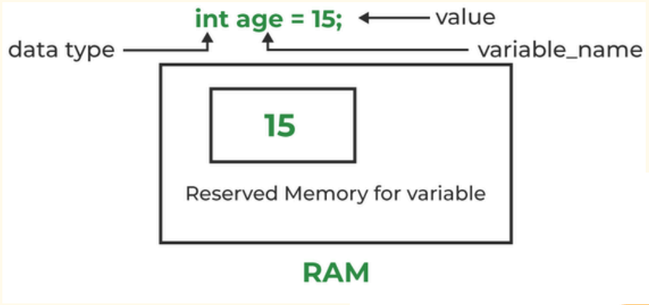
Access Modifiers

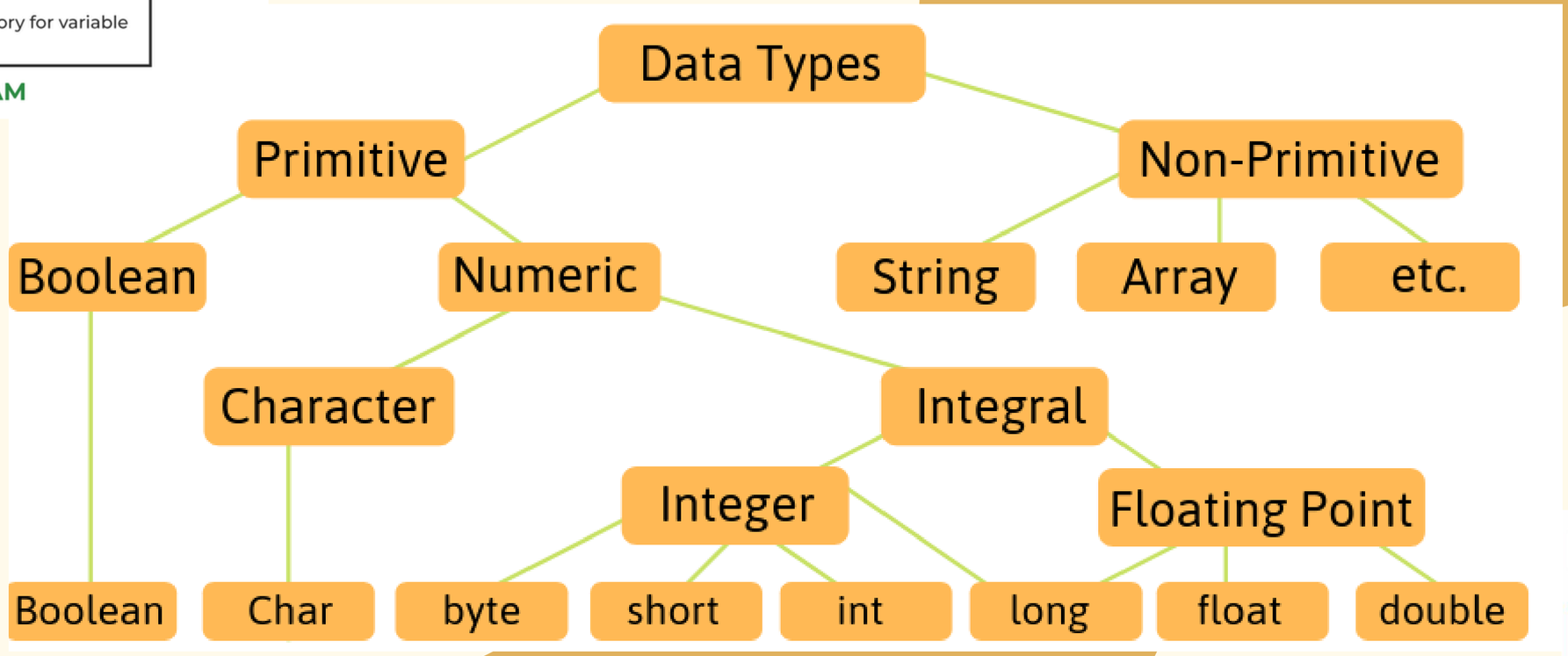
Modifiers ini mengontrol ruang lingkup class dan method.

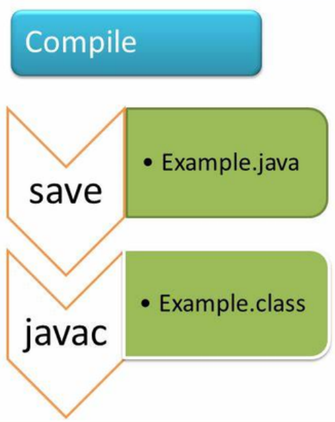
- Access : default, public, protected, private
- Non-access : final, abstract, strictfp

DATATYPE

variable







DigitalSkola
Uncover The World of Digital Skills with Us

DigitalSkola

TEST JAVA DATATYPE

/MyClass.java

nama file dan class sama

```java
public class MyClass {
    public static void main(String args[]) {
      int x=10;
      int y=25;
      int z=x+y;
      int a = z+x+y;
      float  myFloatNum = 5.99f;
      char myLetter = 'D';
      String myText = "Hello";
      String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

      System.out.println("Sum of a = " + a);
      System.out.println("Float = " + myFloatNum);
      System.out.println("char myLetter = " + myLetter);
      System.out.println("cars Array = " + cars[0]);
    }
}
```

Output

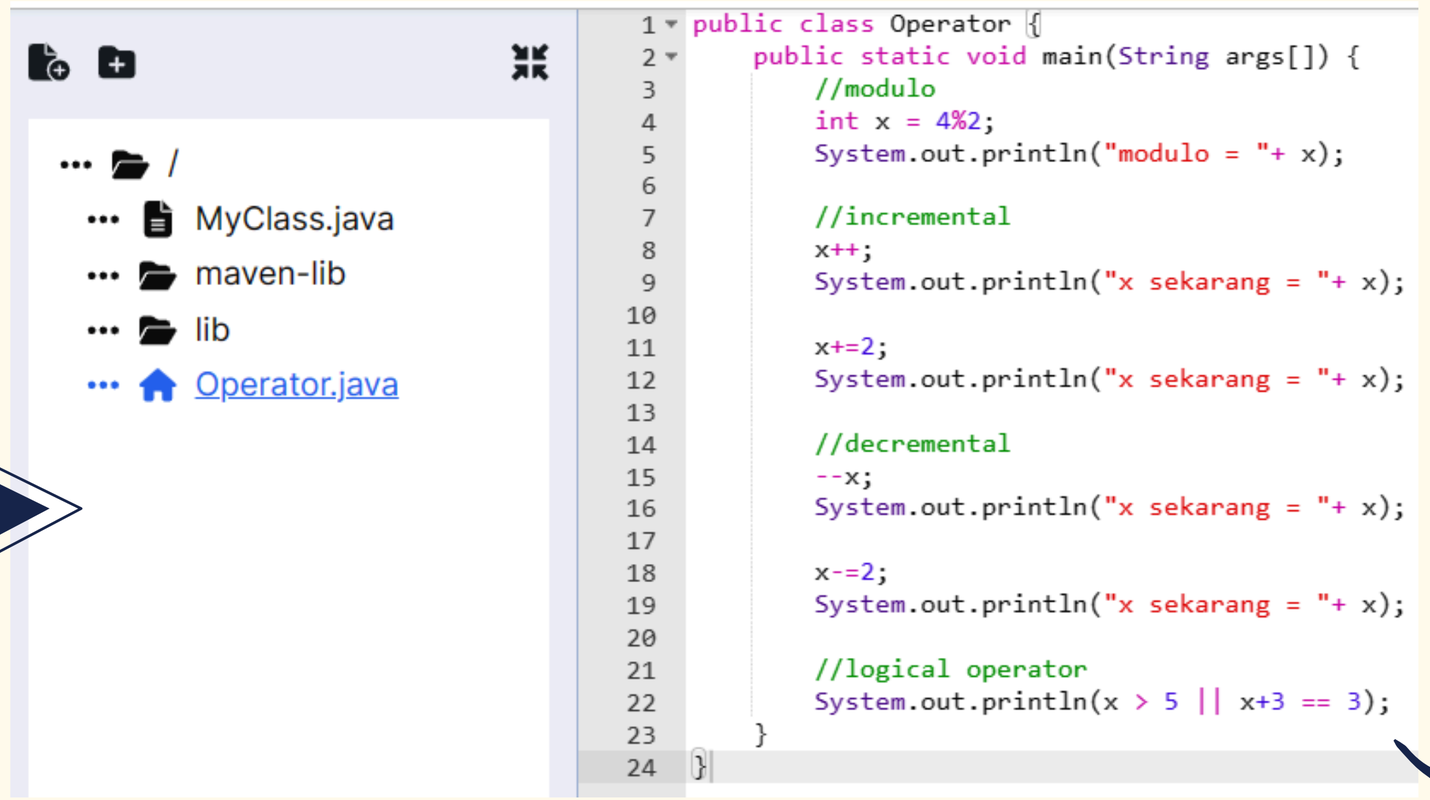
```
Sum of a = 70
Float = 5.99
char myLetter = D
cars Array = Volvo
```

Masukkan / input kode data sesuai syntax yang benar untuk menghasilkan berbagai output diantaranya int, float, char, & string untuk text juga array lalu ditutup dengan system.out.print

OPERATOR

Buat file baru untuk Operator.java lalu ganti base / homenya dengan titik tiga dan pilih ‘Make it a start file’ agar bisa running sesuai file yang diinginkan.

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators




```java
public class Operator {
    public static void main(String args[]) {
        //modulo
        int x = 4%2;
        System.out.println("modulo = "+ x);

        //incremental
        x++;
        System.out.println("x sekarang = "+ x);

        x+=2;
        System.out.println("x sekarang = "+ x);

        //decremental
        --x;
        System.out.println("x sekarang = "+ x);

        x-=2;
        System.out.println("x sekarang = "+ x);

        //logical operator
        System.out.println(x > 5 || x+3 == 3);
    }
}
```

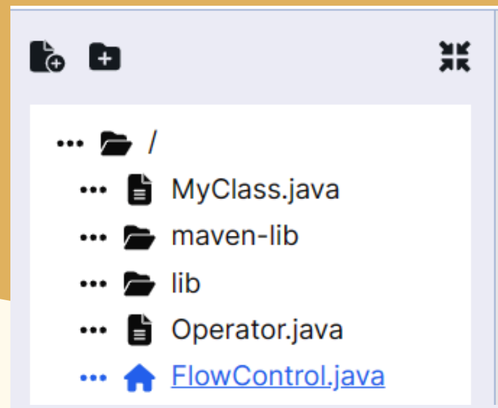
Output

```
modulo = 0
x sekarang = 1
x sekarang = 3
x sekarang = 2
x sekarang = 0
true
```

DigitalSkola

FLOW CONTROL

if-else




```java
public class FlowControl {
    public static void main(String args[]) {
        int x = 10;

        if (x < 10){
            System.out.println("Nilai dibawah 10");
        }
        else if (x == 10) {
             System.out.println("Nilai sama dengan 10");
        }
        else {
            System.out.println("Nilai lebih dari 10");
        }

        System.out.println("Program selesai");

    }
}
```

Output

```
Nilai sama dengan 10
Program selesai
```

switch

```java
public class FlowControl {
    public static void main(String args[]) {
        int x = 10;

        // if else
        if (x < 10){
            System.out.println("Nilai dibawah 10");
        }
        else if (x == 10) {
             System.out.println("Nilai sama dengan 10");
        }
        else {
            System.out.println("Nilai lebih dari 10");
        }

        System.out.println("Program selesai");


        //switch
        int day = 2;
        switch(day){
            case 1:
                 System.out.println("Senin");
                 break;
            case 2:
                 System.out.println("Selasa");
                 break;
            case 3:
                 System.out.println("Rabu");
                 break;
        }
    }
}
```

```
Nilai sama dengan 10
Program selesai
Number Incorrect
```

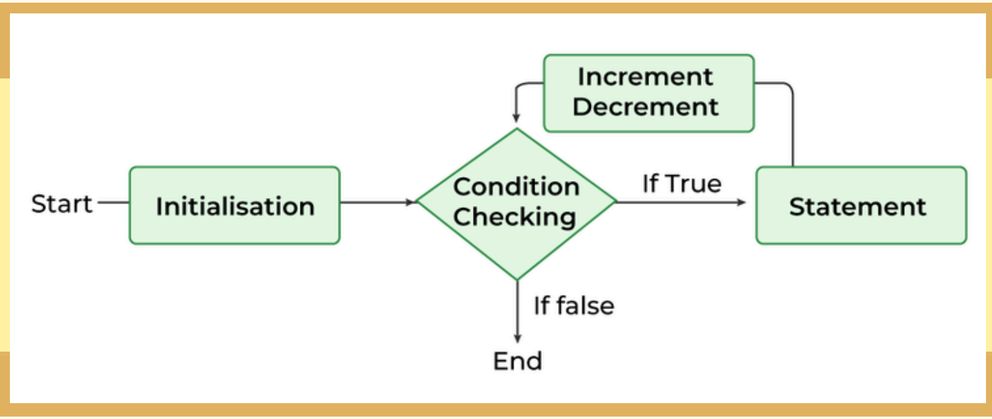
menambahkan jika datanya dari luar / default

```java
default:
    System.out.println("Number Incorrect")
    break;
```

```
Nilai sama dengan 10
Program selesai
Selasa
```

DigitalSkola
Uncover The World of Digital Skills with Us

LOOP

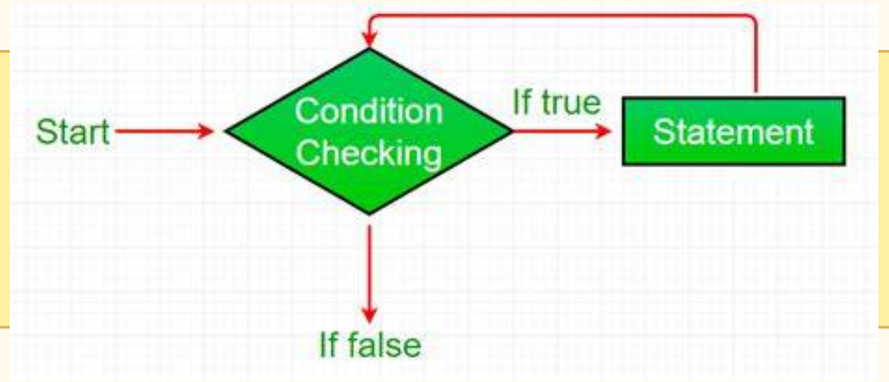


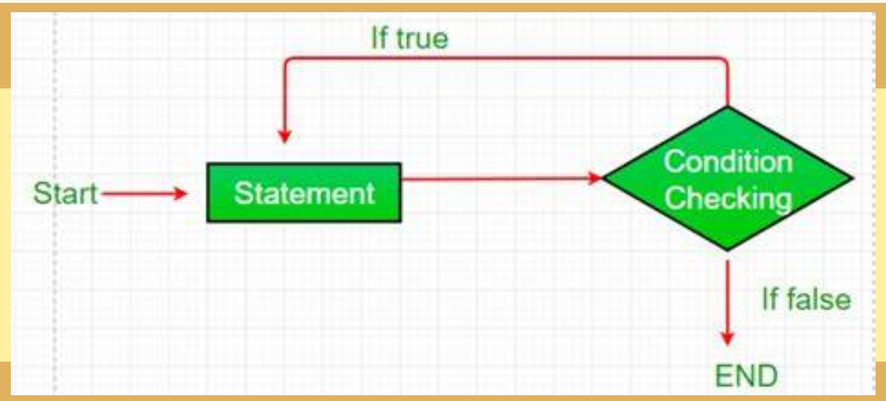
~ for Loop ~

- Digunakan ketika jumlah iterasi sudah diketahui sebelumnya.
- Terdiri dari tiga bagian: inisialisasi, kondisi, dan langkah iterasi.
- Inisialisasi dilakukan sekali sebelum loop dimulai.

~ while Loop ~

- Kondisi terlebih dahulu baru statement.
- Ketika kondisi terpenuhi maka akan berputar terus, jika tidak maka loop berakhir / selesai.
- Digunakan ketika jumlah iterasi belum diketahui.



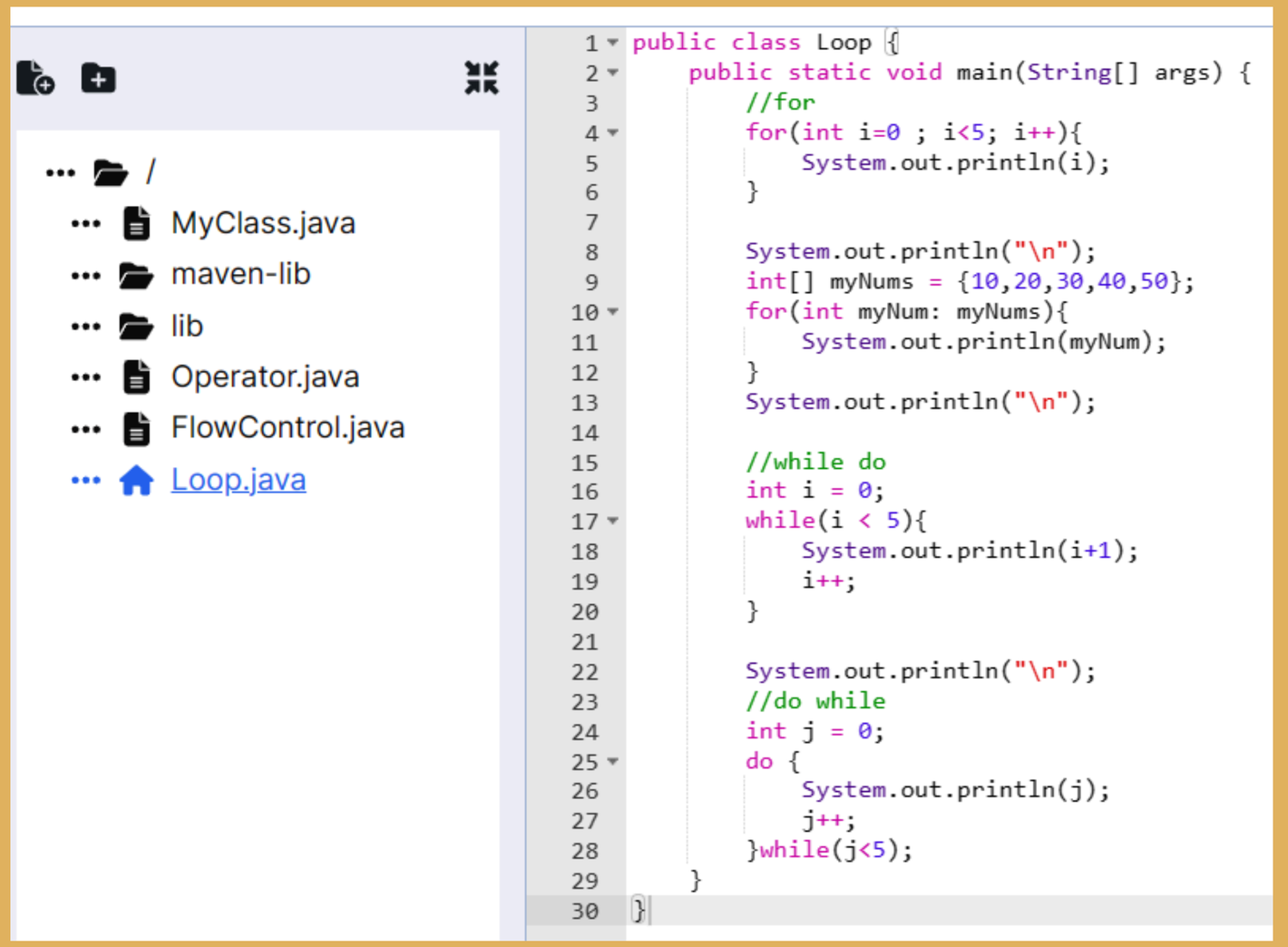


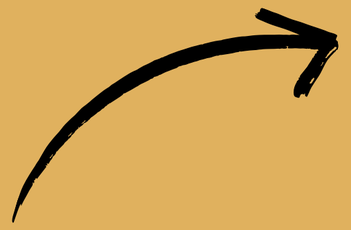
~ do while Loop ~

- Statement terlebih dahulu baru kondisi.
- Ketika sesuai maka akan running terus, jika tidak maka loop akan selesai.
- Menjalankan blok kode minimal satu kali, kemudian melanjutkan jika kondisi terpenuhi.

LOOP

Membuat file baru dengan nama Loop.java lalu masukkan kode syntaxnya untuk for Loop, while Loop, dan do while Loop





Output

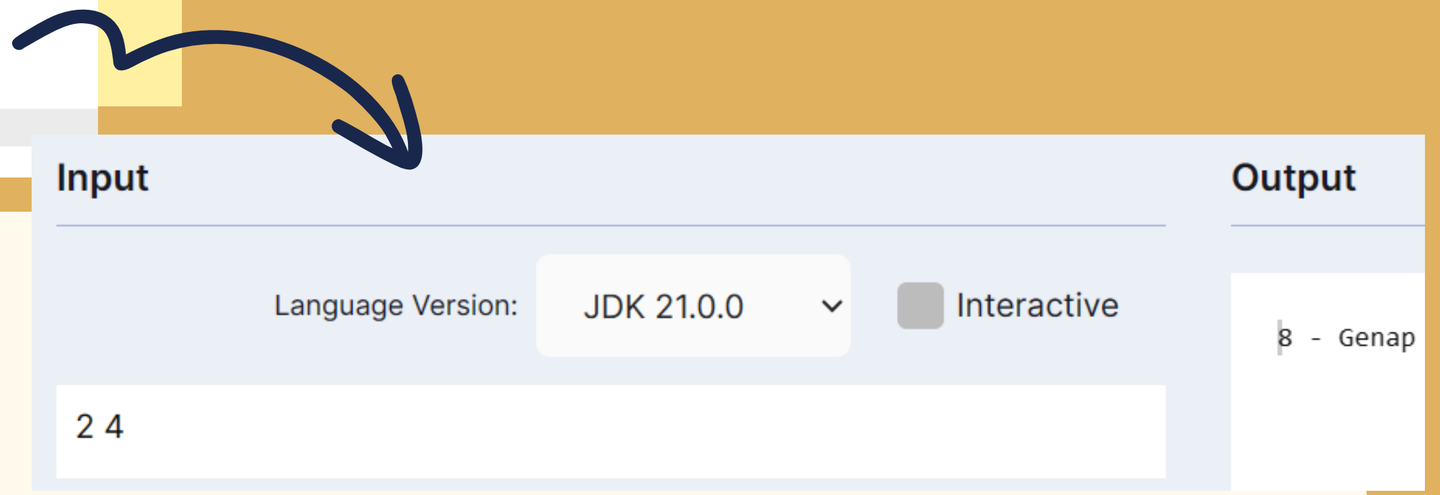
```
0
1
2
3
4


10
20
30
40
50


1
2
3
4
5


0
1
2
3
4
```

ARGUMENTS

```java
public class StudyCase {
    public static void main(String[] args) {
        //string by default
        System.out.println(args[0]);

        //parsing dari string to integer
        // int argInt = Integer.parseInt(args[1]);
        // System.out.println(argInt);

        int hasil=1;
        for (String arg:args){
            int argInt = Integer.parseInt(arg);
            hasil *= argInt;
        }

        System.out.print(hasil + " - ");

        if (hasil % 2 == 0) {
            System.out.print("Genap");
        } else {
            System.out.print("Ganjil");
        }
    }
}
```

Masukkan argumentsnya kedalam inputan untuk menghasilkan output sesuai print out yang diinginkan.

Dalam kasus berikut argument tersebut dikalikan.



DigitalSkola
Uncover The World of Digital Skills with Us

PROJECT 5

TUJUAN PROJEK

Project 5 - Create CRUD API Using Flask and Docker

Tools yang harus disiapkan:

- docker
- postman
- visual code

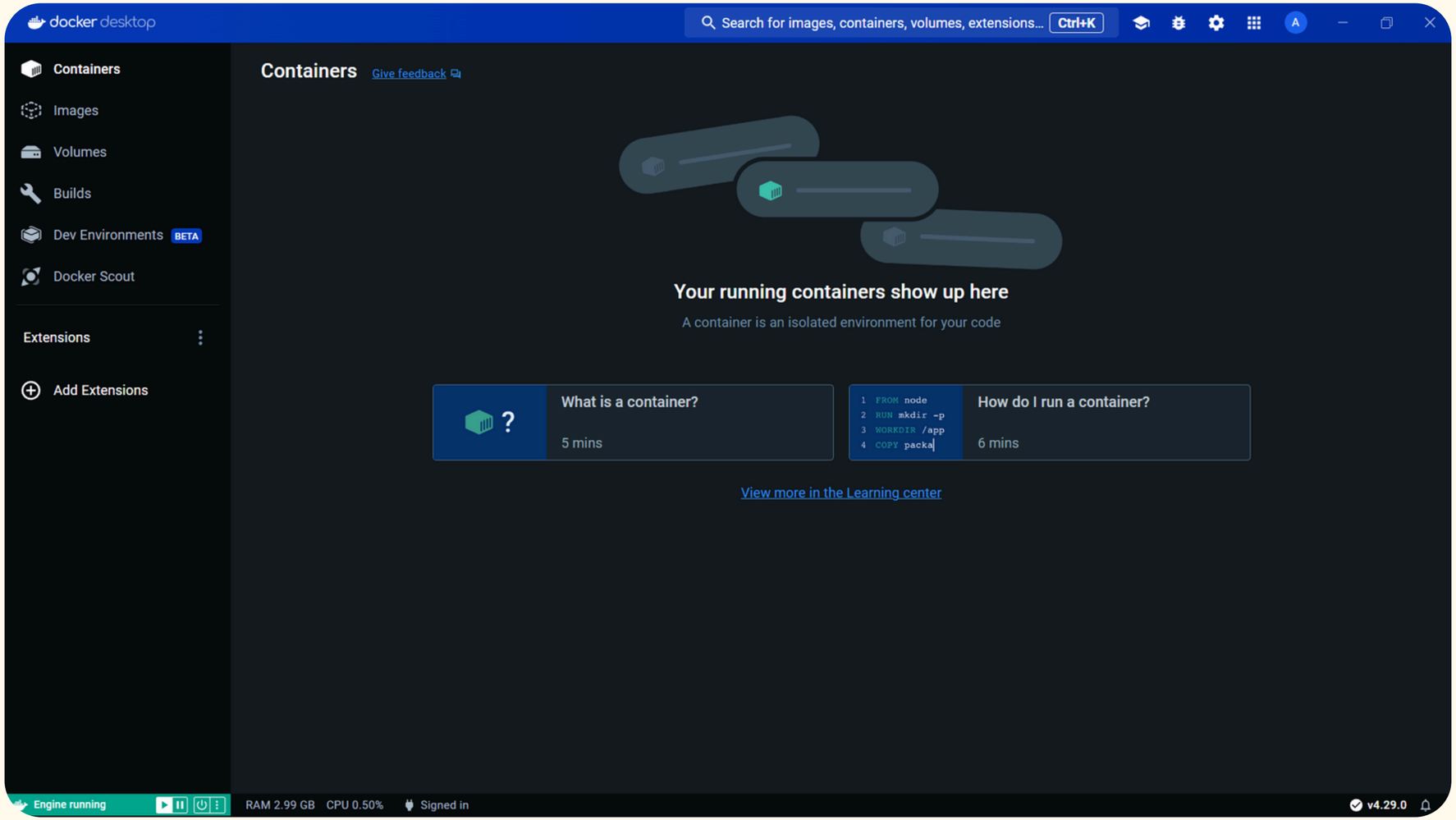
API yang dibuat:

- create user
- update user
- get user
- delete user

Projek ini bertujuan untuk membuat sebuah API CRUD menggunakan Flask dan Docker. Alat yang dibutuhkan adalah Docker untuk manajemen kontainer, Postman untuk pengujian API, dan Visual Studio Code sebagai code editor. API yang akan dibuat mencakup operasi dasar CRUD untuk entitas user: create, update, get, dan delete.

DigitalSkola

STEP 1: Menyalakan Docker Menggunakan Docker Desktop



DigitalSkola

STEP 2: Membuat file docker-compose.yml

File docker-compose.yml ini digunakan untuk mendefinisikan layanan yang akan dijalankan dalam lingkungan Docker. Tujuannya untuk menyiapkan sebuah layanan PostgreSQL, database yang akan digunakan sebagai basis data untuk aplikasi Flask yang akan dibuat.

docker-compose.yml

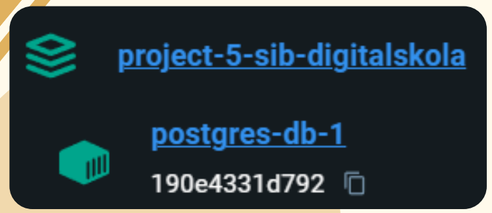
```yaml
version: "2"
services:
  postgres-db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: admin
      POSTGRES_USER: postgres
      PGDATA: /var/lib/postgres/data
      POSTGRES_DB: sample
    volumes:
      - pg-data:/var/lib/postgres/data
    ports:
      - 5432:5432/tcp
volumes:
  pg-data: #docker volume create pg-data
    external: true
```

STEP 3: Menjalankan Kontainer

Command '**docker volume create pg-data**' dijalankan untuk membuat volume Docker dengan nama **pg-data**. Tujuannya untuk menyiapkan tempat penyimpanan data yang akan digunakan oleh kontainer Docker (Postgres).

```
andrew@MSI:/mnt/d/Dokumen/TUGAS-TUGAS SIB DigitalSkola/TUGAS PROJEK (Individu + Kelompok)/PROJEK 5/PROJECT-5-SIB-DigitalSkola$ docker volume create pg-data
pg-data
andrew@MSI:/mnt/d/Dokumen/TUGAS-TUGAS SIB DigitalSkola/TUGAS PROJEK (Individu + Kelompok)/PROJEK 5/PROJECT-5-SIB-DigitalSkola$ docker-compose up -d
[+] Running 15/15
 ✓ postgres-db Pulled                                                                                                    296.3s
   ✓ b0a0cf830b12 Pull complete                                                                                           90.1s
   ✓ dda3d8fbd5ed Pull complete                                                                                            1.8s
   ✓ 283a477db7bb Pull complete                                                                                           11.9s
   ✓ 91d2729fa4d5 Pull complete                                                                                           13.0s
   ✓ 9739ced65621 Pull complete                                                                                           36.9s
   ✓ ae3bb1b347a4 Pull complete                                                                                           22.7s
   ✓ f8406d9c00ea Pull complete                                                                                           24.6s
   ✓ c199bff16b05 Pull complete                                                                                           27.6s
   ✓ e0d55fdb4d15 Pull complete                                                                                          270.8s
   ✓ c1cb13b19080 Pull complete                                                                                           38.8s
   ✓ 873532e5f8c7 Pull complete                                                                                          284.9s
   ✓ 050d9f8c3b1c Pull complete                                                                                           43.2s
   ✓ 710e142705f8 Pull complete                                                                                           45.0s
   ✓ cb628c265f09 Pull complete                                                                                           46.7s
[+] Running 2/2
 ✓ Network project-5-sib-digitalskola_default          Created                                                            0.2s
 ✓ Container project-5-sib-digitalskola-postgres-db-1  Started                                                            3.0s
```

Menjalankan kontainer Docker berdasarkan konfigurasi yang didefinisikan dalam file "docker-compose.yml" dengan command **'docker-compose up -d'**



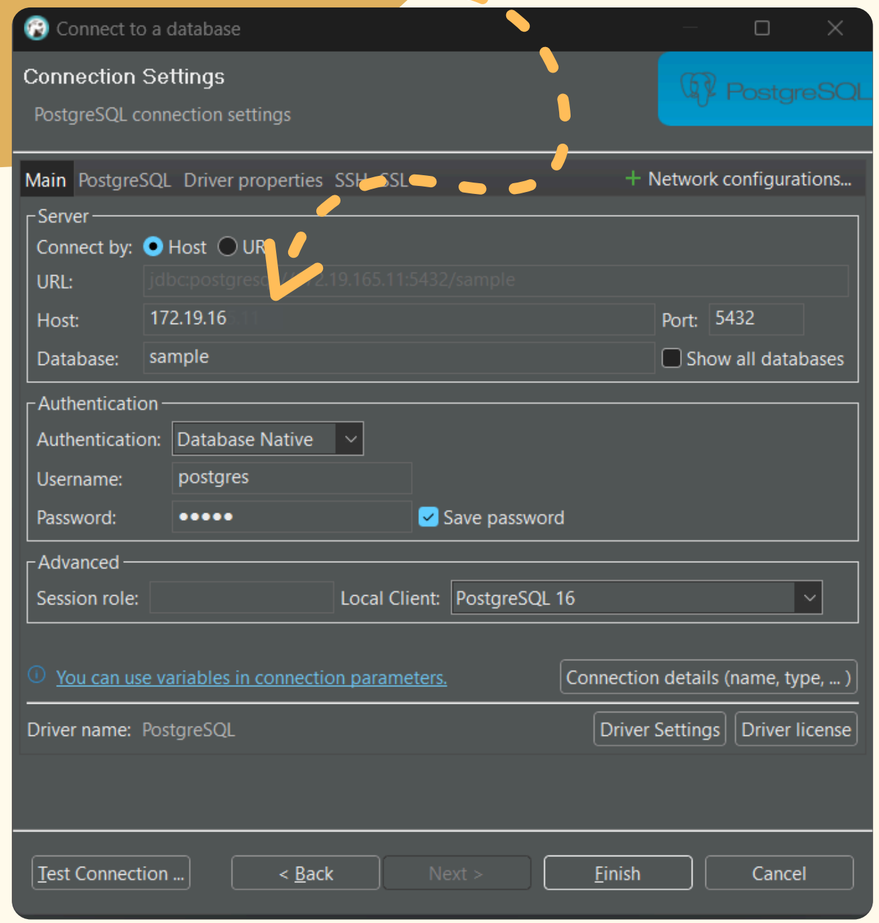
Melihat status pada docker desktop: tandanya sudah aktif

```
andrew@MSI:/mnt/d/Dokumen/TUGAS-TUGAS SIB DigitalSkola/TUGAS PROJEK (Individu + Kelompok)/PROJEK 5/PROJECT-5-SIB-DigitalSkola$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS                    NAMES
190e4331d792   postgres   "docker-entrypoint.s…"   28 seconds ago   Up 24 seconds   0.0.0.0:5432->5432/tcp   project-5-sib-digitalskola-postgres-db-1
```

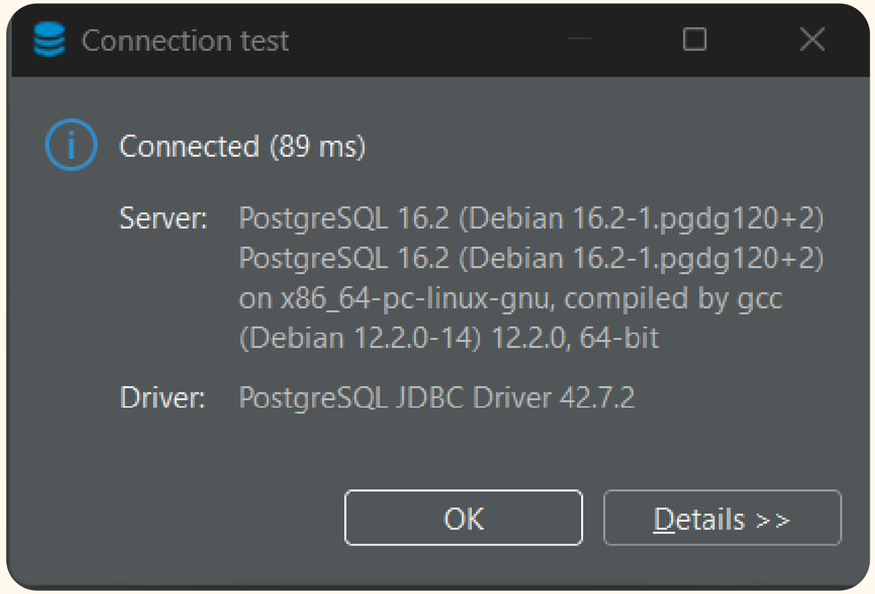
Cek status dengan command **'docker ps'**

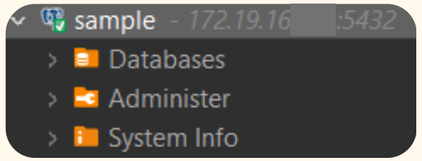
STEP 4: Test Connection Postgres

Host menggunakan IP network



Tes koneksi ke database Postgres menggunakan DBeaver





Connection tes “connected” artinya sudah berhasil terkoneksi ke Postgres dengan konfigurasi dari file docker-compose.yml

DigitalSkola
Uncover The World of Digital Skills with Us

STEP 5: Membuat Endpoint untuk Memeriksa Kesehatan Aplikasi dan Koneksi Database pada Flask

requirements.txt

```
Flask
psycopg2
python-decouple
flask-sqlalchemy
```

File requirements.txt berisikan requirements yang diperlukan.

app > main.py > ...

```python
from flask import Flask, jsonify
from decouple import config
import psycopg2


app = Flask(__name__)


@app.route("/health")
def health():
    return jsonify({"status": "oke"})


@app.route("/db_check")
def db_check():
    conn_pg = psycopg2.connect(
        host=config('MB_DB_HOST'),
        database=config('MB_DB_DBNAME'),
        user=config('MB_DB_USER'),
        password=config('MB_DB_PASS'),
        port=int(config('MB_DB_PORT')),
    )
    cur = conn_pg.cursor()
    return jsonify({"status": 200, "db": "connected"})


if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0")
```

File main.py menyediakan endpoints yang dapat digunakan untuk memeriksa kesehatan aplikasi secara umum (/health) dan untuk memeriksa koneksi ke database (/db_check)

STEP 6: Konfigurasi Dockerfile untuk Membangun Image Docker Aplikasi Flask

Dockerfile > ...

```
1   FROM python:3.7
2
3   COPY requirements.txt /requirements.txt
4
5   COPY ./app /app
6
7   WORKDIR /
8
9   RUN pip install -r requirements.txt
10
11  ENTRYPOINT ["python"]
12
13  CMD ["app/main.py", "--reload"]
```

Dockerfile bertujuan untuk membangun image Docker yang berisi aplikasi Flask beserta dependensinya, dan mengatur agar aplikasi Flask dijalankan saat container Docker berjalan.

DigitalSkola
Uncover The World of Digital Skills with Us

STEP 7: Build Image Docker

```
andrew@MSI:/mnt/d/Dokumen/TUGAS-TUGAS SIB DigitalSkola/TUGAS PROJEK (Individu + Kelompok)/PROJEK 5/PROJECT-5-SIB-DigitalSkola$ docker build -t project-5-sib-digitalskola .
[+] Building 30.8s (9/9) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 228B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/python:3.7                                                                 1.3s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [1/5] FROM docker.io/library/python:3.7@sha256:eedf63967cdb57d8214db38ce21f105003ed4e4d0358f02bedc057341bcf92a0           0.0s
 => [internal] load build context                                                                                             0.0s
 => => transferring context: 91B                                                                                              0.0s
 => CACHED [2/5] COPY requirements.txt /requirements.txt                                                                      0.0s
 => CACHED [3/5] COPY ./app /app                                                                                              0.0s
 => [4/5] RUN pip install -r requirements.txt                                                                                28.5s
 => exporting to image                                                                                                        0.7s
 => => exporting layers                                                                                                       0.6s
 => => writing image sha256:af9dc73b7c39fb6f873fe6fb983fdc1eddbb95a3f3792429fdd9b388d42b68da                                  0.0s
 => => naming to docker.io/library/project-5-sib-digitalskola                                                                 0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
```

Command **'docker build -t project-5-sib-digitalskola'** digunakan untuk membangun image Docker dari Dockerfile yang ada di dalam direktori.

```
andrew@MSI:/mnt/d/Dokumen/TUGAS-TUGAS SIB DigitalSkola/TUGAS PROJEK (Individu + Kelompok)/PROJEK 5/PROJECT-5-SIB-DigitalSkola$ docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
project-5-sib-digitalskola   latest    af9dc73b7c39   52 minutes ago   1.03GB
postgres                  latest    8e4fc9e18489   2 months ago     431MB
```

Mengecek docker image yang sudah dibuat dengan command **'docker images'**

STEP 8: Menambahkan layanan flask-app pada file docker-compose.yml

docker-compose.yml

```yaml
services:
  postgres-db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: admin
      POSTGRES_USER: postgres
      PGDATA: /var/lib/postgres/data
      POSTGRES_DB: sample
    volumes:
      - pg-data:/var/lib/postgres/data
    ports:
      - "5432:5432/tcp"
  flask-app:
    image: project-5-sib-digitalskola
    environment:
      MB_DB_DBNAME: postgres
      MB_DB_HOST: postgres-db
      MB_DB_PASS: admin
      MB_DB_PORT: 5432
      MB_DB_TYPE: postgres
      MB_DB_USER: postgres
    volumes:
      - ./app:/app
    links:
      - postgres-db:postgres-db
    ports:
      - "5000:5000/tcp"
volumes:
  pg-data: #docker volume create pg-data
    external: true
```

Mematikan docker compose **'docker-compose down'** kemudian menjalankan kembali dengan command '**docker-compose up -d**'

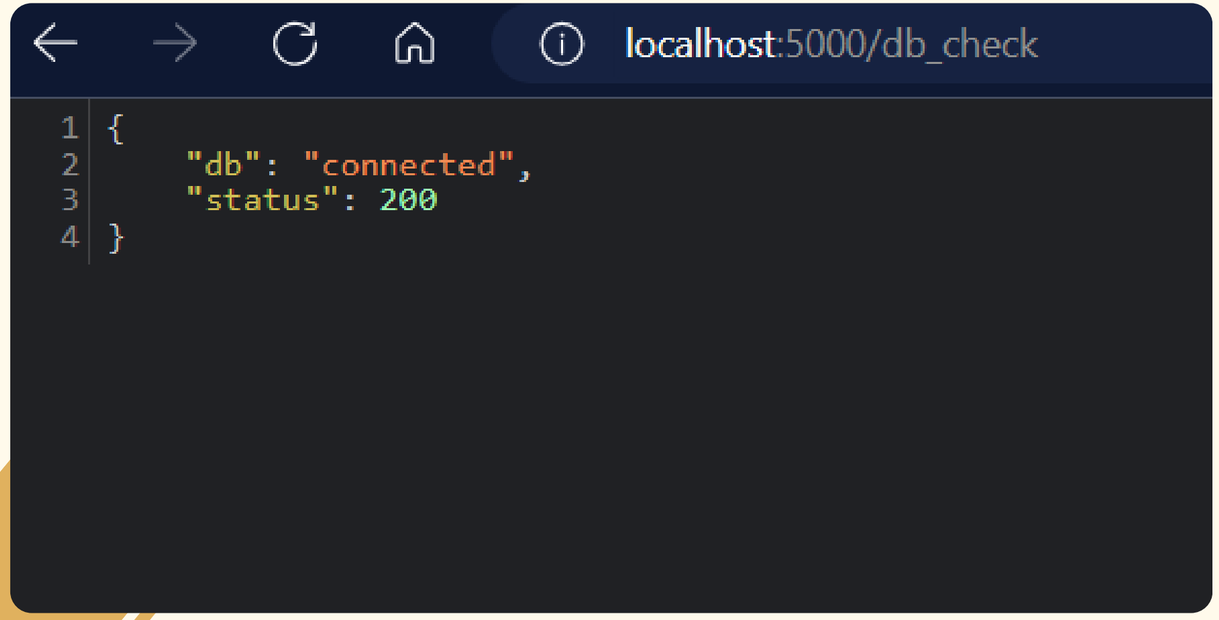
```
andrew@MSI:/mnt/d/Dokumen/TUGAS-TUGAS SIB DigitalSkola/TUGAS PROJEK (Individu + Kelompok)/PROJEK 5/PROJECT-5-SIB-DigitalSkola$ docker-compose down
[+] Running 3/3
 ✓ Container project-5-sib-digitalskola-flask-app-1    Removed                                                    0.5s
 ✓ Container project-5-sib-digitalskola-postgres-db-1  Removed                                                    0.6s
 ✓ Network project-5-sib-digitalskola_default          Removed                                                    0.3s
andrew@MSI:/mnt/d/Dokumen/TUGAS-TUGAS SIB DigitalSkola/TUGAS PROJEK (Individu + Kelompok)/PROJEK 5/PROJECT-5-SIB-DigitalSkola$ docker-compose up -d
[+] Running 3/3
 ✓ Network project-5-sib-digitalskola_default          Created                                                    0.1s
 ✓ Container project-5-sib-digitalskola-postgres-db-1  Started                                                    0.2s
 ✓ Container project-5-sib-digitalskola-flask-app-1    Started                                                    0.1s
```

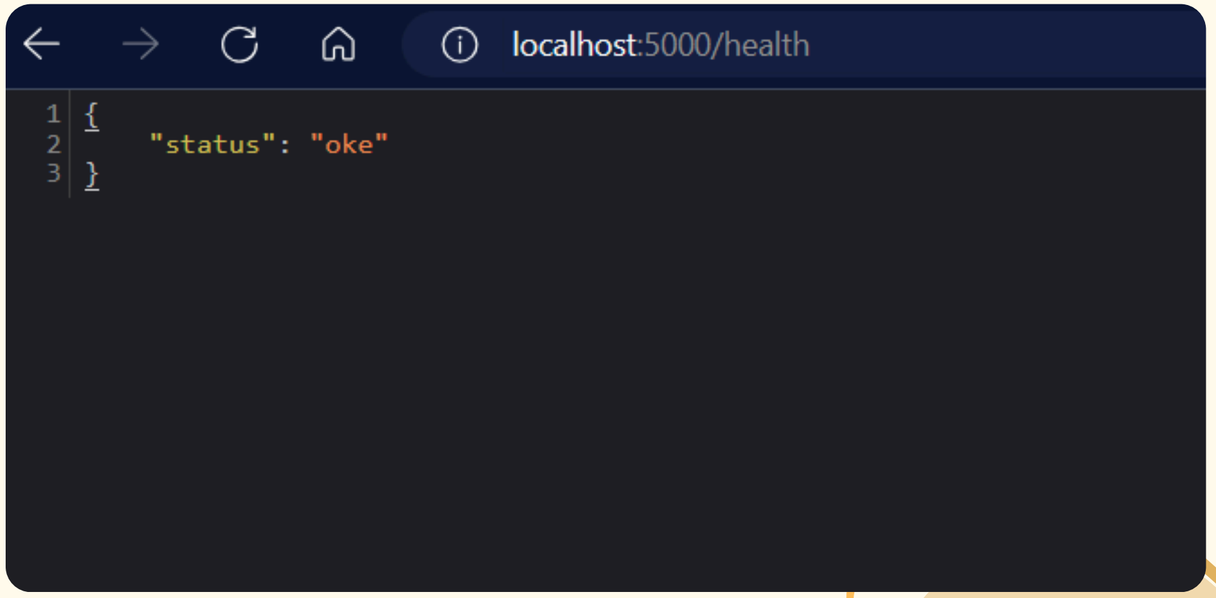
Cek status dengan command 'docker ps'

```
andrew@MSI:/mnt/d/Dokumen/TUGAS-TUGAS SIB DigitalSkola/TUGAS PROJEK (Individu + Kelompok)/PROJEK 5/PROJECT-5-SIB-DigitalSkola$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS                    NAMES
5e59edf01ebf   project-5-sib-digitalskola     "python app/main.py …"   12 seconds ago   Up 11 seconds   0.0.0.0:5000->5000/tcp   project-5-sib-digitalskola-flask-app-1
afb8131df8b5   postgres                       "docker-entrypoint.s…"   12 seconds ago   Up 11 seconds   0.0.0.0:5432->5432/tcp   project-5-sib-digitalskola-postgres-db-1
```

DigitalSkola

STEP 9: Memeriksa Endpoint Koneksi /db_check dan /health pada localhost 5000





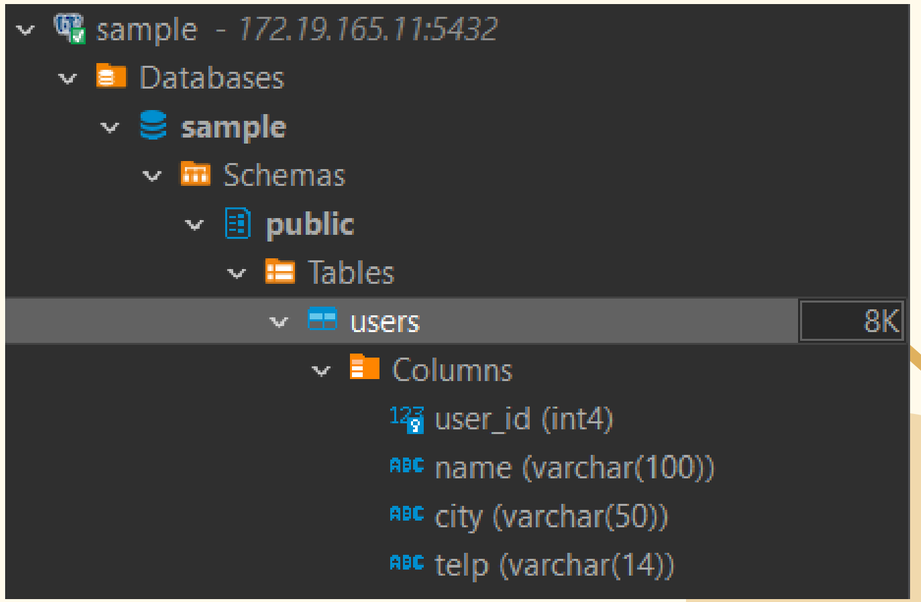
STEP 10: Membuat DDL Pada Database Postgres di DBeaver

*<sample> Script-5

```
create sequence user_id_seq
    start with 1
    increment by 1
    no minvalue
    no maxvalue
    cache 1;
CREATE TABLE public.users (
    user_id int4 NOT NULL DEFAULT nextval('user_id_seq'::regclass),
    name varchar(100) NULL,
    city varchar(50) NULL,
    telp varchar(14) NULL,
    CONSTRAINT users_pkey PRIMARY KEY (user_id)
);
```

Membuat struktur database untuk menyimpan informasi pengguna dalam proyek Flask

Terdapat tabel baru dengan nama “users” yang berisikan kolom user_id, name, city, dan telp.



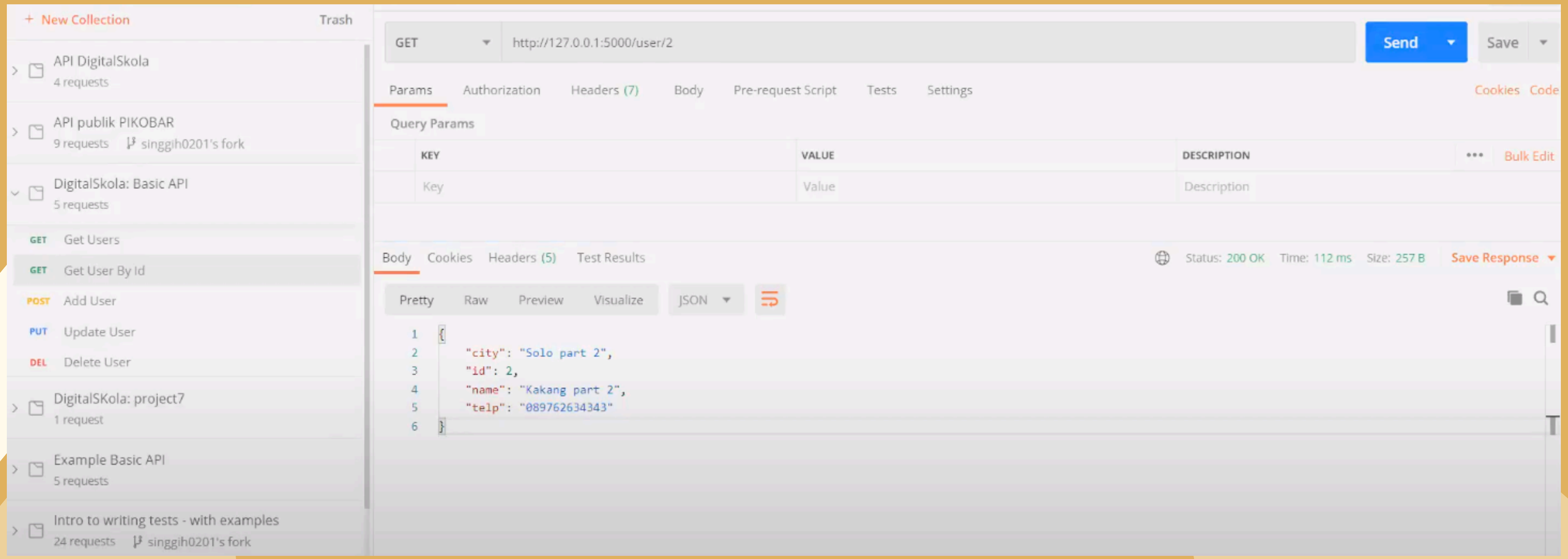
STEP 11: Penambahan Endpoint CRUD dalam Aplikasi Flask

app > main.py > ...

```python
from flask import Flask, jsonify, request
from decouple import config
import psycopg2
from flask_sqlalchemy import SQLAlchemy


DB_URI = f"postgresql+psycopg2://{config('MB_DB_USER')}:{config('MB_DB_PASS')}@{config('MB_DB_HOST')}:{str(config('MB_DB_PORT'))}/{config('MB_DB_DBNAME')}"
app = Flask(__name__)
app.config["SQLALCHEMY_DATABASE_URI"] = DB_URI
app.config["SQLALCHEMY_TRACK_MODIFIACTIONS"] = True
db = SQLAlchemy(app)


class Users(db.Model):
    id = db.Column('user_id', db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    city = db.Column(db.String(50))
    telp = db.Column(db.String(14))


@app.route("/health")
def health():
    return jsonify({"status": "oke"})


@app.route("/db_check")
def db_check():
    conn_pg = psycopg2.connect(
        host=config('MB_DB_HOST'),
        database=config('MB_DB_DBNAME'),
        user=config('MB_DB_USER'),
        password=config('MB_DB_PASS'),
        port=int(config('MB_DB_PORT')),
    )
    cur = conn_pg.cursor()
    return jsonify({"status": 200, "db": "connected"})


@app.route("/user", methods=["GET", "POST", "PUT", "DELETE"])
def user():
    if request.method == 'GET':
        users = Users.query.all()
        results = [{"id": u.id, "name": u.name, "city": u.city, "telp": u.telp} for u in users]
        return jsonify(results)


if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0")
```

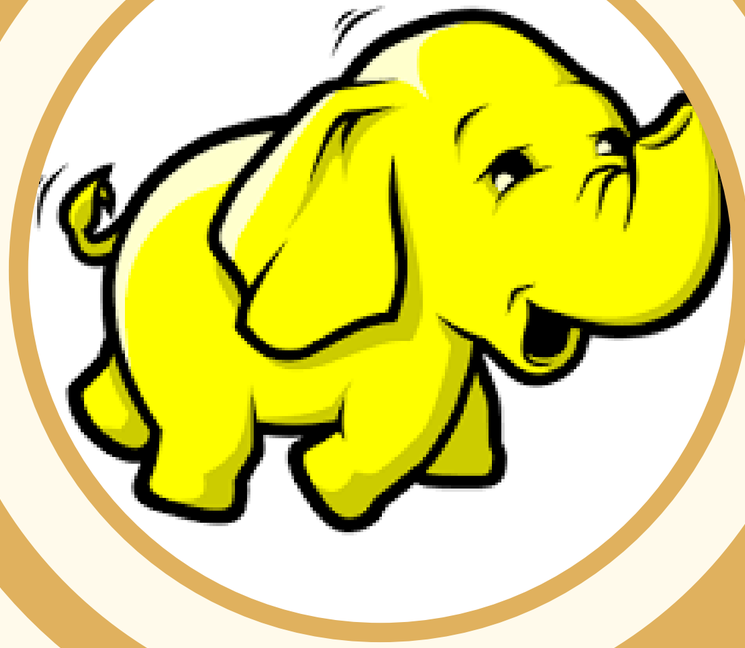
Menambahkan fitur-fitur CRUD yang memungkinkan aplikasi untuk melakukan operasi Create, Read, Update, dan Delete pada data pengguna dalam database PostgreSQL.

STEP 12: Menampilkan Hasil Permintaan API Flask pada Postman



DigitalSkola
Uncover The World of Digital Skills with Us

HADOOP



DigitalSkola

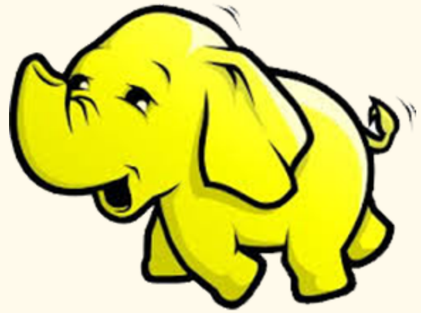
What is Hadoop?

- Apache top level project, open-source implementation of frameworks for reliable, scalable, distributed computing and data storage.
- It is a flexible and highly-available architecture for large scale computation and data processing on a network of commodity hardware
- Goals / Requirements:
  - Abstract and facilitate the storage and processing of large and/or rapidly growing data sets
  - Structured and non-structured data
  - Simple programming models
  - High scalability and availability
  - Use commodity (cheap!) hardware with little redundancy
  - Fault-tolerance and Move computation rather than data

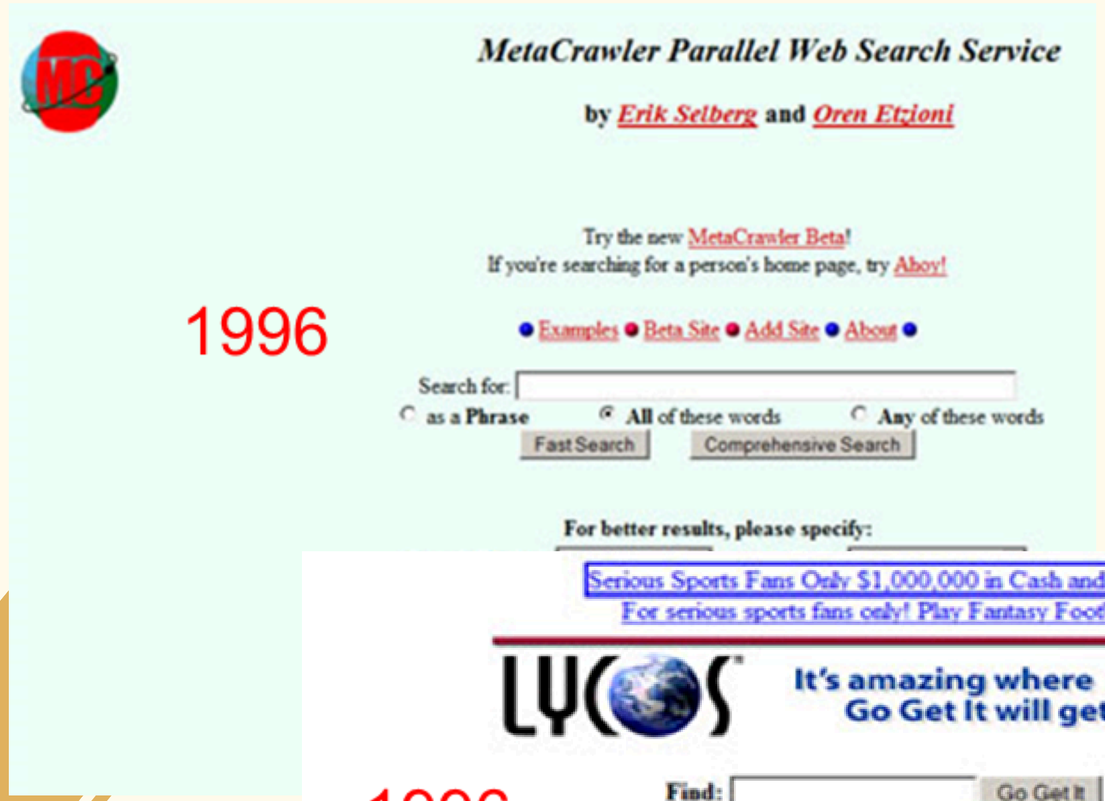
Brief History of Hadoop

Designed to answer the question:

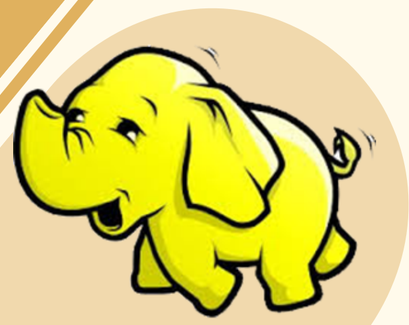
“How to process big data with reasonable cost and time?”



Search engines in 1990s



Google search engines



2013

Google

Google Search

I'm Feeling Lucky

Hadoop's Developers

Doug Cutting

2005: Doug Cutting and Michael J. Cafarella developed Hadoop to support distribution for the Nutch search engine project.

The project was funded by Yahoo.

2006: Yahoo gave the project to Apache Software Foundation.

Google Origins

2003

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google•

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

2004 jeff@google.com, sanjay@google.com

Google, Inc.

Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber

{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com

2006 *Google, Inc.*

Abstract

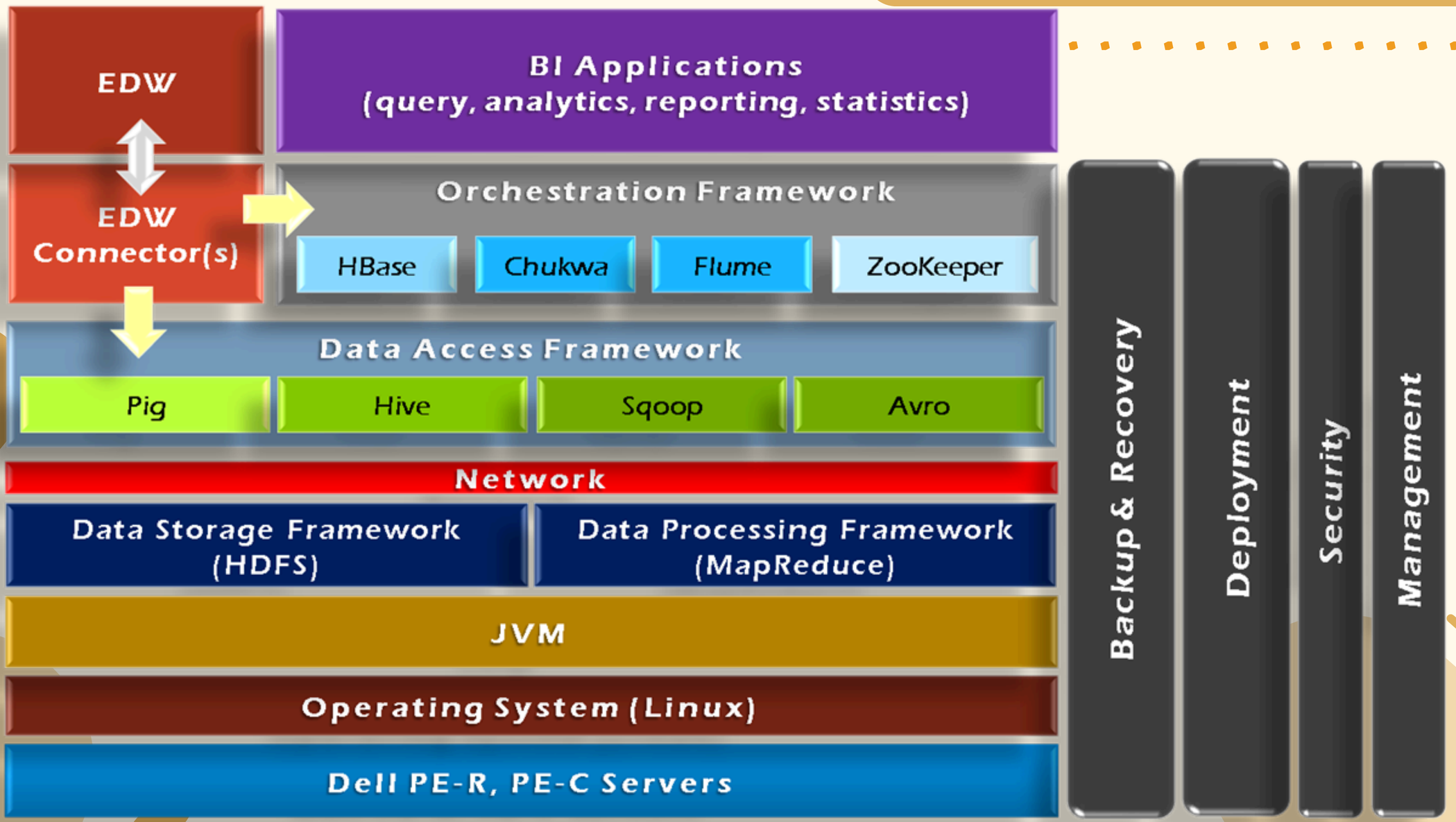
gtable is a distributed storage system for managing
tured data that is designed to scale to a very large
petabytes of data across thousands of commodity
rs. Many projects at Google store data in Bigtable,
dline web indexing, Google Earth, and Google Fi-

achieved scalability and high performance, but Big
provides a different interface than such systems. Big
does not support a full relational data model; inste
provides clients with a simple data model that sup
dynamic control over data layout and format, an
lows clients to reason about the locality properties o

Some Hadoop Milestones

- 2008 - Hadoop Wins Terabyte Sort Benchmark (sorted 1 terabyte of data in 209 seconds, compared to previous record of 297 seconds)
- 2009 - Avro and Chukwa became new members of Hadoop Framework family
- 2010 - Hadoop's Hbase, Hive and Pig subprojects completed, adding more computational power to Hadoop framework
- 2011 - ZooKeeper Completed
- 2013 - Hadoop 1.1.2 and Hadoop 2.0.3 alpha.
  - Ambari, Cassandra, Mahout have been added

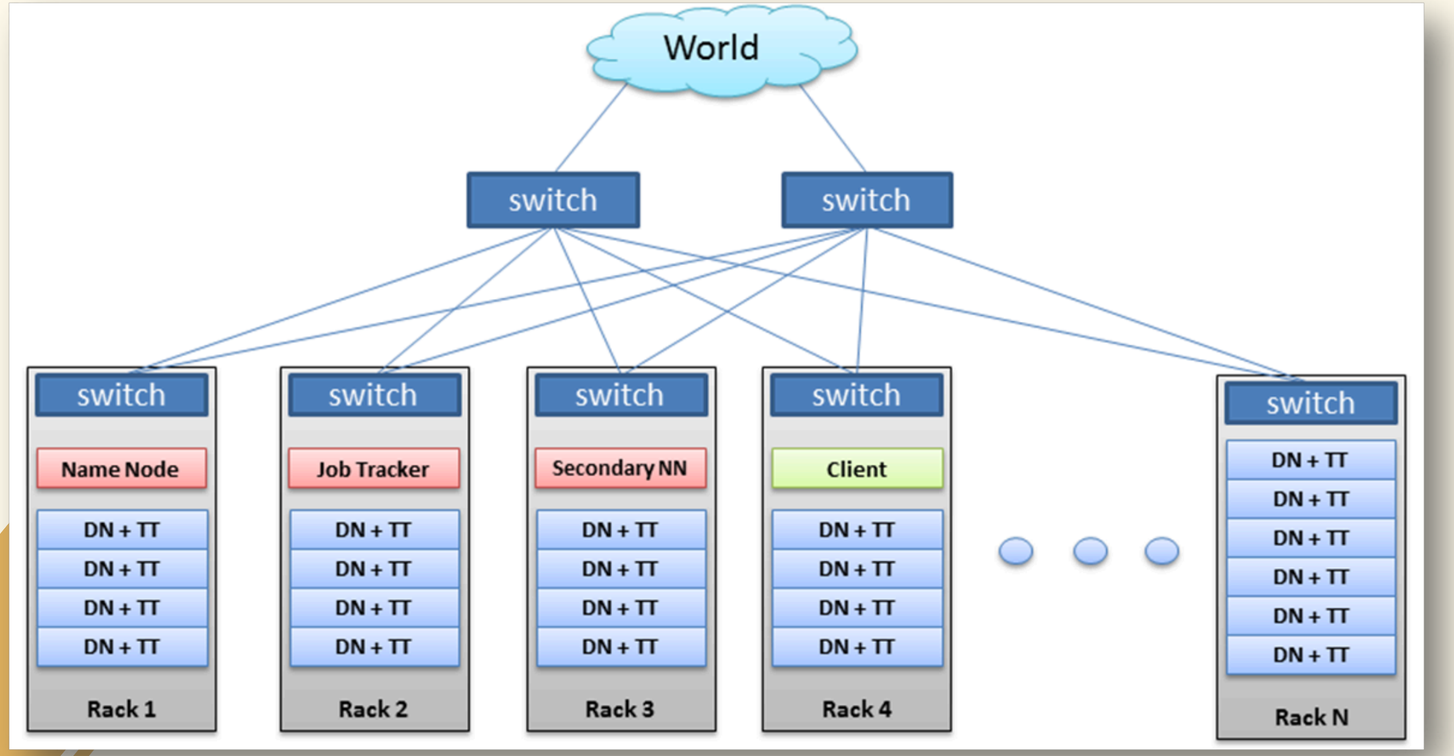
Hadoop Framework Tools



Hadoop's Architecture

- Distributed, with some centralization.
- Main nodes of cluster are where most of the computational power and storage of the system lies.
- Main nodes run TaskTracker to accept and reply to MapReduce tasks, and also DataNode to store needed blocks closely as possible.
- Central control node runs NameNode to keep track of HDFS directories & files, and JobTracker to dispatch compute tasks to TaskTracker.
- Written in Java, also supports Python and Ruby

Hadoop's Architecture

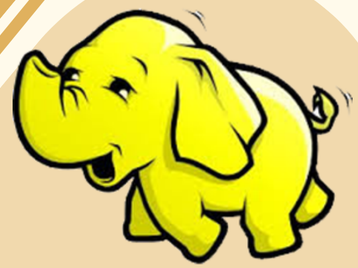


- HadoopDistributed Filesystem
- Tailored to needs of MapReduce
- Targeted towards many reads of filestreams
- Writes are more costly
- High degree of data replication (3x by default)
- No need for RAID on normal nodes
- Large blocksize(64MB)
- Location awareness of DataNodes in network

Hadoop's Architecture

DataNode:

- Stores the actual data in HDFS
- Can run on any underlying filesystem (ext3/4, NTFS, etc)
- Notifies NameNode of what blocks it has
- NameNode replicates blocks 2x in local rack, 1x elsewhere



NameNode:

- •Stores metadata for the files, like the directory structure of a typical FS.
- The server holding the NameNode instance is quite crucial, as there is only one.
- Transaction log for file deletes/adds, etc. Does not use transactions for whole blocks or file-streams, only metadata.
- Handles creation of more replica blocks when necessary after a DataNode failure

Hadoop in the Wild

Big player

Hadoop is in use at most organizations that handle big data: Yahoo!, Facebook, Amazon, Netflix, Etc...

Some examples of scale

Yahoo!'s Search Webmap runs on 10,000 core Linux cluster and powers Yahoo! Web search

Some examples of scale

FB's Hadoop cluster hosts 100+ PB of data (July, 2012) & growing at ½ PB/day (Nov, 2012)

Three main applications

- Advertisement (Mining user behavior to generate recommendations
- •Searches (group related documents)
- Security (search for uncommon patterns)

HDFS

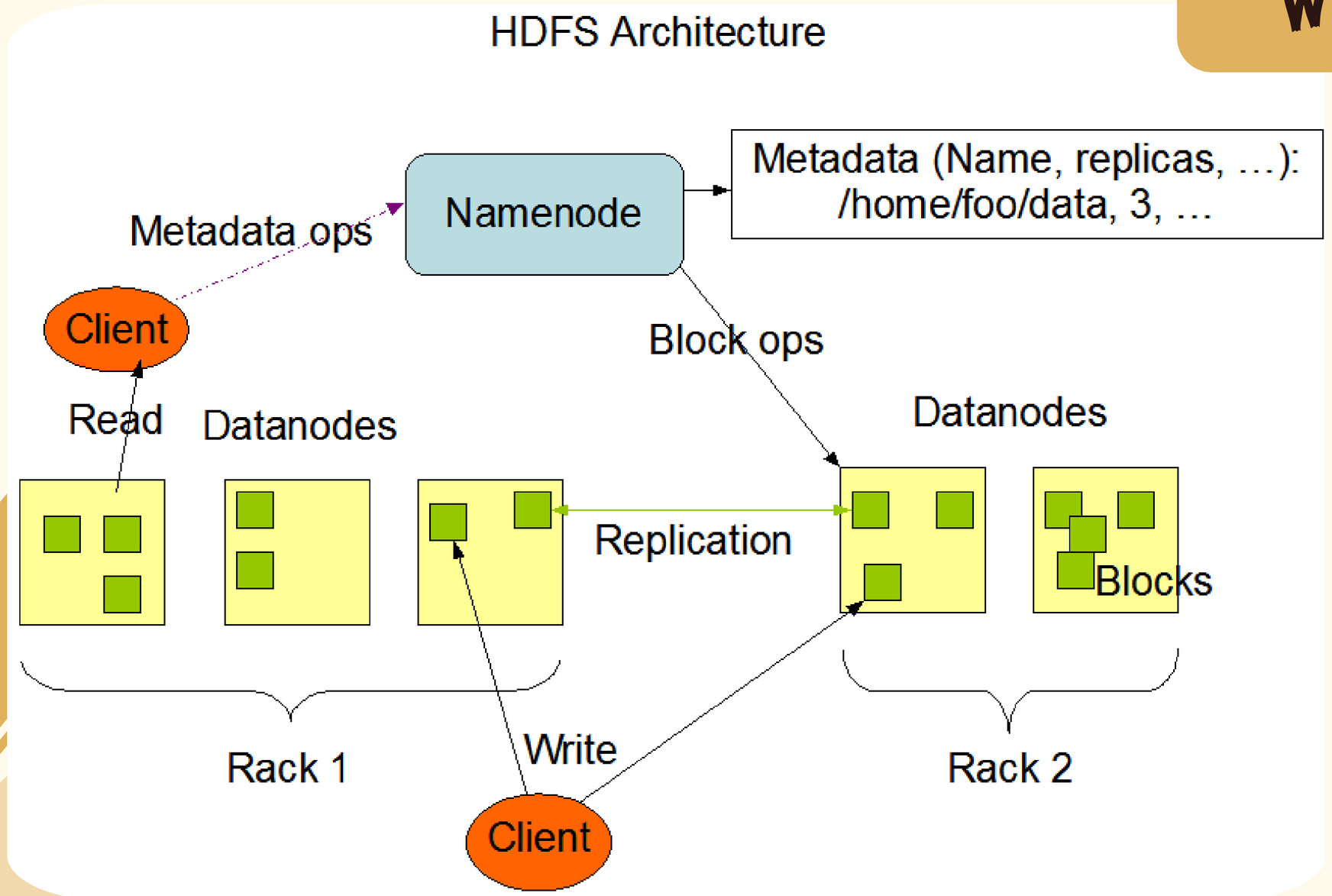
What is HDFS?

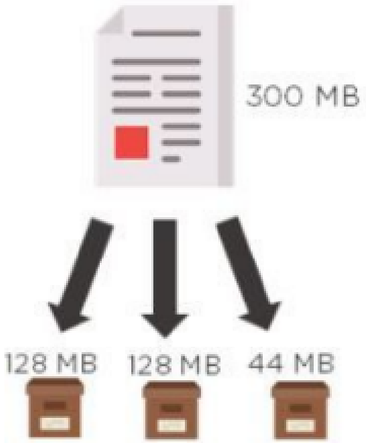
- HDFS (Hadoop Distributed File System) adalah filesystem yang didesain untuk menyimpan file dengan ukuran sangat besar (ratusan MB, GB, TB, bahkan sampai PB) dengan streaming data access pattern (write-once, read-many-times), berjalan di atas suatu cluster of commodity hardware.

- HDFS merupakan bagian dari Apache Hadoop, sebuah kerangka kerja open-source yang mampu menyimpan, memproses, dan menganalisis data.

- HDFS mengatur storage melalui sejumlah mesin melalui network

What is HDFS?


HDFS Architecture
Namenode
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client


300 MB
128 MB
128 MB
44 MB

HDFS Characteristic

Skalabilitas

HDFS dapat menangani set data besar dan dapat diskalakan hingga ribuan node dalam satu kluster Hadoop.

Toleransi Kesalahan Tinggi

HDFS sangat toleran terhadap kesalahan dan dirancang untuk dijalankan pada perangkat keras berbiaya rendah.

Akses Data Cepat

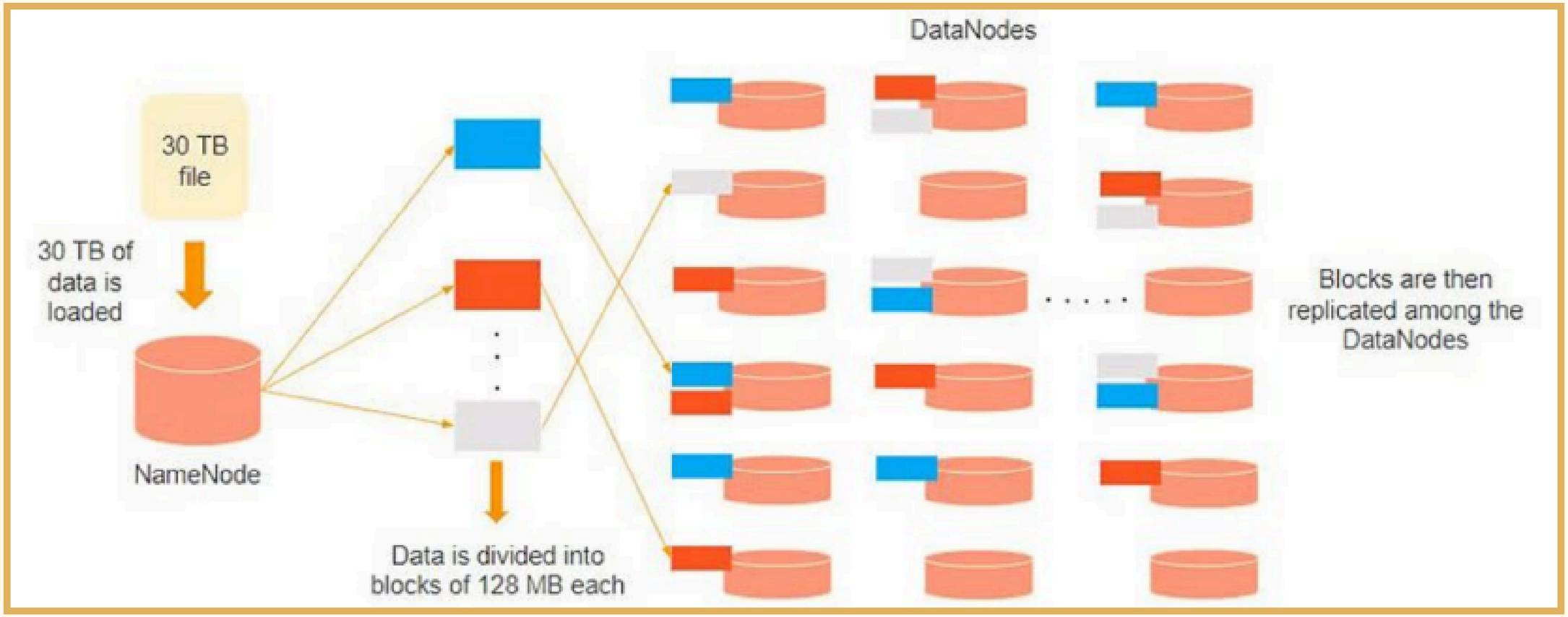
HDFS menyediakan akses berkecepatan tinggi ke data aplikasi dan cocok untuk aplikasi yang memiliki set data besar.

Penyimpanan Blok Data

HDFS menyimpan data dalam bentuk blok, di mana ukuran setiap blok data adalah 128MB.

NAMENODE & DATANODE

Sebuah HDFS cluster memiliki dua tipe nodes yang beroperasi dengan master-worker pattern. Namenode (master) dan sejumlah datanodes (workers).



NAMENODE & DATANODE

Namenode

- Bertindak sebagai master
- Menyimpan semua metadata dari semua file dalam HDFS.
- Metadata ini mencakup izin file, nama, dan lokasi setiap blok data.
- Memetakan blok-blok ke DataNodes.

Datanode

- Bertindak sebagai workers
- Bertugas untuk menyimpan dan mengambil blok data sesuai dengan instruksi dari NameNode.
- Melakukan pembuatan blok, penghapusan, dan replikasi seperti yang diinstruksikan oleh NameNode.

Namenode Improvement

Backup Metadata

- Hadoop dapat dikonfigurasi untuk membuat backup metadata ke banyak filesystem.
- Backup ini bersifat sinkron dan atomik, yang menjaga integritas data.
- Backup biasanya dibuat ke disk lokal dan remote NFS mount.

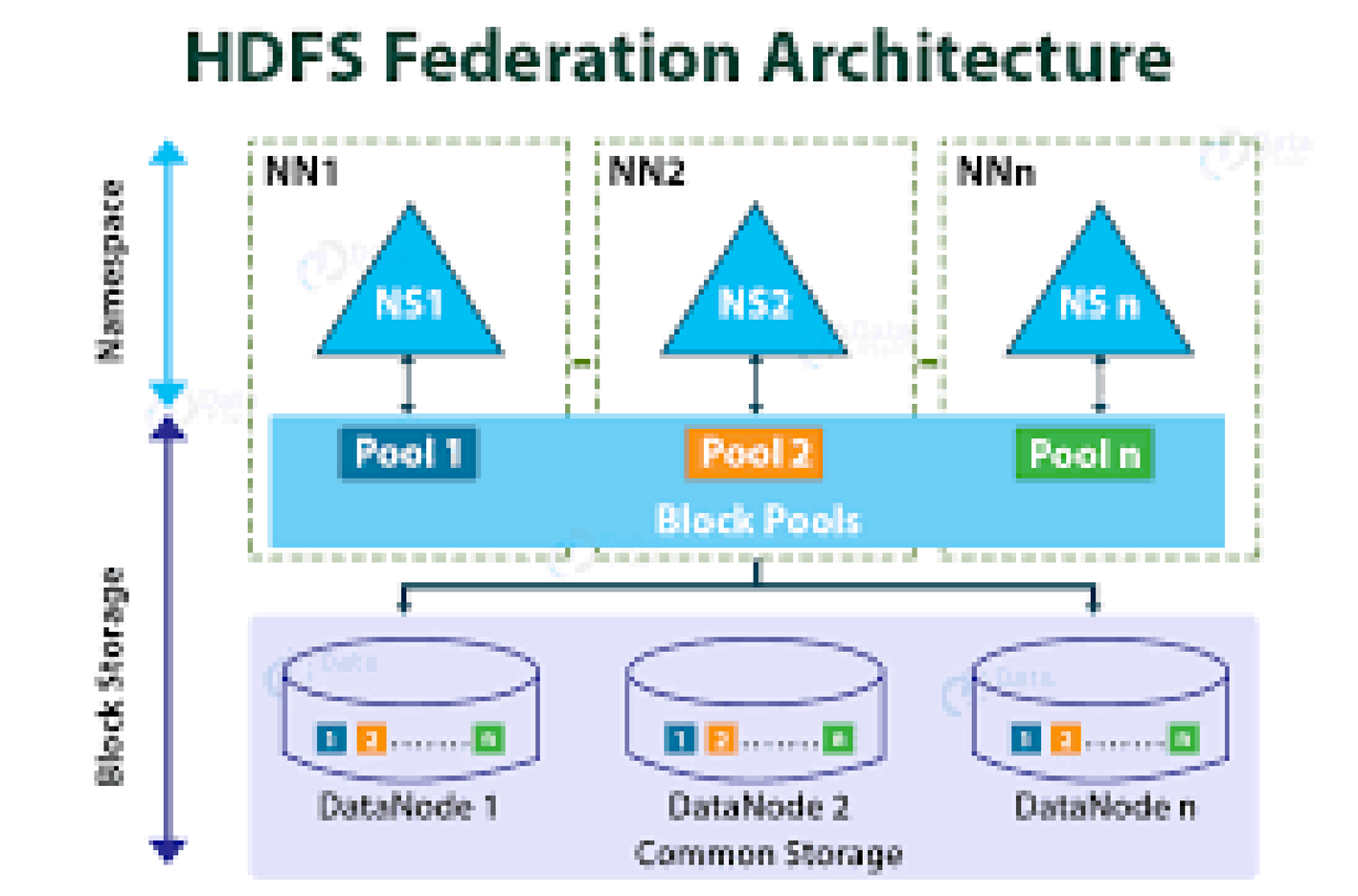
Secondary Namenode

- Secondary NameNode dijalankan untuk menggabungkan image namespace dengan edit log.
- Secondary NameNode biasanya berjalan pada mesin yang berbeda dan membutuhkan sumber daya yang sama dengan NameNode.
- Secondary NameNode menyimpan salinan dari image namespace yang digabungkan.
- Jika Primary NameNode gagal, Secondary NameNode dapat digunakan, tetapi mungkin ada sedikit data loss karena state Secondary NameNode biasanya tertinggal dari Primary.

Block Caching

- Secara umum, DataNode membaca blok langsung dari disk.
- Namun, untuk file yang sering diakses, blok dapat di-cache secara eksplisit di memori DataNode, dalam off-heap block cache.
- Secara default, sebuah blok hanya di-cache pada satu DataNode, meskipun jumlahnya dapat dikonfigurasi sesuai kebutuhan.
- Job schedulers (untuk MapReduce, Spark, dan framework lainnya) dapat memanfaatkan blok yang di-cache ini dengan menjalankan tugasnya pada DataNode tempat blok tersebut di-cache, untuk meningkatkan kinerja baca.

HDFS Federation



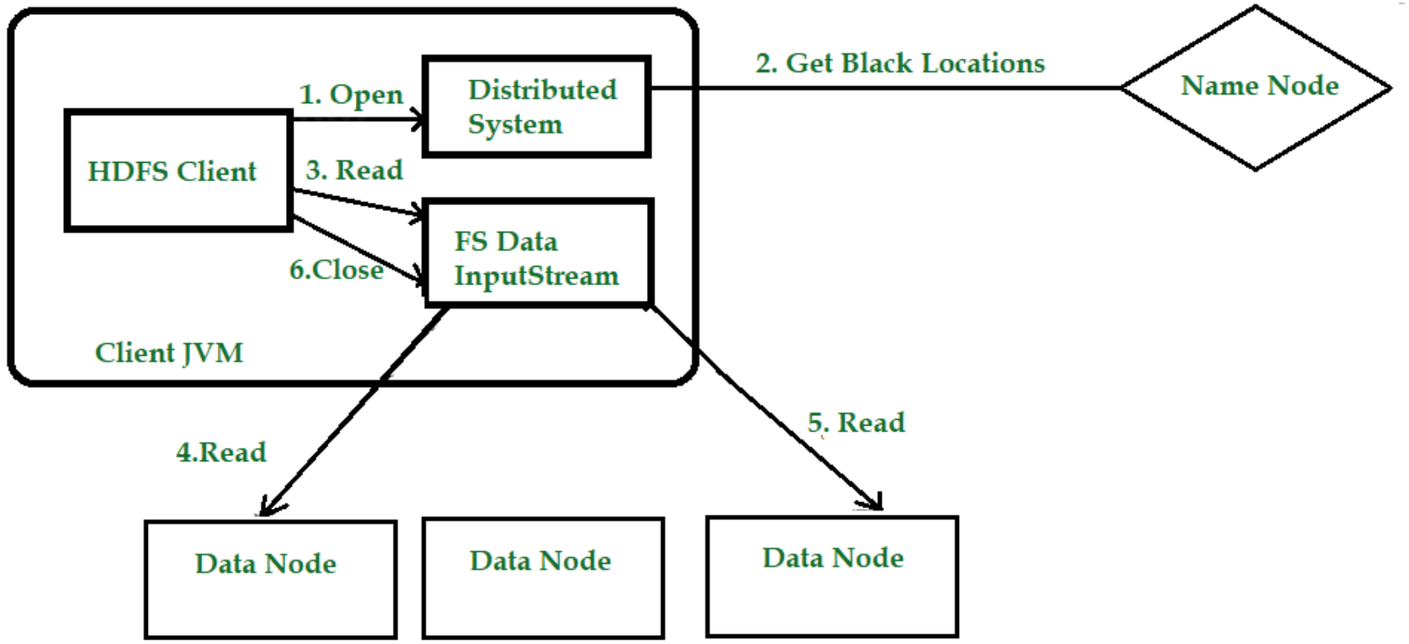
HDFS Federation memungkinkan penambahan NameNode untuk scaling. Setiap NameNode hanya mengelola sebagian namespace filesystem. Hal ini memungkinkan peningkatan skalabilitas dan reliabilitas dalam Hadoop

DigitalSkola

High Availability HDFS

- Hadoop mendukung High Availability dengan memiliki lebih dari satu NameNode yang bekerja dalam konfigurasi active-standby. Jika terjadi kegagalan pada NameNode aktif, maka NameNode standby akan mengambil alih.
- Beberapa perubahan yang perlu dilakukan untuk mencapai High Availability antara lain:
  - Namenode perlu menggunakan highly available shared storage untuk share edit log nya ke sesama namenode.
  - Datanodes perlu mengirim block reports ke semua namenode.
  - Clients harus dikonfigurasi supaya dapat menghandle namenode failover.
  - Role dari secondary namenode tetap dilakukan oleh standby namenode.

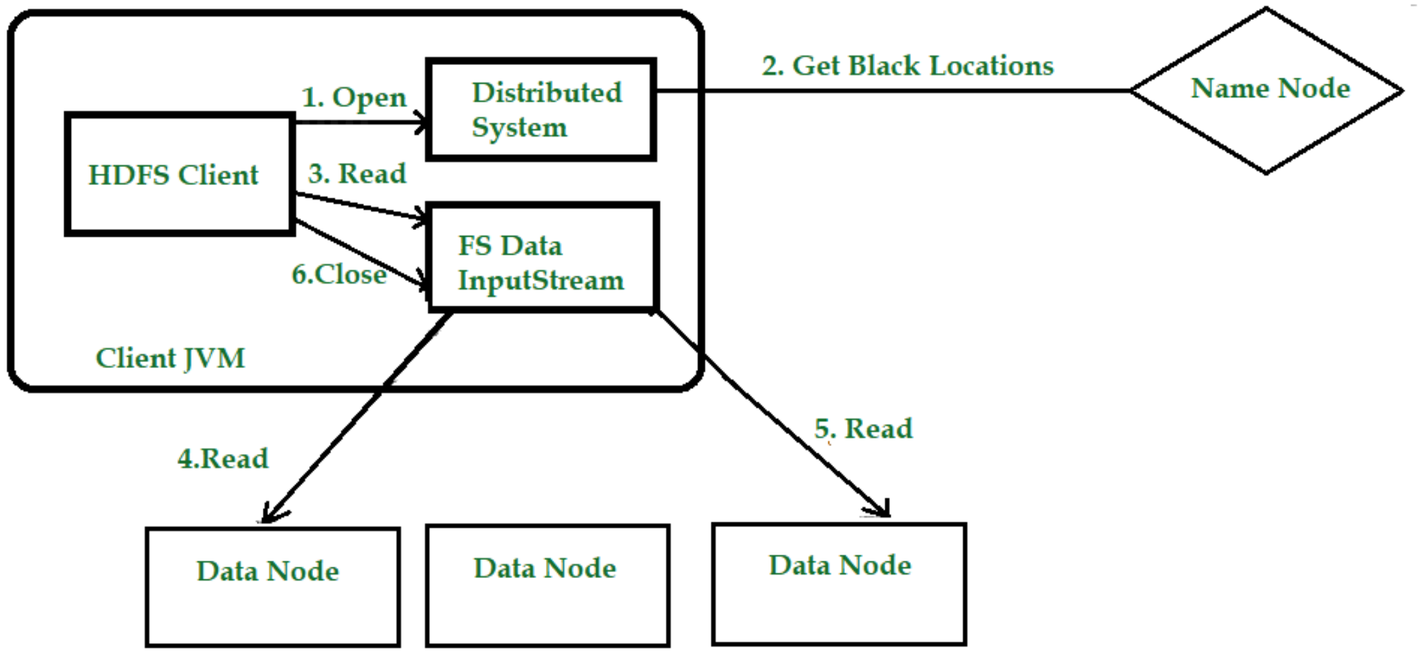
Anatomy File Read



- Klien membuka file yang ingin dibaca.
- DFS memanggil NameNode untuk menentukan lokasi beberapa blok pertama dalam file. Untuk setiap blok, NameNode mengembalikan alamat DataNode yang memiliki salinan blok tersebut.
- Klien kemudian memanggil read() pada stream. DFSInputStream, yang telah menyimpan alamat node data untuk beberapa blok pertama dalam file, kemudian terhubung ke DataNode pertama (terdekat) untuk blok pertama dalam file.

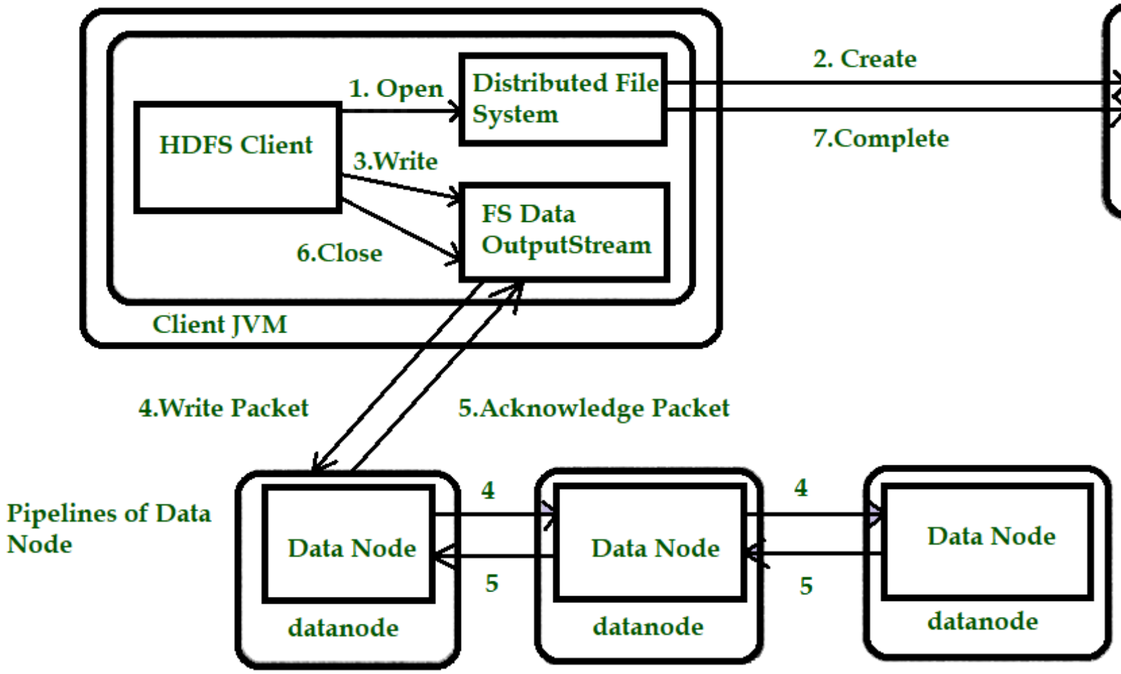
DigitalSkola
Uncover The World of Digital Skills with Us

Anatomy File Read



- Data diteruskan dari DataNode kembali ke klien, yang memanggil read() berulang kali pada stream.
- Ketika akhir blok tercapai, DFSInputStream akan menutup koneksi ke DataNode, kemudian menemukan DataNode terbaik untuk blok berikutnya. Ini terjadi secara transparan bagi klien, yang dari sudut pandangnya hanya membaca stream tanpa henti.
- Ketika klien telah selesai membaca file, fungsi close() dipanggil pada FSDataInputStream.

Anatomy File Write



- Klien membuat file baru.
- Klien menulis data ke file. Data dibagi menjadi paket.
- Paket dikirim ke sekelompok DataNode (pipeline).
- Setiap DataNode dalam pipeline menyimpan paket dan meneruskannya ke DataNode berikutnya.
- Klien menunggu pengakuan dari semua DataNode bahwa paket telah disimpan.
- Klien memberi tahu NameNode bahwa file sudah lengkap.

DigitalSkola

TERIMAKASIH