

# Automated Application Security Testing

Nikolai Limbrunner

# Agenda

1. Introduction
2. Static Application Security Testing
3. Software Composition Analysis
4. Dynamic Application Security Testing
5. Interactive/Real-Time Application Security Testing
6. How do they fit in our pipeline?
7. Which one to choose?
8. Take home

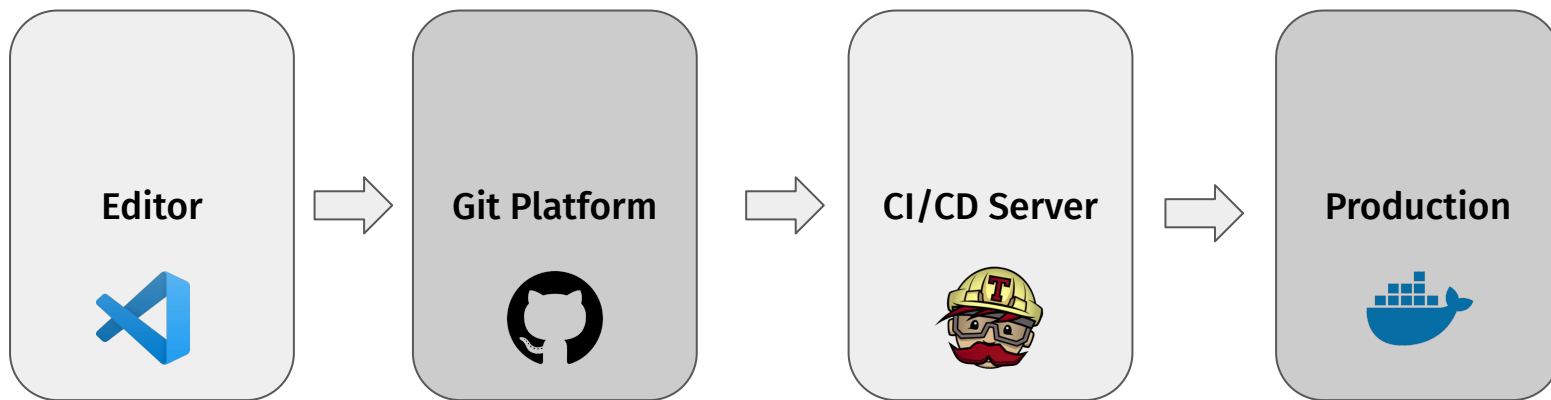
# 0. Mentimeter Quiz



7066 7401

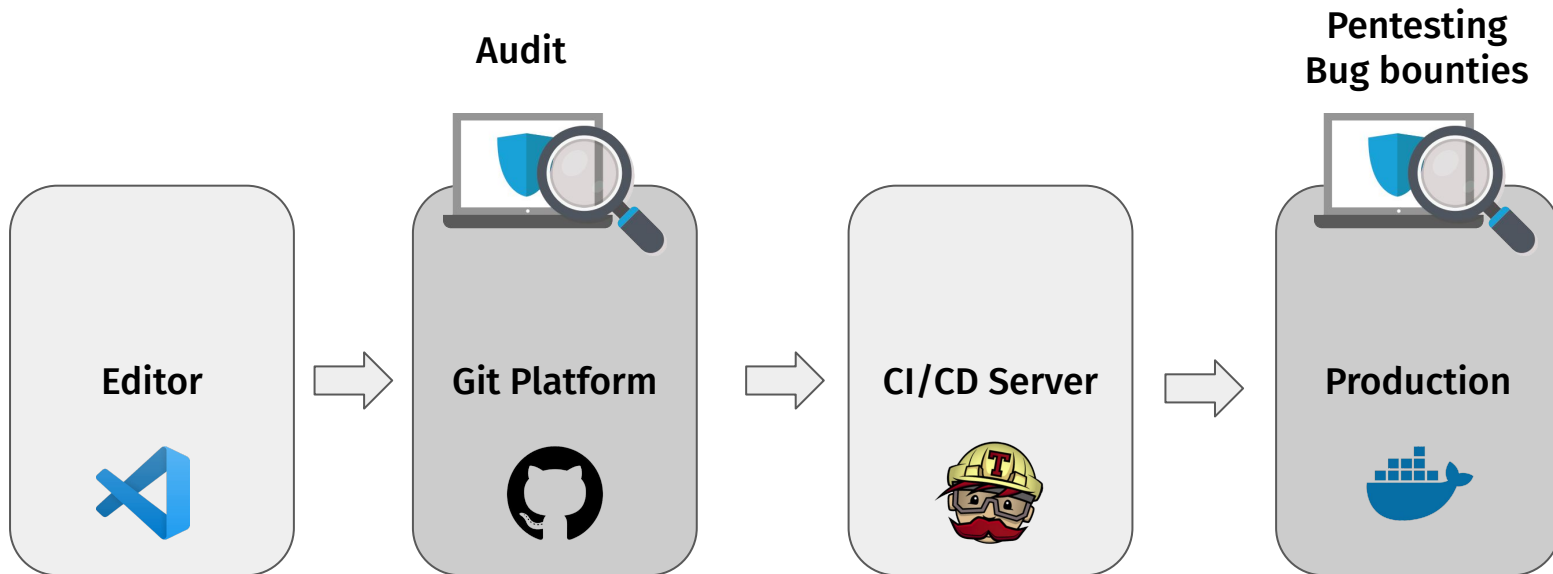
# 1. Introduction

Pipeline for a full stack application



# 1. Introduction

## Classical Security Approach



How can we integrate automated security testing in our pipeline?

## 2. Static Application Security Testing (SAST)

- Scan code for vulnerable code parts
- Use patterns and customized rules
- Language and framework dependent
- Scan can take hours

```
//establishing connection and use database with the name market in the specified
Connection conn = DriverManager.getConnection("jdbc:sqlite:D:\\Java Course\\SQLit


//scan user's input
Scanner scan = new Scanner(System.in);
System.out.println("Username : ");
String username = scan.nextLine();

System.out.println("Password : ");
String password = scan.nextLine();

//sql statement that vulnerable to sql inject
Statement statement = conn.createStatement();
statement.execute("SELECT * FROM MsUser WHERE username='" + username + "'" + " AND
System.out.println("SELECT * FROM MsUser WHERE username='" + username + "'" + " AND

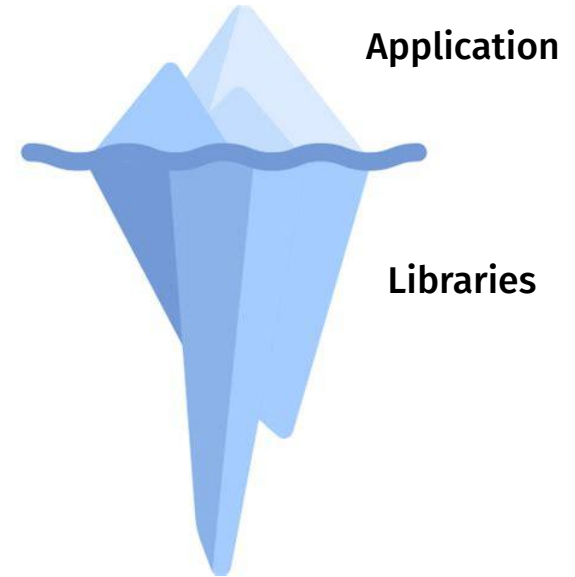
//get result
ResultSet result = statement.getResultSet();
if (result.next()) {
    System.out.println("Successfully login as " + result.getString("username"));
} else {
    System.out.println("Wrong username or password");
}

result.close();
```



### 3. Software Composition Analysis (SCA)

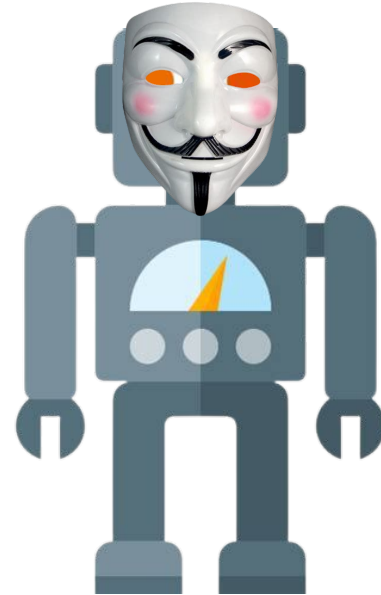
- Finds vulnerabilities in 3rd party libraries
- Checks for open CVEs on dependencies
- Fast and accurate
- CI/CD friendly





## 4. Dynamic Application Security Testing (DAST)

- Find vulnerabilities through simulated attacks on whole application
- Automated penetration testing
- Often out-of-band scans



## 5. Interactive / Real-Time Application Security Testing (IAST/RAST)

- An agent starts up with you application and instruments code
- Follow data through all components and eventually to a sink if there is a finding
- Improved DAST

```
//establishing connection and use database with the name market in the specification
Connection conn = DriverManager.getConnection("jdbc:sqlite:D:\\Java Course\\SQL\\market.db");

//scan user's input
Scanner scan = new Scanner(System.in);
System.out.println("Username : ");
String username = scan.nextLine();

System.out.println("Password : ");
String password = scan.nextLine();

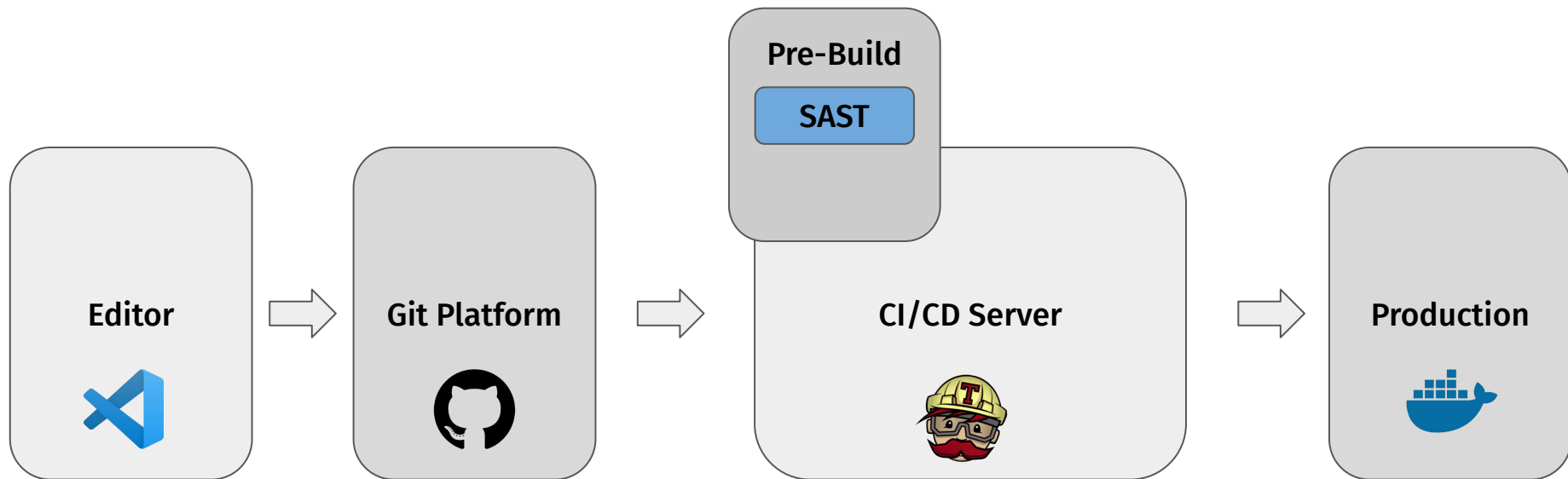
//sql statement that vulnerable to sql injection
Statement statement = conn.createStatement();
statement.execute("SELECT * FROM Mouser WHERE username=" + username + " AND password=" + password);
System.out.println("SELECT * FROM Mouser WHERE username=" + username + " AND password=" + password);

//get result
ResultSet result = statement.getResultSet();
if (result.next()) {
    System.out.println("Successfully login as " + username + " with password " + password);
} else {
    System.out.println("Wrong username or password");
}

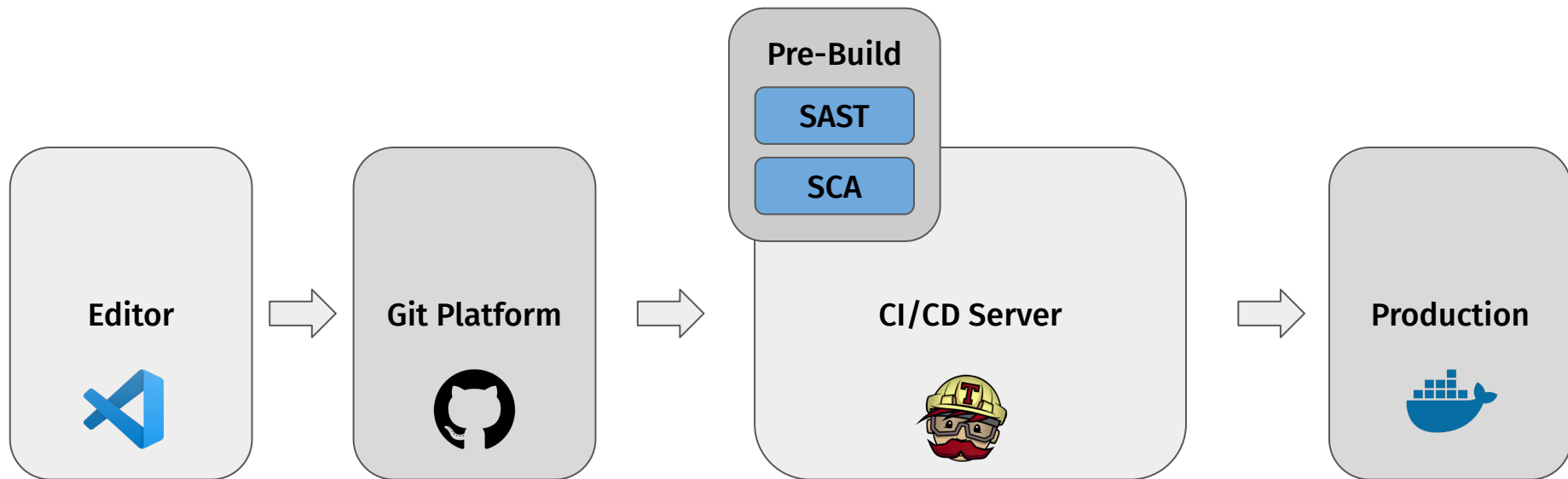
result.close();
```



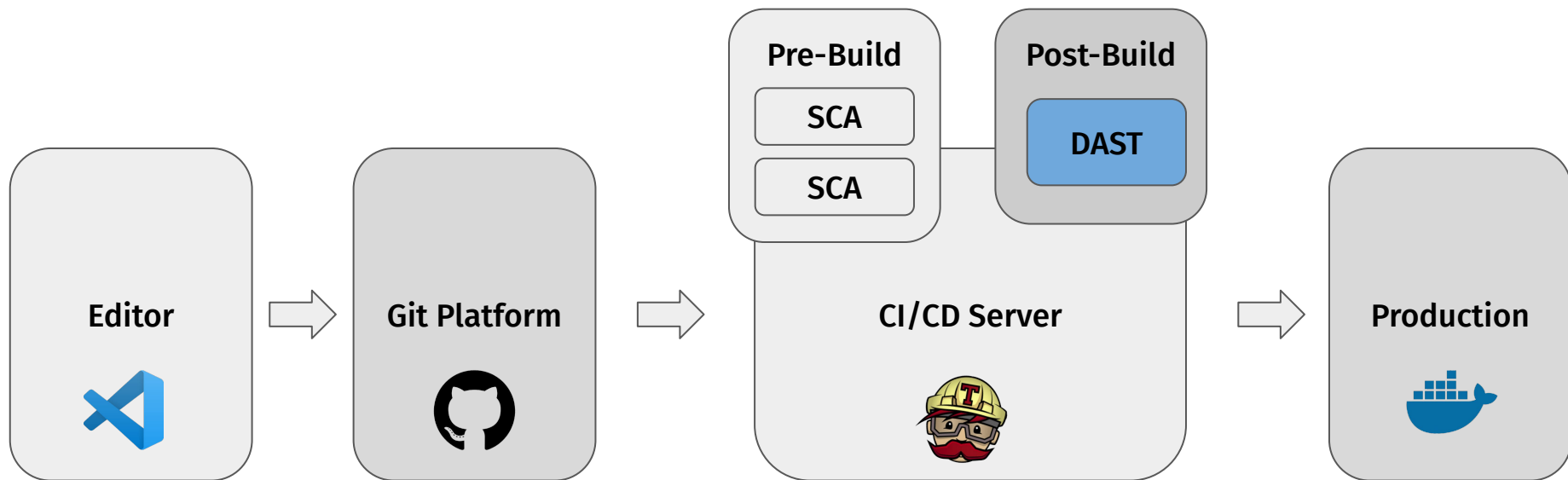
## 6. Where do they fit in our pipeline?



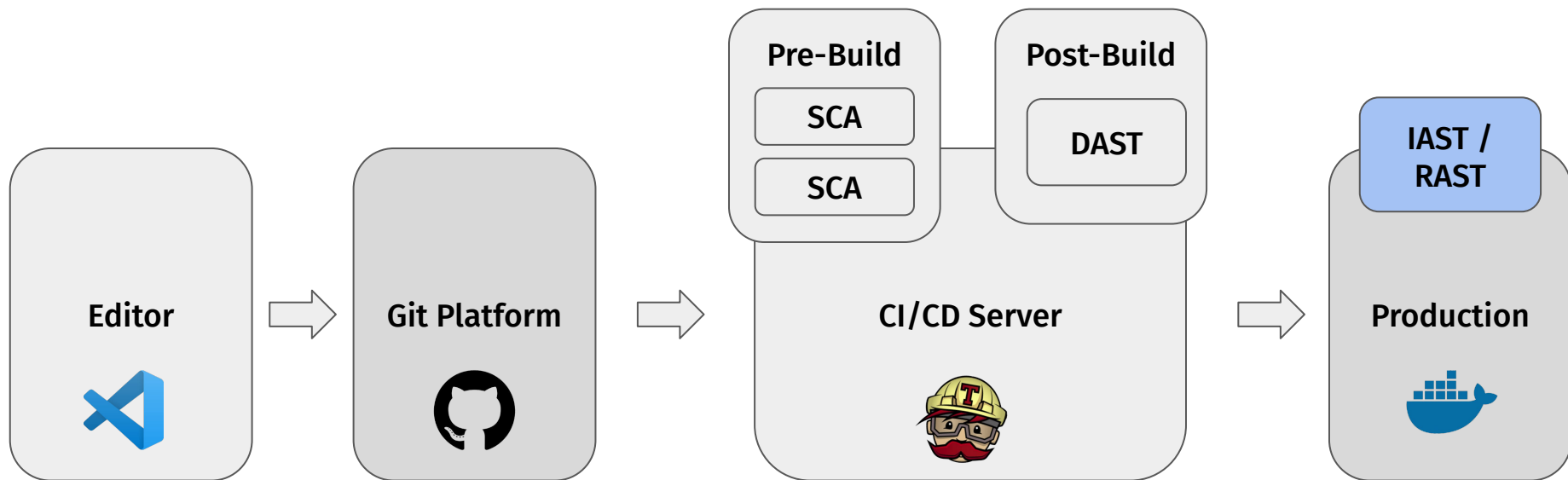
## 6. Where do they fit in our pipeline?



## 6. Where do they fit in our pipeline?



## 6. Where do they fit in our pipeline?



## 7. Which one to choose?

- Depends on the application, language and framework
- Can be integrated differently well into CI/CD pipelines
- Combination of approaches



## 8. Take home

Security can and should be integrated  
into your pipeline