CS 433 – Advance Programming

Dr. Atif Aftab Ahmed Jilani
Assistant Professor
atif.jilani@nu.edu.pk

National University
Of Computer and Emerging Sciences

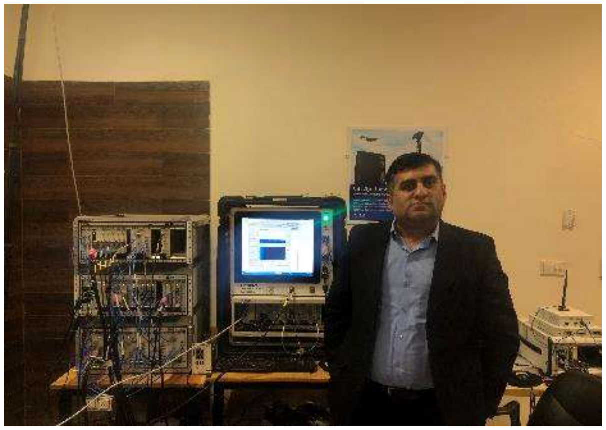
quest lab

Who am I?

- Assistant Professor @ FAST NU, Islamabad
  - PhD (CS) in the Area of Software testing, Test data Generation
  - Lead Scientist @ QUEST Lab, Pakistan
  - Certified CTFL
- Recipient of Research Grant: ~ 100 Millions for UAV Dependability Lab (NCRA)
- Over 15 high quality international research publications
  - Including multiple high impact factor journal publications
- Over 14 years of Industry and Academia experience
  - Expertise: Web Development, Computer Programming, Software Testing , Test Data Generation, Search based Software Engineering, Java (SE & EE), Microsoft .NET (C#), UML, Empirical Software Engineering
- Industry Consultants
  - MTBC Pakistan
  - PAC Kamra
- ISTQB Certified Trainer
  - PAEC, Pakistan Air Force, AvDi PAC Kamra, NESCOM, Eleven Technologies , HBL
  - FAST- NU, International Islamic University
- Voice Chair, IEEE Computer Society, Islamabad Section
- Member ACM,
- OMG Certified UML Professional

Contact Details

- **Dr. Atif Aftab Ahmed Jilani – Section A/B**
  - Email: atif.jilani@nu.edu.pk
  - Office: Room (502 -B, 5th Floor Computer Science Block C)
  - Office Hours :Will be Displayed outside my office.
- Schedule
  - 2 Classes of 1.5 hours



Projects & Activities

Lecture Format

- Start with Q & A
- Main Lecture
  - No any short break in the middle
- This is not entirely a lecture course rather an in-class Lab
  - You need to code and develop hands on experience on various programming techniques.

Overview of Course

- Goal is to give students a roadmap and make it easier for them to learn the details on their own
  - If you work hard, you will know the basics of a lot of different technologies and libraries
  - You will understand when they are used and the relative strengths and weaknesses
- The course will help to utilize the incredible power of component oriented, Network Centric and distributed computing to create effective, scalable, maintainable, and adaptable applications to solve an extremely wide range of problems using **DevOps practices and latest tools**.

Target Students

- Undergraduate computer science students interested to pursue a software development career
- The course is designed for students keen to learn programming which covers the following:

DevOps practices using Java language and how it is important in modern development

Automated Unit Testing

Configuration Management using GIT/GitHub

Writing object-oriented Programs and establishing CI/CD Continuous Integration and Continuous Deployment Pipelines

Exception Handling practices and implementation; try, catch finally block methods

I/O programming; streams, byte stream, charter stream, data stream, NIO (new IO)

Multi-Threading Programs, Synchronization

Java Data base Connectivity

Event Handling, GUI based programming; Java FX

distributed programming; client server model, socket programming; TCP, UDP, IP API,

Hibernation

Mobile Application Development using Android

Lecture Format

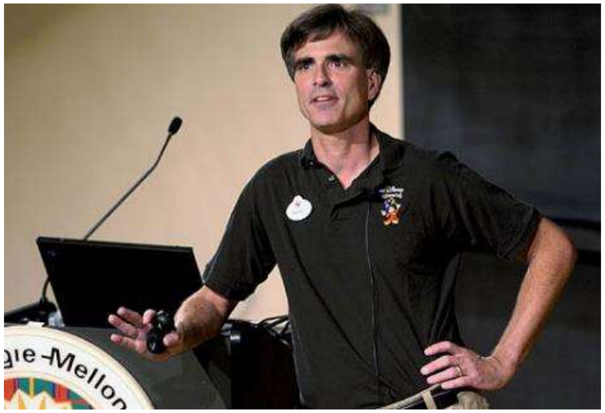
- Discussions are important
- This is not a pure lecture-based course
  - There are in-class labs
  - A parallel mandatory lab session

- Start with Q & A
- Main Lecture
  - No short breaks in the middle

Assignments and Projects

- Where ~80% of your learning will take place
- For learning, not evaluation -> low marks
- Posted to google class web site and Slate
- Usually creating complete program or parts of programs based on provided code
- Some assignments done as individual, some can be done with a partner
- Sometime test cases provided
- Create your own test cases
- Graded on correctness, style, efficiency, generality, comments, testing
  - not graded on a linear scale or on effort
- Program must work, compile errors / runtime errors lose all correctness points
- Copying solution code or giving code to someone else is CHEATING -> F in the course
- Sharing test cases okay and encouraged

Succeeding in the Course



- Randy Pausch,
  CS Professor at CMU said:

- "When I got tenure a year early at Virginia, other Assistant Professors would come up to me and say, 'You got tenure early!?!?! What's your secret?!?!?' and I would tell them, 'Call me in my office at 10pm on Friday night and I'll tell you.' "

- Meaning: Some things don't have an easy solution.
- Some things simply require a lot of hard work.

Succeeding in the Course

- Download code and lecture materials from the course folder
- Walk through examples and browse documentation (2 to 3 hours)
- Keep on learning new techniques
- Do **Javabat problems (http://codingbat.com/java)**
- Do the extra section problems
- Start on assignments early
- Ask questions and get help when needed
- Read the “Code of honor”.
- Talk with your peers
  - Share ideas, not code/syntax.
  - Mention it in the report.
- Don’t get stuck for too long

Tentative Marks Distribution

Assignments	20	%
Project	10	%
Quiz	5	%
Mid Term Exam	25	%
Final	40	%
Total	100	%

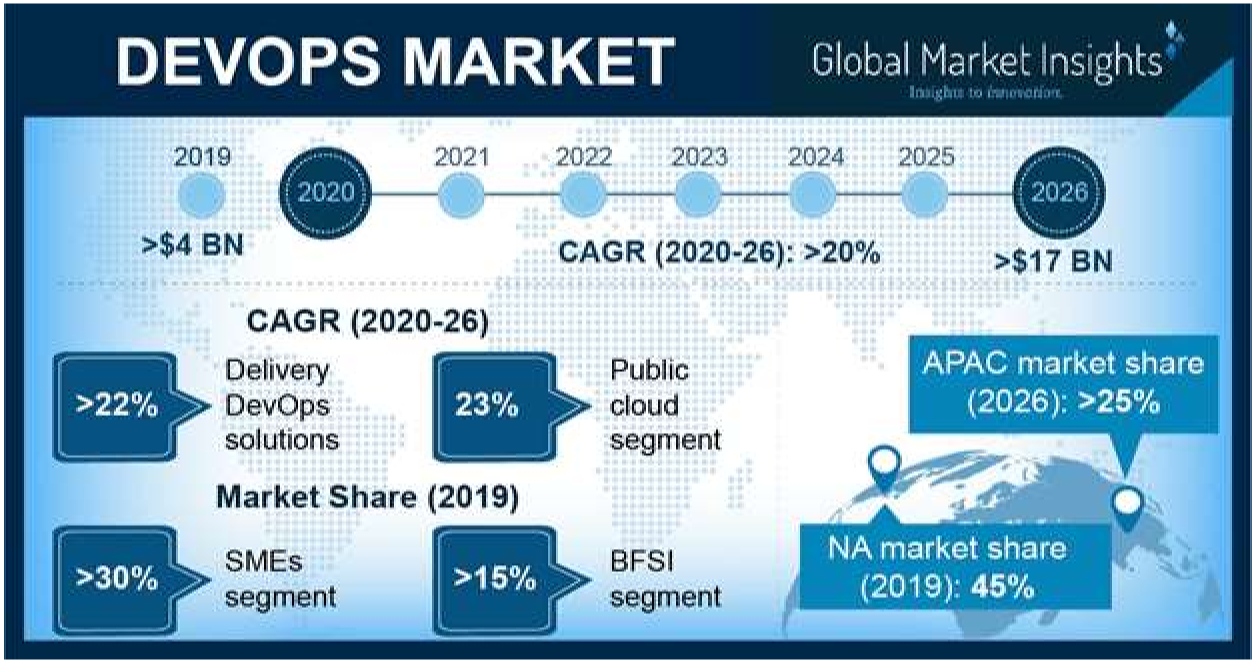
LET'S SEE WHERE THE WORLD IS MOVING !

ACADEMYSMART.COM

TOP 15 HIGHEST PAYING SOFTWARE JOBS IN 2021

1. BIG DATA ENGINEER
2. DEVOPS ENGINEER
3. INFORMATION SYSTEMS SECURITY MANAGER
4. MOBILE APPLICATIONS DEVELOPER
5. APPLICATIONS ARCHITEC
6. DATA ARCHITECT
7. DATABASE MANAGER
8. DATA SECURITY ANALYST
9. DATA SCIENTIST
10. NETWORK/CLOUD ARCHITECT
11. NETWORK/CLOUD ENGINEER
12. SENIOR WEB DEVELOPER
13. SITE RELIABILITY ENGINEER
14. SYSTEMS ENGINEER
15. SOFTWARE ENGINEER

DevOps Market Trend



DEVOPS MARKET

Global Market Insights

Insights to innovation.

2019 2020 2021 2022 2023 2024 2025 2026

>$4 BN

CAGR (2020-26): >20%

>$17 BN

CAGR (2020-26)

>22% Delivery DevOps solutions

23% Public cloud segment

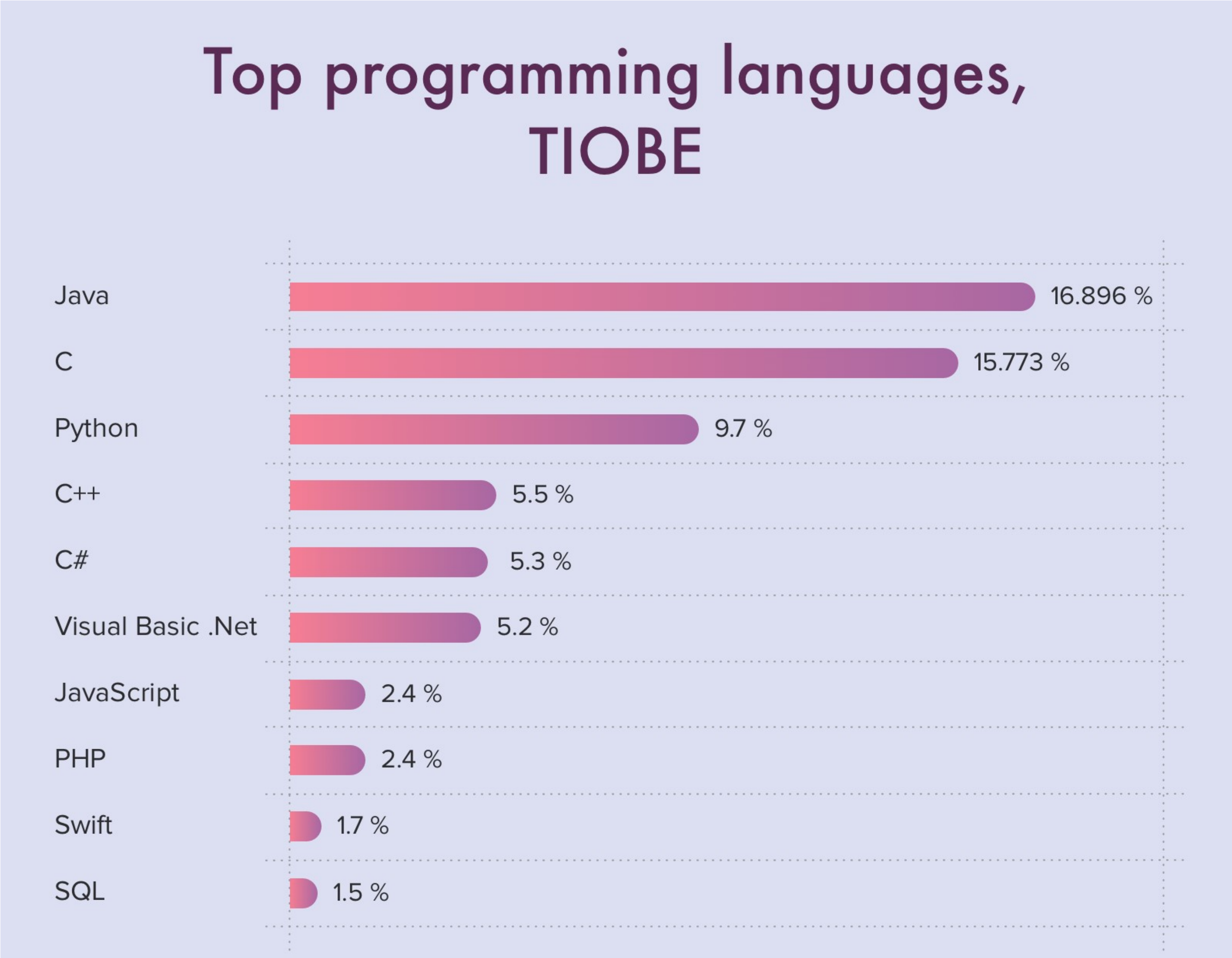
Market Share (2019)

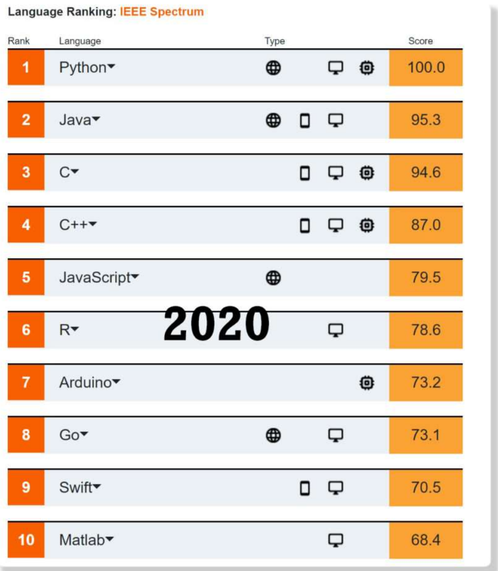
>30% SMEs segment

>15% BFSI segment

APAC market share (2026): **>25%**

NA market share (2019): **45%**

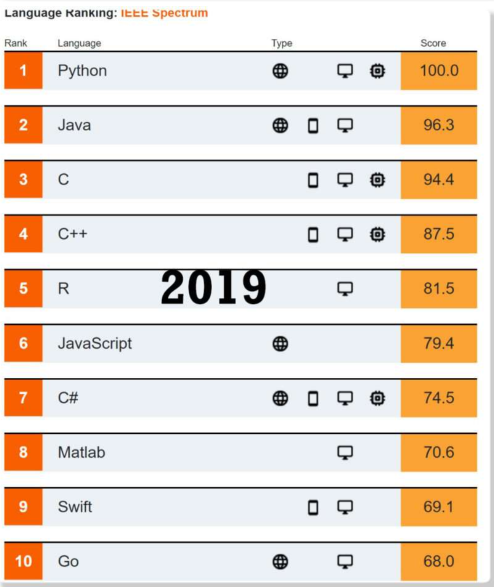

Top programming languages, TIOBE
Java
16.896 %
C
15.773 %
Python
9.7 %
C++
5.5 %
C#
5.3 %
Visual Basic .Net
5.2 %
JavaScript
2.4 %
PHP
2.4 %
Swift
1.7 %
SQL
1.5 %



Language Ranking: IEEE Spectrum

Rank	Language	Type	Score
1	Python▾	🌐 🖥 ⚙	100.0
2	Java▾	🌐 📱 🖥	95.3
3	C▾	📱 🖥 ⚙	94.6
4	C++▾	📱 🖥 ⚙	87.0
5	JavaScript▾	🌐	79.5
6	R▾	🖥	78.6
7	Arduino▾	⚙	73.2
8	Go▾	🌐 🖥	73.1
9	Swift▾	📱 🖥	70.5
10	Matlab▾	🖥	68.4

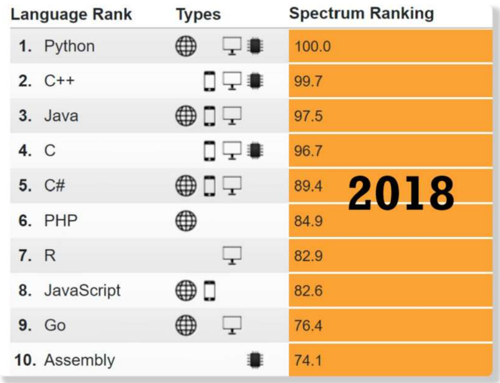
2020



Language Ranking: IEEE Spectrum

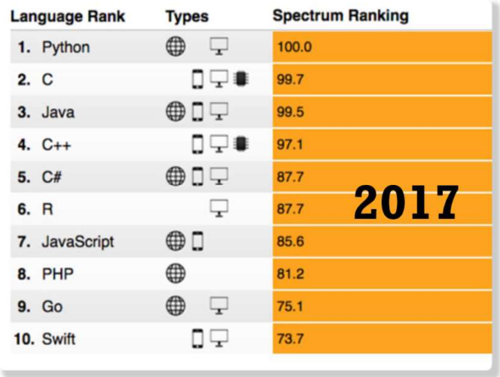
Rank	Language	Type	Score
1	Python	🌐 🖥 ⚙	100.0
2	Java	🌐 📱 🖥	96.3
3	C	📱 🖥 ⚙	94.4
4	C++	📱 🖥 ⚙	87.5
5	R	🖥	81.5
6	JavaScript	🌐	79.4
7	C#	🌐 📱 🖥 ⚙	74.5
8	Matlab	🖥	70.6
9	Swift	📱 🖥	69.1
10	Go	🌐 🖥	68.0

2019



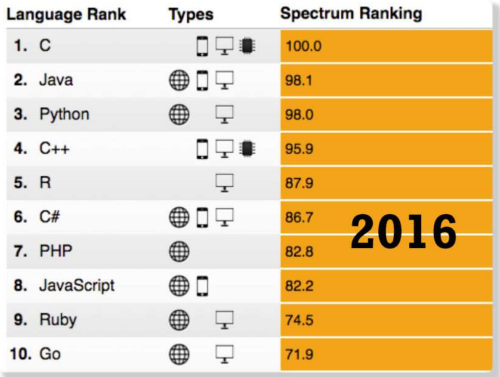
Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥 ⚙	100.0
2. C++	📱 🖥 ⚙	99.7
3. Java	🌐 📱 🖥	97.5
4. C	📱 🖥 ⚙	96.7
5. C#	🌐 📱 🖥	89.4
6. PHP	🌐	84.9
7. R	🖥	82.9
8. JavaScript	🌐 📱	82.6
9. Go	🌐 🖥	76.4
10. Assembly	⚙	74.1

2018



Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥	100.0
2. C	📱 🖥 ⚙	99.7
3. Java	🌐 📱 🖥	99.5
4. C++	📱 🖥 ⚙	97.1
5. C#	🌐 📱 🖥	87.7
6. R	🖥	87.7
7. JavaScript	🌐 📱	85.6
8. PHP	🌐	81.2
9. Go	🌐 🖥	75.1
10. Swift	📱 🖥	73.7

2017



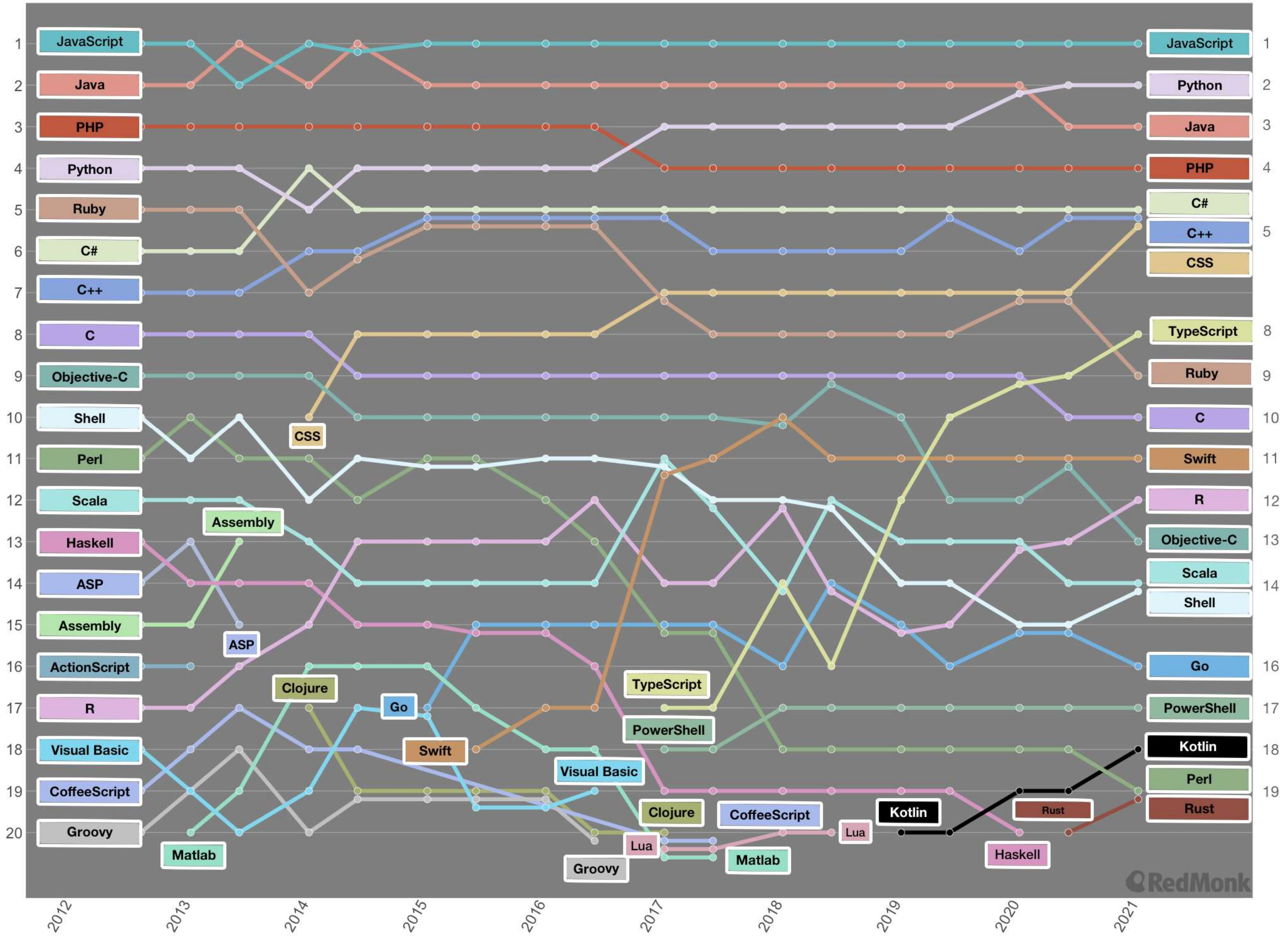
Language Rank	Types	Spectrum Ranking
1. C	📱 🖥 ⚙	100.0
2. Java	🌐 📱 🖥	98.1
3. Python	🌐 🖥	98.0
4. C++	📱 🖥 ⚙	95.9
5. R	🖥	87.9
6. C#	🌐 📱 🖥	86.7
7. PHP	🌐	82.8
8. JavaScript	🌐 📱	82.2
9. Ruby	🌐 🖥	74.5
10. Go	🌐 🖥	71.9

2016

Source: spectrum.ieee.org/static/interactive-the-top-programming-languages-2020

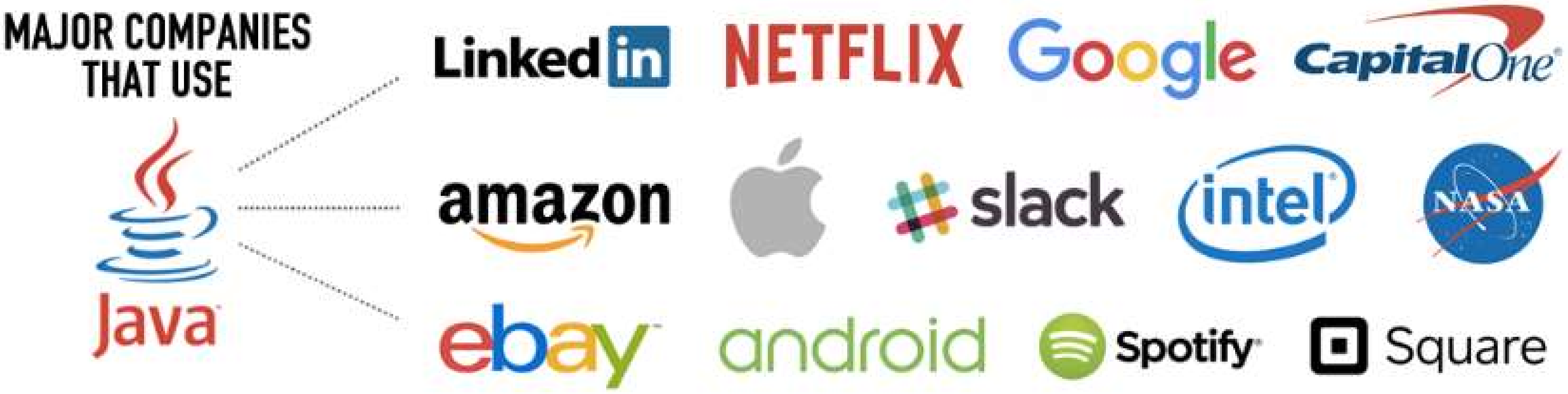
RedMonk Language Rankings

September 2012 - January 2021



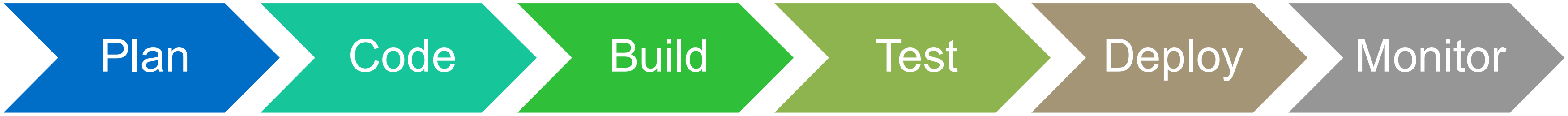
quest lab

Java Spread


MAJOR COMPANIES THAT USE
Java
LinkedIn
NETFLIX
Google
CapitalOne
amazon
slack
intel
NASA
ebay
android
Spotify
Square

CODE
PLAN
DEPLOY
BUILD
DEV
RELEASE
OPS
OPERATE
TEST
MONITOR

Traditional Software Development Life Cycle



Plan

Code

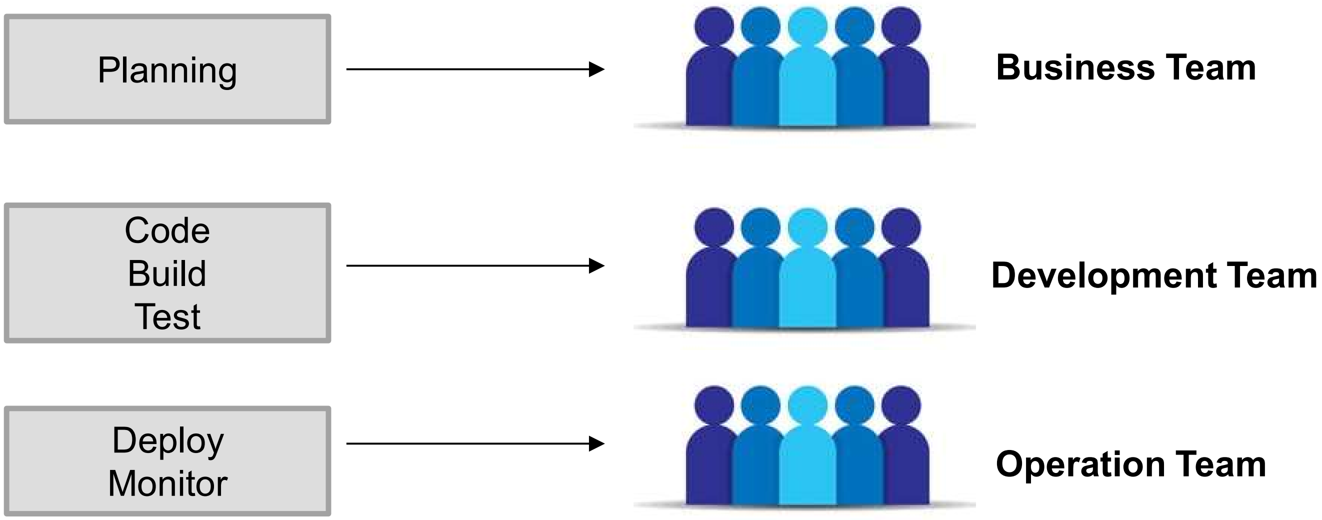
Build

Test

Deploy

Monitor

Teams

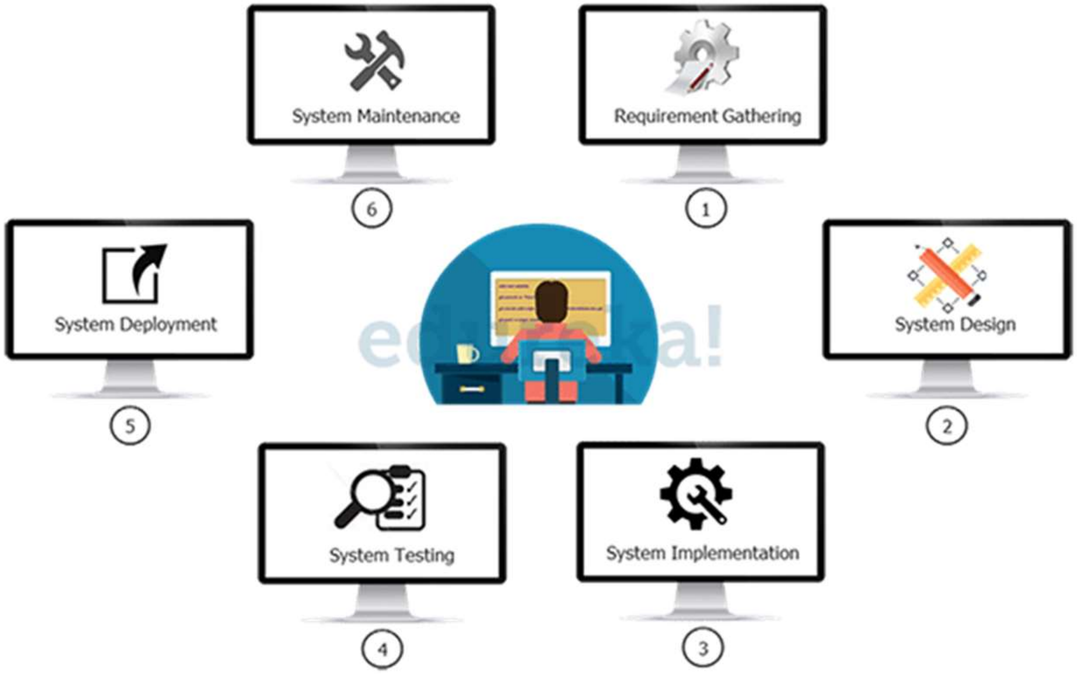


Planning → Business Team

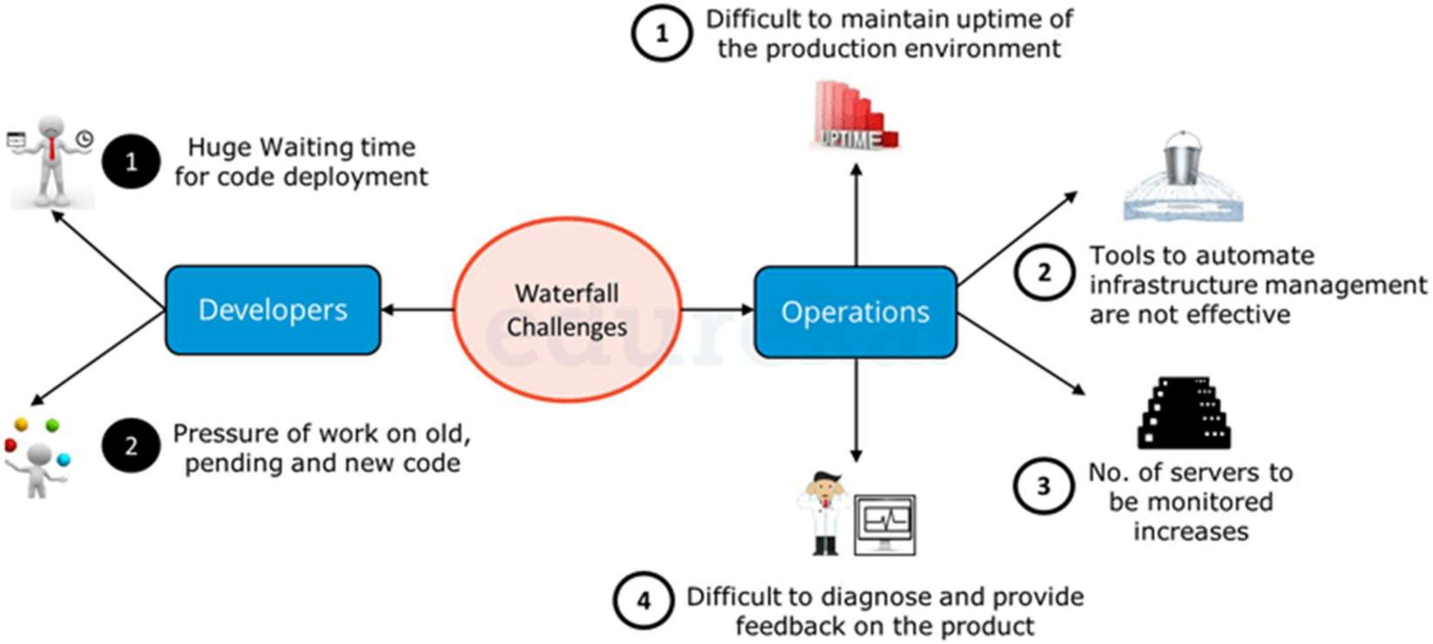
Code
Build
Test → Development Team

Deploy
Monitor → Operation Team

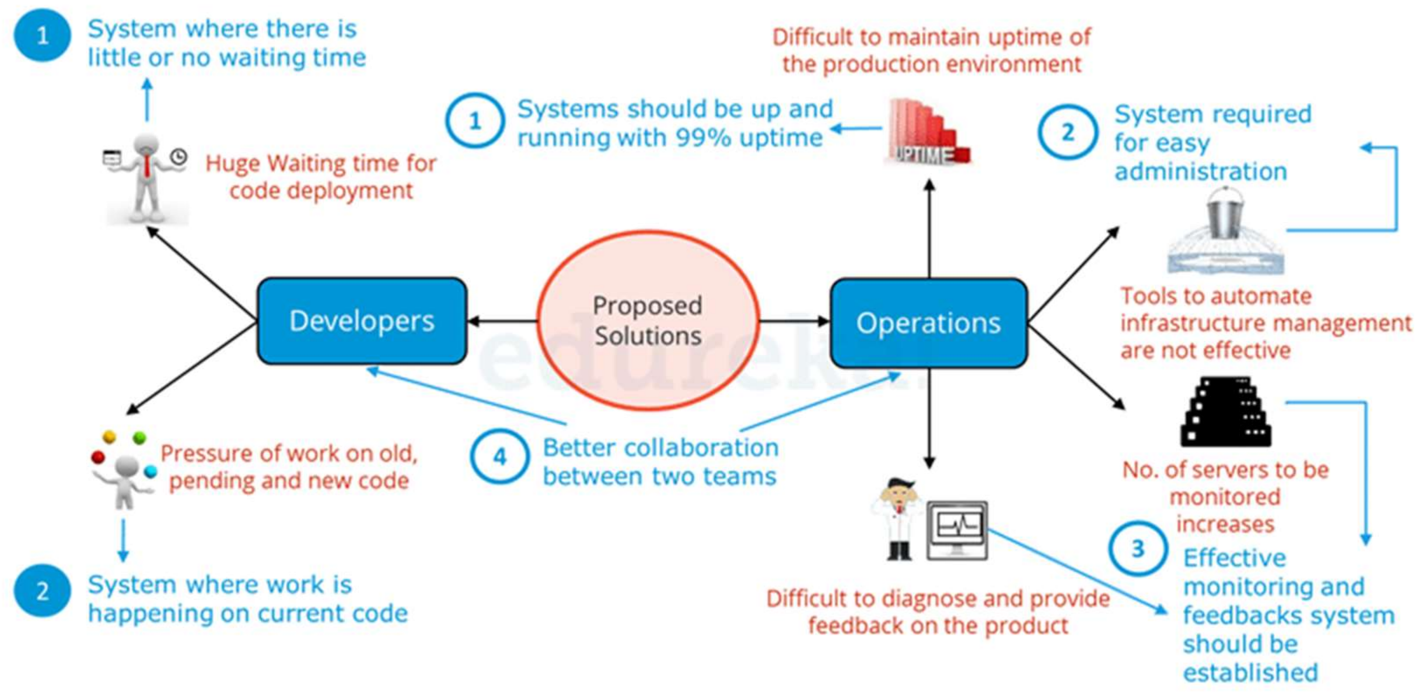
Traditional Process Models- Waterfall



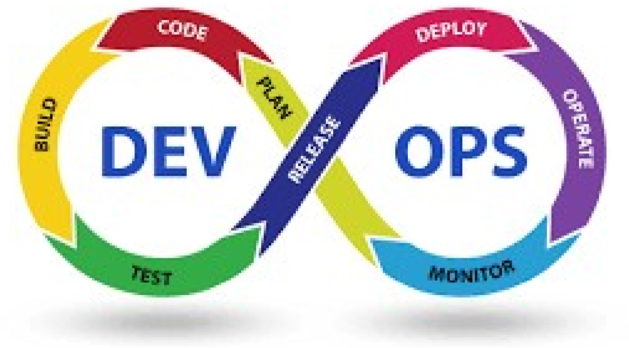
Challenges


1
Difficult to maintain uptime of the production environment
UPTIME
1
Huge Waiting time for code deployment
Developers
Waterfall Challenges
Operations
2
Tools to automate infrastructure management are not effective
2
Pressure of work on old, pending and new code
3
No. of servers to be monitored increases
4
Difficult to diagnose and provide feedback on the product

Solution Proposed to the challenges of Waterfall Model


1
System where there is little or no waiting time
Huge Waiting time for code deployment
Developers
Proposed Solutions
Operations
Pressure of work on old, pending and new code
2
System where work is happening on current code
4
Better collaboration between two teams
Difficult to maintain uptime of the production environment
1
Systems should be up and running with 99% uptime
UPTIME
2
System required for easy administration
Tools to automate infrastructure management are not effective
No. of servers to be monitored increases
3
Effective monitoring and feedbacks system should be established
Difficult to diagnose and provide feedback on the product

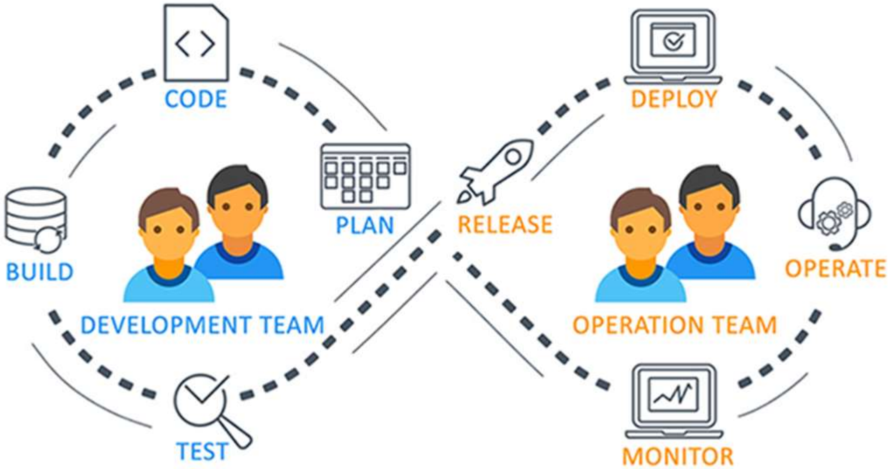
DevOps



DevOps Lifecycle

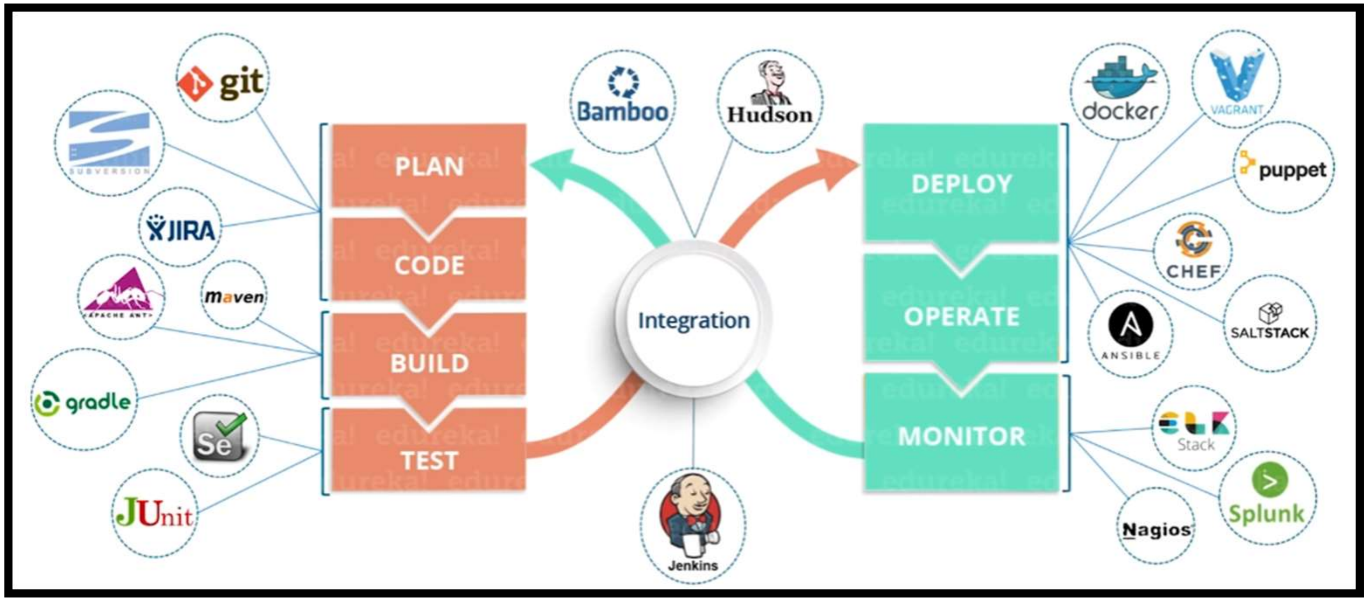
Development and Operations

- DevOps integrates developers and operations team to improve collaboration and productivity.



“A single group of Engineers (developers, system admins, QA’s. Testers etc. turned into DevOps Engineers) has end to end responsibility of the Application (Software) right from gathering the requirement to development, to testing, to infrastructure deployment, to application deployment and finally monitoring & gathering feedback from the end users, then again implementing the changes.”

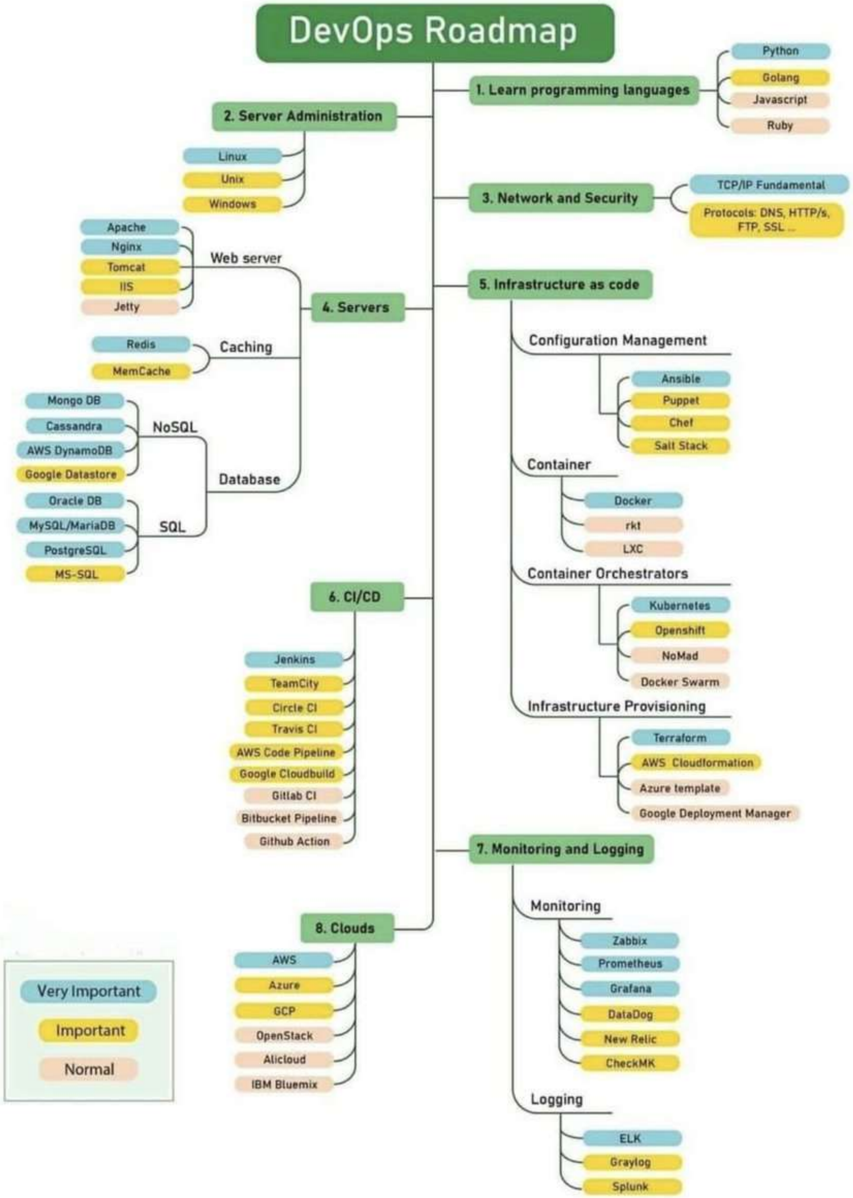
DevOps diagram with various DevOps Tools



DevOps Roadmap

- Learn Programming Language
- Server Administration
- Networks and Security
- Servers
- Infrastructure as code
- CI/CD
- Monitoring
- Clouds

Detailed View



DevOps Teams

- **Developers & Designers (Devs)** build the software logic following user stories, create features that work and prove that via unit tests.
- **Quality engineers (QEs)** helps maintain software quality, review the features, write acceptance/end-to-end tests, and approve the software correspondence to the requirements.
- **Product owners (POs) & Business Analysts (BAs)** cover the aspect of business value for the end-user, coming up with the user stories. Often, they have to coordinate and analyze the acceptance test results to check their relevance and implement changes in user story when necessary.
- **Operations (Ops)** take care of the software released. They make sure it is available for the users. Thus, they work on the code logistics design to move developers` code to a production environment where people access the website/app.

The 2021 DevOps roadmap

1. Learning a programming language

- Get a good grasp of a programming language. It doesn't matter which one, but it's needed for writing automation code.

2. Understand different OS concepts

- Need to learn about process management, threads & concurrency, sockets, I/O management, virtualization, memory system, etc.

3. Learn to Live in terminal

- Terminal commands are essential for a DevOps engineer for monitoring, text manipulation, system performance, etc

The 2021 DevOps roadmap

4. Network, Security & Protocols

- Need to be familiar with various types of protocols which play a major role in communicating with different devices across the network like TCP/IP, HTTP, HTTPS, SMTP, FTP etc.

5. What is and how to setup

- A DevOps engineer should know how to set up a web server like IIS, Apache Tomcat.

6. Learn Infrastructure as code

- This is one of the most critical component in the learning path of a DevOps engineer. Need to learn about app containerization and have thorough understand of container tools like Docker and Kubernetes

The 2021 DevOps roadmap

7. Learn some Continuous Integration and Delivery (CI/CD) tools

- Continuous Integration/Continuous Deployment is now a core part of setting a DevOps culture. Get familiar with CI/CD tools like Gitlab, Jenkins, GitHub actions etc.

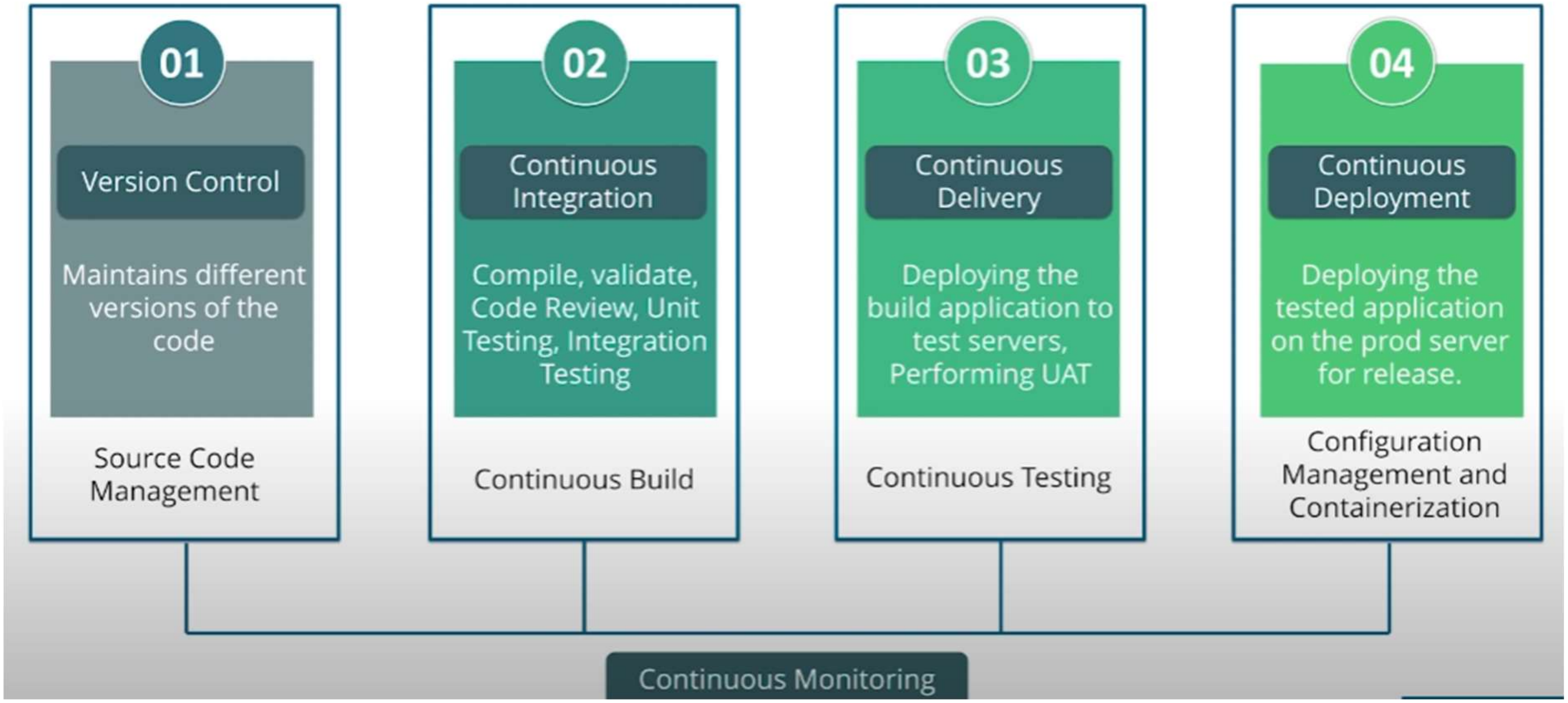
8. Learn to monitor software and infrastructure

- When you have thousands of services running, it's important to make sure that the system is running in fine health.

9. Learn about Cloud Providers

- Most of the apps today are built as cloud-native. AWS, Azure and Google Cloud are the leading players and they provide free courses about their tools too.

DevOps Stages



01

Version Control

Maintains different versions of the code

Source Code Management

02

Continuous Integration

Compile, validate, Code Review, Unit Testing, Integration Testing

Continuous Build

03

Continuous Delivery

Deploying the build application to test servers, Performing UAT

Continuous Testing

04

Continuous Deployment

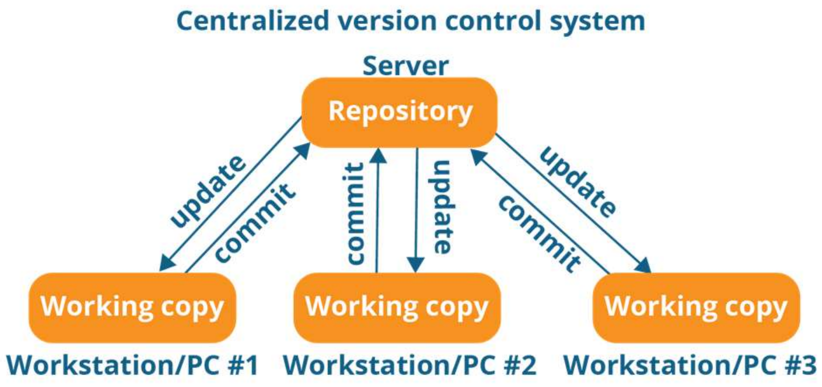
Deploying the tested application on the prod server for release.

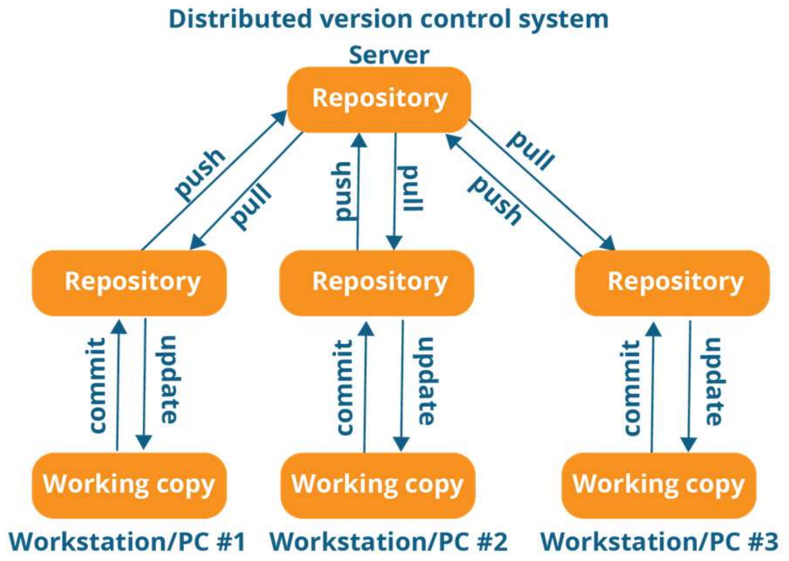
Configuration Management and Containerization

Continuous Monitoring

Stage#1: Source Code Management

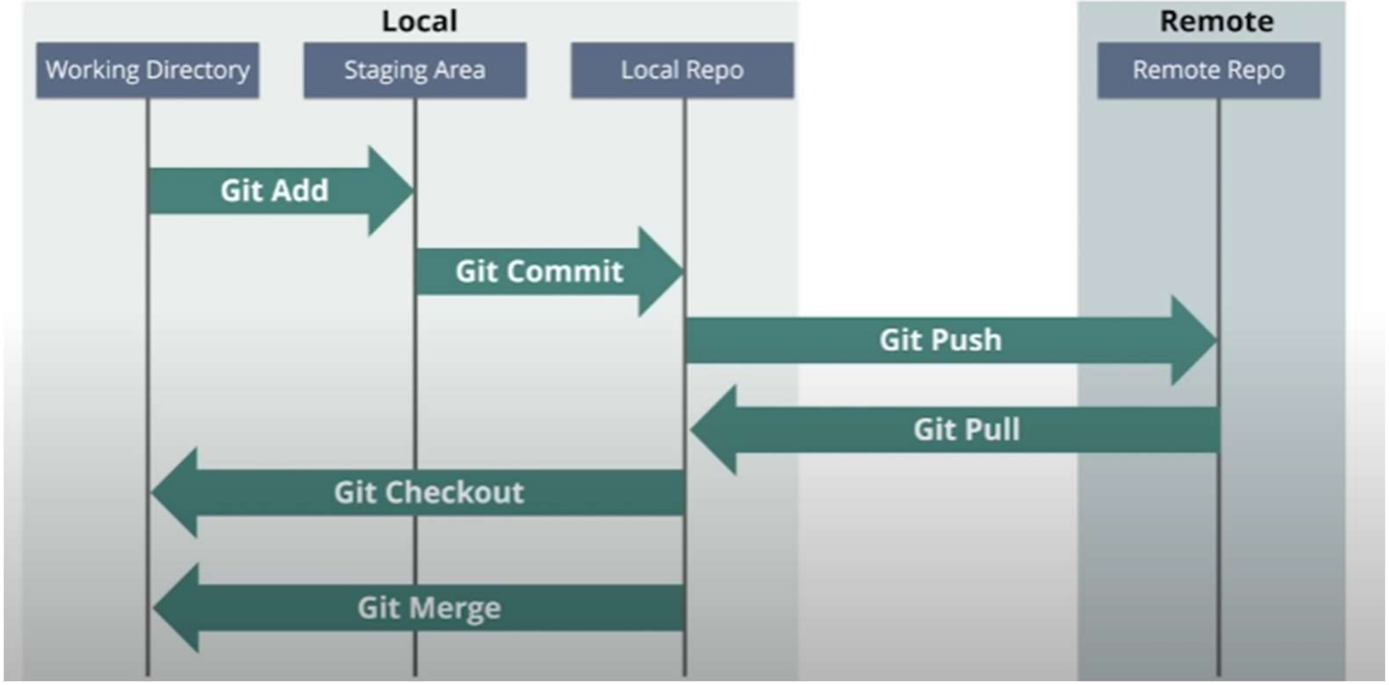
- The management of changes to documents, computer programs, large websites and other collection of information.



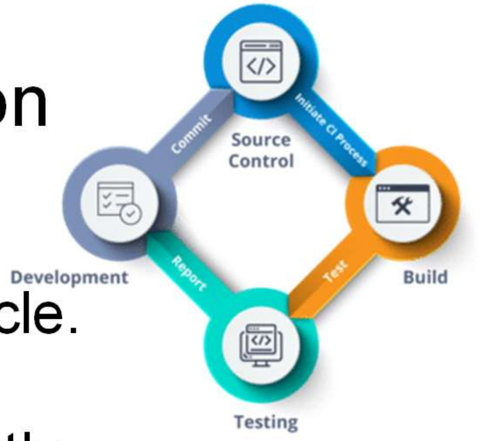


Source Code Management

- Git is a Distributed Version Control tool that supports distributed non-linear workflows by providing data assurance for developing quality software.

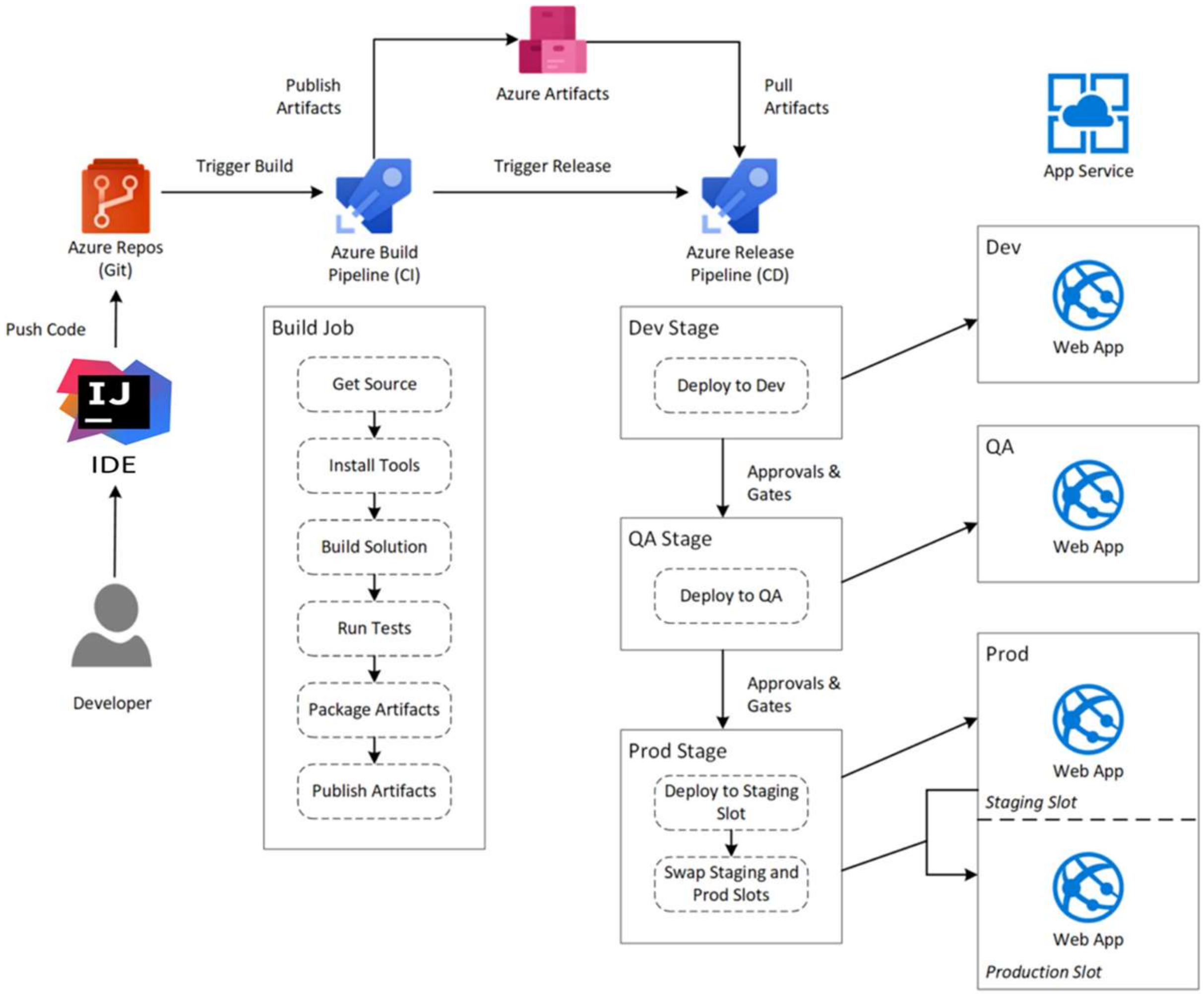


Stage#2: Continuous Integration



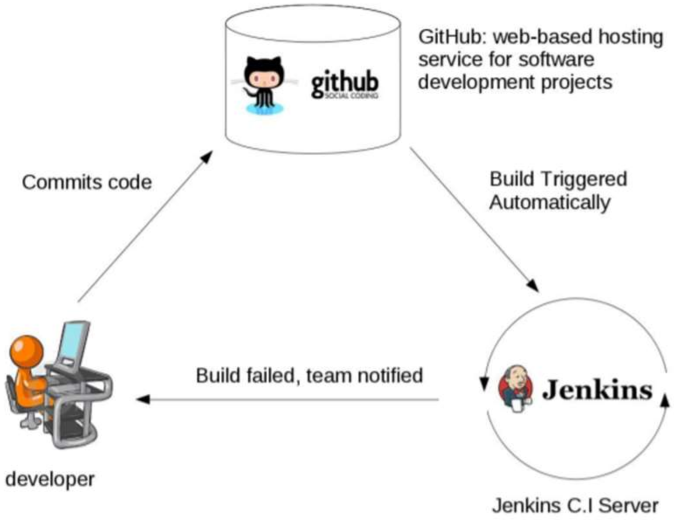
- This stage is the core of the entire DevOps life cycle. It is a practice in which the developers require to commit changes to the source code more frequently.
- Every commit is build and this allows early detection of problems if they are present. Building code not only involves compilation but it also includes code review, unit testing, integration testing, and packaging.
- The code supporting new functionality is continuously integrated with the existing code.
- There are tools for building/ packaging the code into an executable file so that you can forward it to the next phases.

CONTINUOUS INTEGRATION – AZURE DEVOPS



Continuous Integration through

- Developers commit code to a shared repository on a regular basis.
- Version control system is being monitored. When a commit is detected, a build will be triggered automatically.
- If the build is not green, developers will be notified immediately

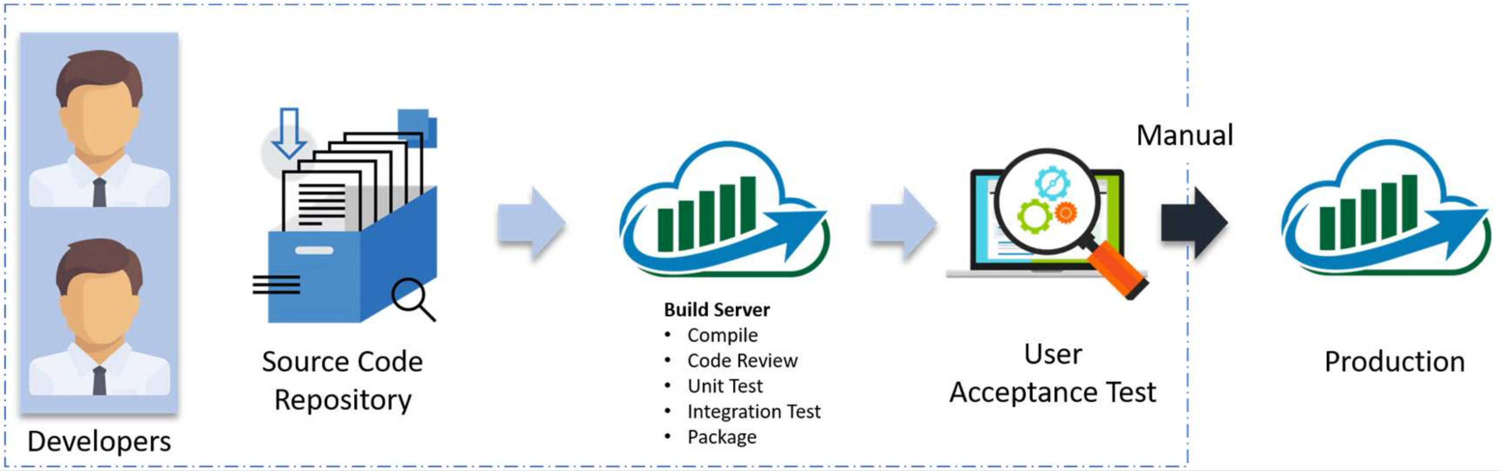

github
GitHub: web-based hosting service for software development projects
Commits code
Build Triggered Automatically
Build failed, team notified
Jenkins
developer
Jenkins C.I Server

Why do we need Continuous Integration?

- Detect problems or bugs, as early as possible, in the development life cycle.
- Since the entire code base is integrated, built and tested constantly , the potential bugs and errors are caught earlier in the life cycle which results in better quality software.

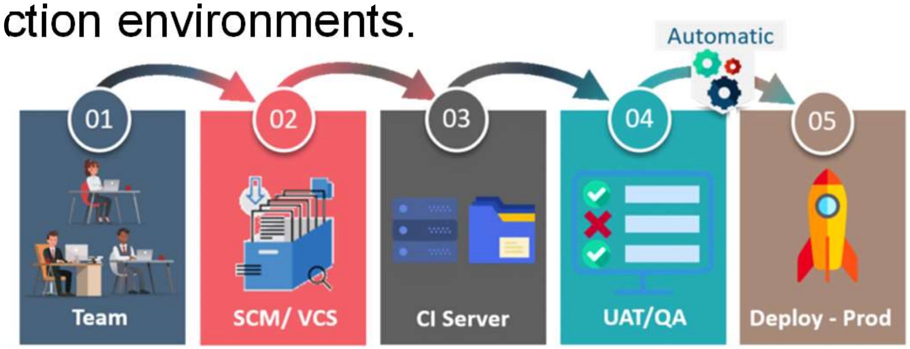
Stage#3: Continuous Testing

- This stage is responsible for testing the developed software continuously for bugs using automation testing tools. (Test environment)
- Selenium is used for automation testing, and the reports are generated by TestNG.
- You can automate this entire testing phase with the help of a Continuous Integration tool called Azure DevOps, Jenkins.

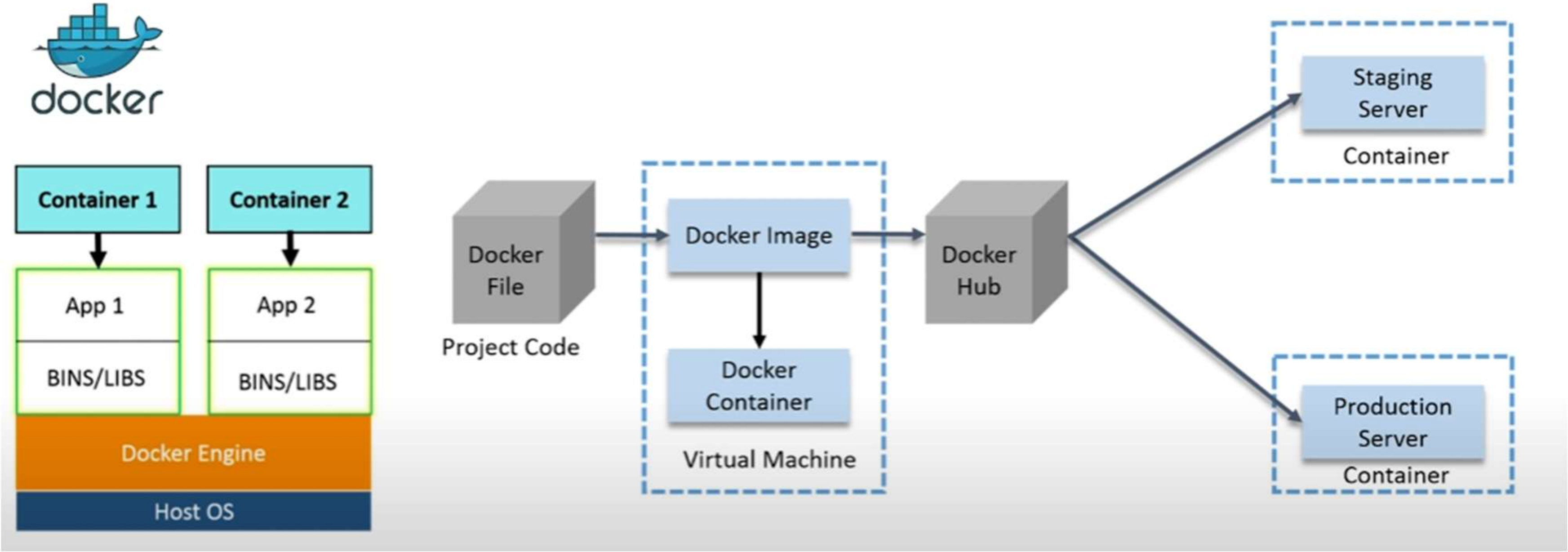


Stage#4: Continuous Deployment

- This stage is responsible for deployment of code on the production servers. It is also important to ensure that you correctly deploy the code on all the servers. There are set of tools here help in achieving Continuous Deployment (CD).
  - Configuration Management is the act of establishing and maintaining consistency in an application's functional requirements and performance.
  - Containerization tools also play an equally crucial role in the deployment stage. The containerization tools help produce consistency across Development, Test, Staging as well as Production environments.



Example: Containerization

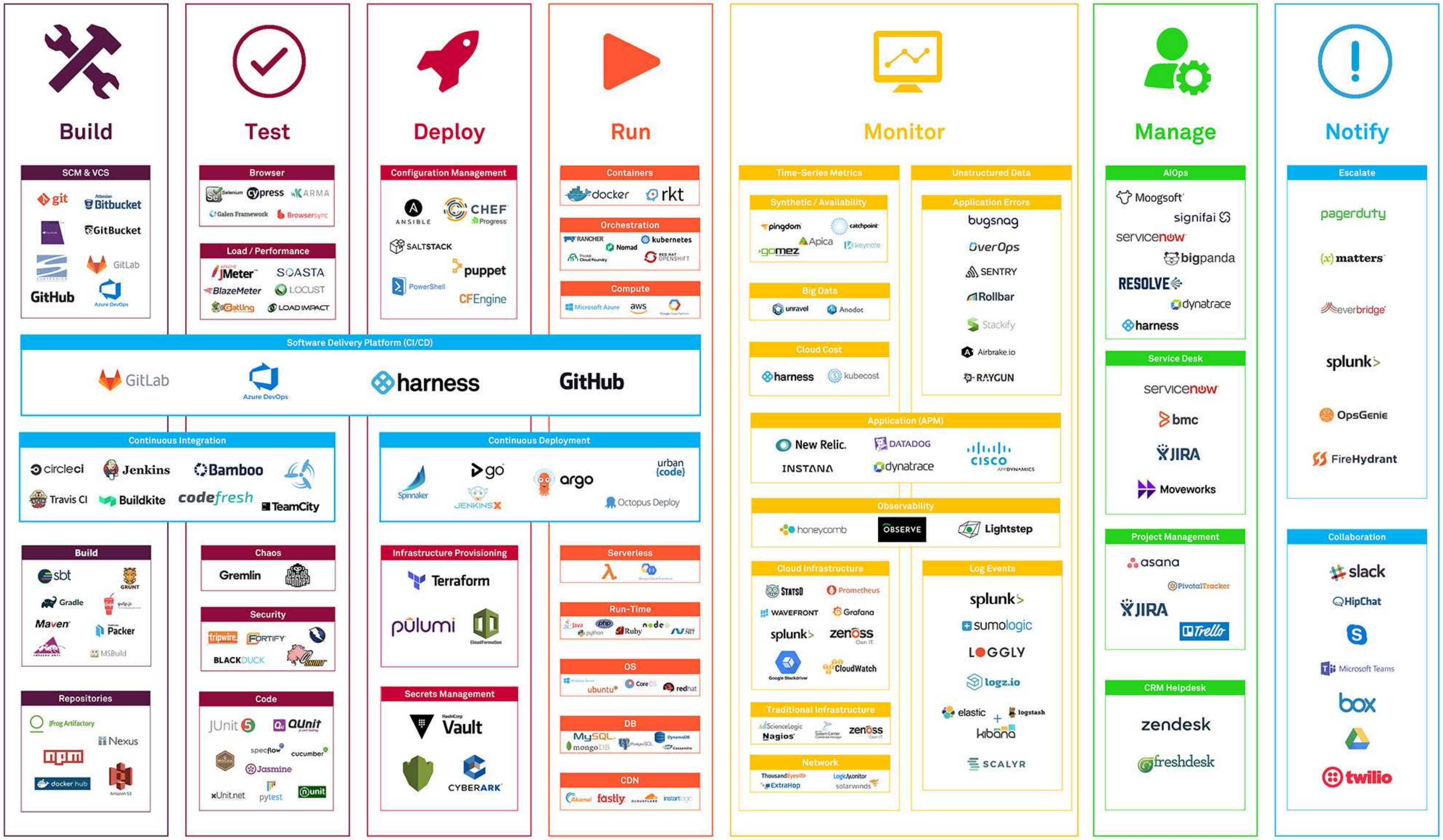


Stage#5: Continuous Monitoring

- This is a very critical stage of the DevOps life cycle where you continuously monitor the performance of your application. It records the following vital information about the software.
  - check the proper functionality of the application.
  - resolve system errors such as low memory, server not reachable, etc.

- This practice involves the participation of the Operations team who will monitor the user activity for bugs or any improper behavior of the system.

DevOps Tools Ecosystem 2021



harness

The Platform for Software Delivery
www.harness.io

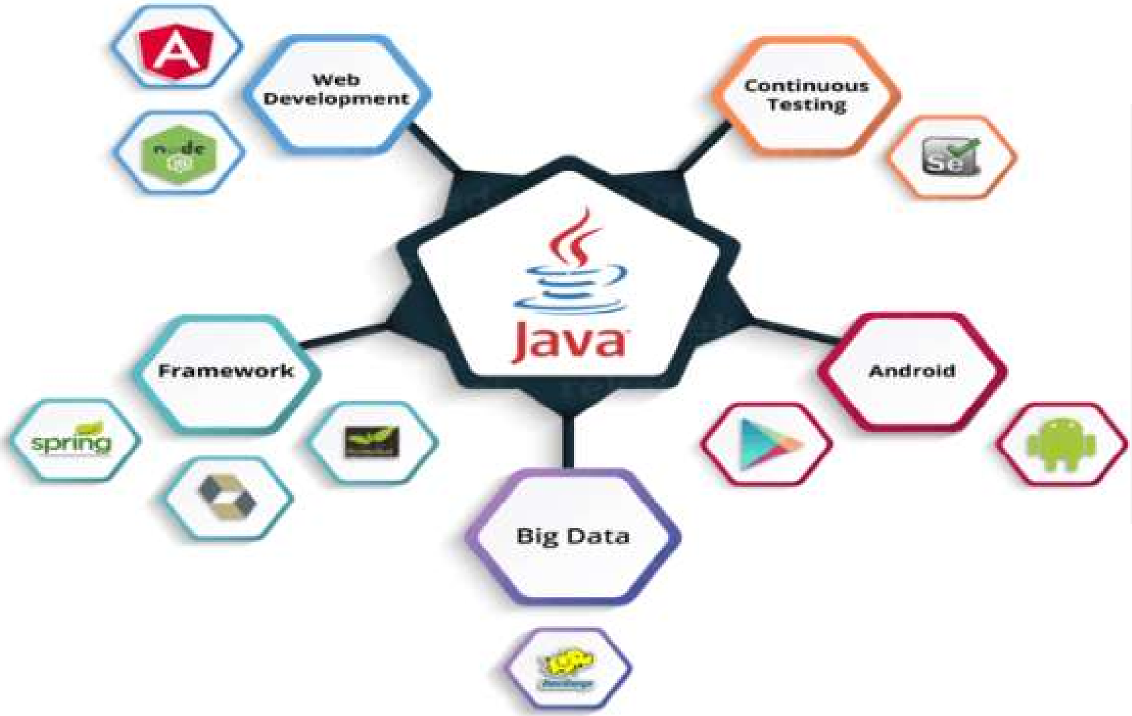
Learning a Programming Language : Java

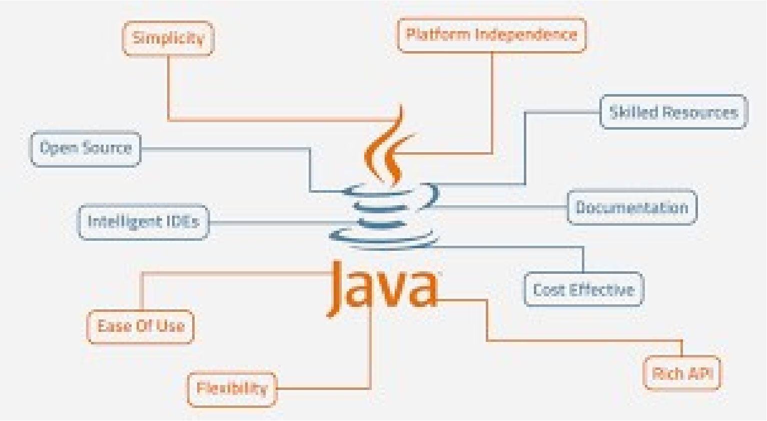
MAJOR COMPANIES THAT USE
Java
LinkedIn
NETFLIX
Google
CapitalOne
amazon
slack
intel
NASA
ebay
android
Spotify
Square

A lot of companies are using Java now-a-days, rather than other languages due to a simple fact: Java is awesomely **platform independent** and **reliable** which makes in famous in almost all fields including android, ios, smart TV, web development.

What about Java?

- A widely used and effective OOP language
- Over 10 million developers and 15 billion devices run Java worldwide!





Platform Independence. How does Java do it?

- Java has been described as WORA (Write once, Run Anywhere)

- Because Java source code is compiled to byte code and the byte code is interpreted, Java code can be executed anywhere an interpreter is available.

- The "Interpreter" is call the Java Virtual Machine

How Java Works

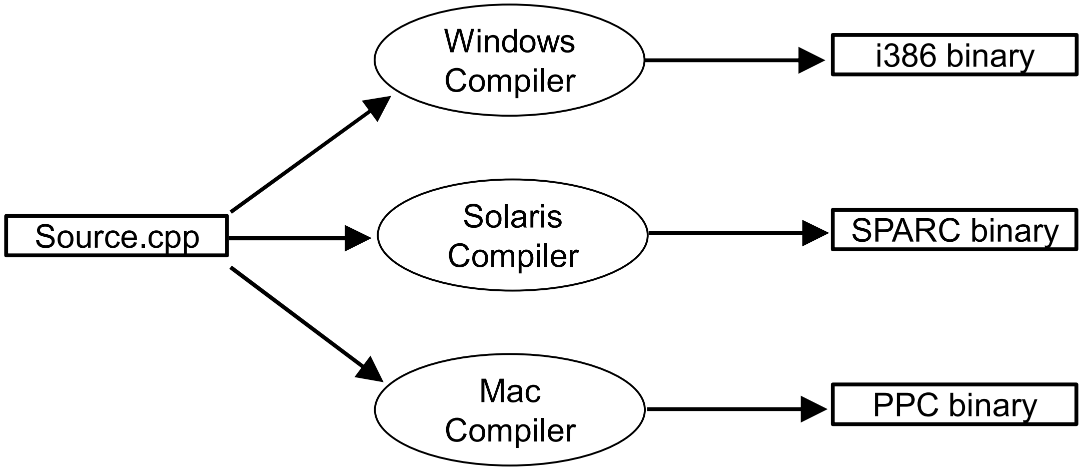
- Java's **platform independence** is achieved by the use of the **Java Virtual Machine**
- A Java program consists of one or more files with a **.java** extension
  - these are plain old text files
- When a Java program is compiled the **.java** files are fed to a compiler which produces a **.class** file for each .java file
- The **.class** file contains Java bytecode.
- **Bytecode** is like machine language, but it is intended for the Java Virtual Machine not a specific chip such as a Pentium or PowerPC chip

More on How Java Works

- To run a Java program the bytecode in a .class file is fed to an interpreter which converts the byte code to machine code for a specific chip (IA-32, PowerPC)
- Some people refer to the interpreter as "The Java Virtual Machine" **(JVM)**
- The interpreter is platform specific because it takes the platform independent bytecode and produces machine language instructions for a particular chip
- So, a Java program could be run an any type of computer that has a JVM written for it.
  - **PC, Mac, Unix, Linux, BeaOS, Sparc**

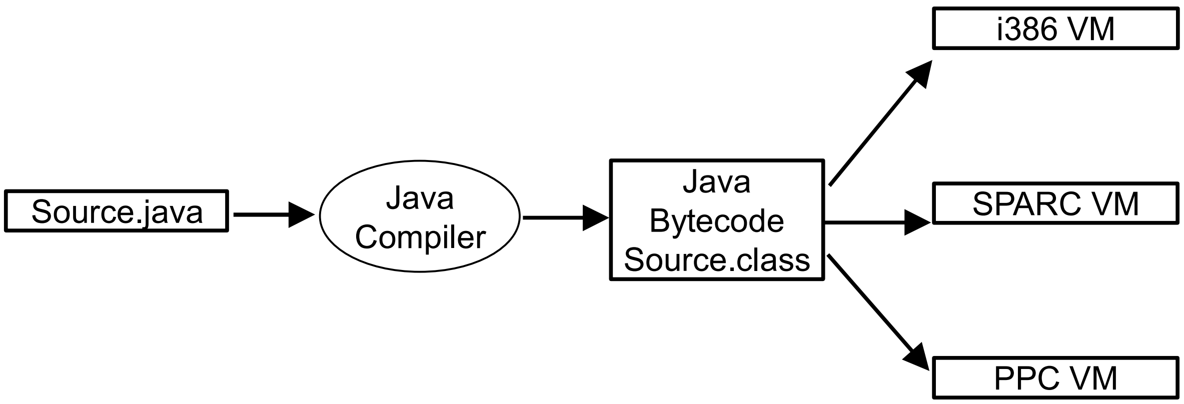
The Java Virtual Machine.

- Traditionally, source code had to be compiled for the target hardware and OS platform:

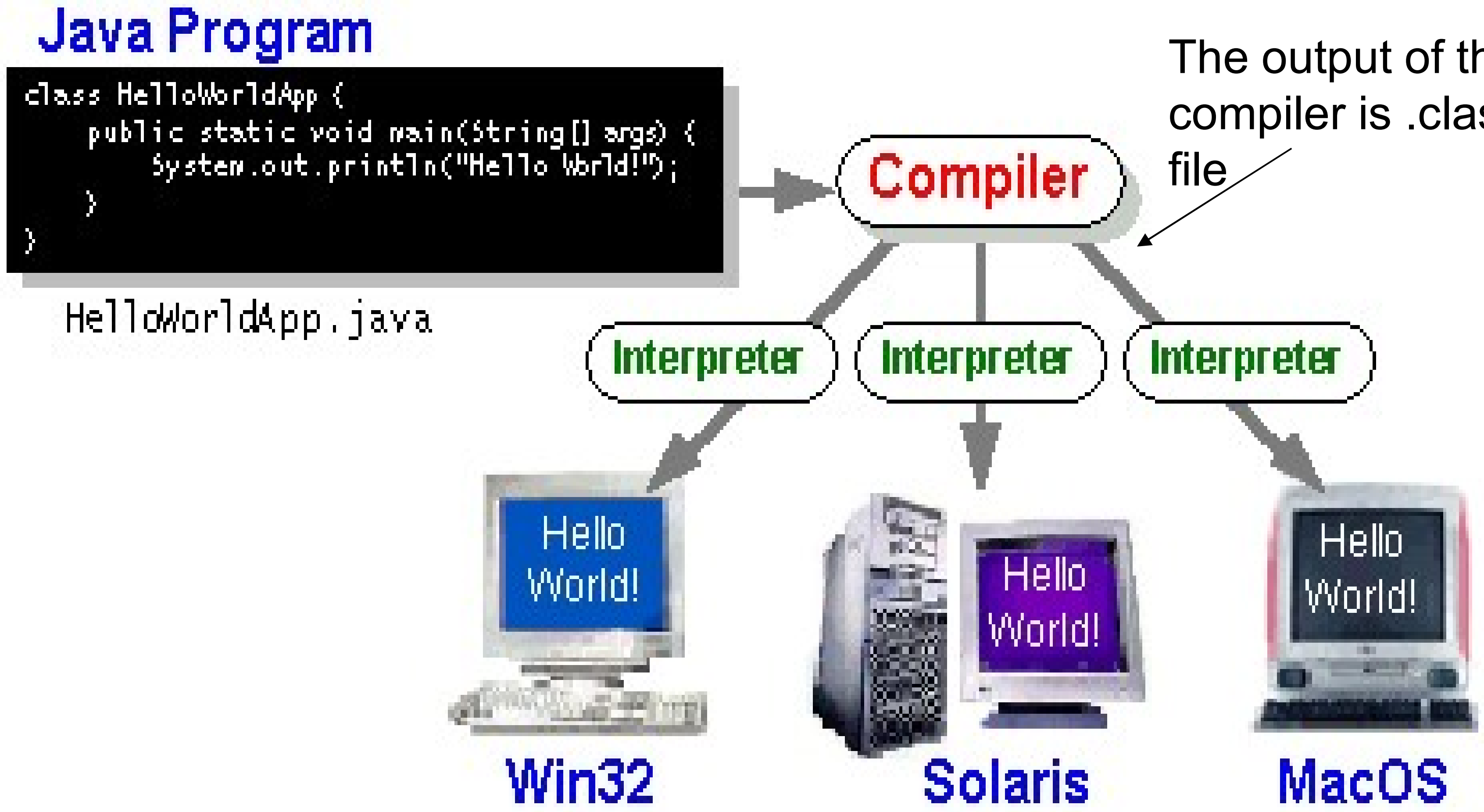


The Java Virtual Machine.

- Java source files (.java) are compiled to Java bytecode (.class)
- Bytecode is interpreted on the target platform within a Java Virtual Machine



A Picture is Worth…



```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

HelloWorldApp.java

The output of the compiler is .class file

The Interpreter's are sometimes referred to as the Java Virtual Machines

What Can Java Do?

- Three types of Java Programs
  - Applets
    - run within a Java-enabled browser
  - Standalone Applications
    - a standalone program that runs directly on the Java platform, examples
  - Server Applications
    - Servlets
    - JSP, JSF, Struts, Spring
  - Mobile Application
    - Android

Object Oriented Benefits

- Better software design
  - different way of thinking about software
  - software can more closely models real world problem being addressed
  - domain experts can more easily participate in initial software design
- Easier to maintain
  - changes are more localized
- Easier to extend
  - easy to make extensions to existing functionality
- Less code to write

Features Important to Developer

- Object-Oriented (benefits discussed earlier)
- Productive
  - fewer compiles are needed than with C/C++
  - faster development cycle since linking is not required
- Familiar
  - syntax resembles C and C++
- Garbage collection
  - avoids memory leaks (failing to free memory that will no longer be used)
  - avoids accessing freed memory
- Extensible

Aspects of C/C++ Improved in Java

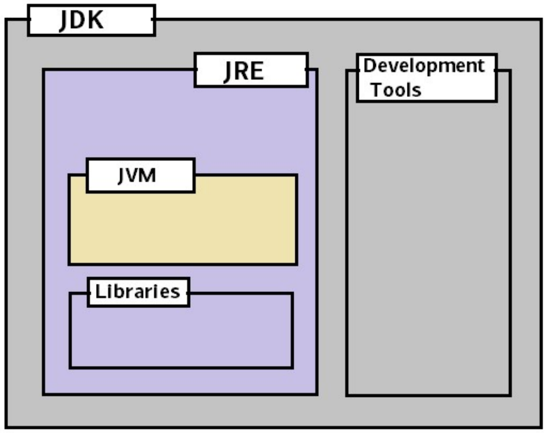
- Time consuming links
- Confusing features
  - multiple inheritance, operator overloading, templates
- Dangerous features
  - unsafe casts between unrelated types(Now controlled through generics)
  - array access w/o bounds checking
  - lack of memory management (garbage collection)
- Non-OO features

Common Terms

1. **JDK** – **Java Development Kit** (in short JDK) is Kit which provides the environment to **develop and execute(run)** the Java program. JDK is a kit(or package) which includes two things
   1. Development Tools(to provide an environment to develop your java programs)
   2. JRE (to execute your java program).

2. **RE** – **Java Runtime Environment** (to say JRE) is an installation package which provides environment to **only run(not develop)** the java program(or application)onto your machine. JRE is only used by them who only wants to run the Java Programs i.e. end users of your system.

3. **JVM** – **Java Virtual machine**(JVM) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for **executing the java program line by line** hence it is also known as interpreter.



HelloWorld.java

- Here is Java's "HelloWorld" implementation:

In the file, HelloWorld.java:

```
public class HelloWorld
{
  public static void main(String[] args)
  {
       System.out.println("Hello World");
  }
}
```

Java IDE Tools

- Eclipse
- NetBeans
- Enide Studio
- Blue J
- JDeveloper
- And so on…

Thanks

National University
Of Computer and Emerging Sciences

quest lab