Mobile App Architecture

Unit 2.4

Dr. Mahesha BR Pandit

09-Nov-21 Version 1.0

Mobile Apps: Overview

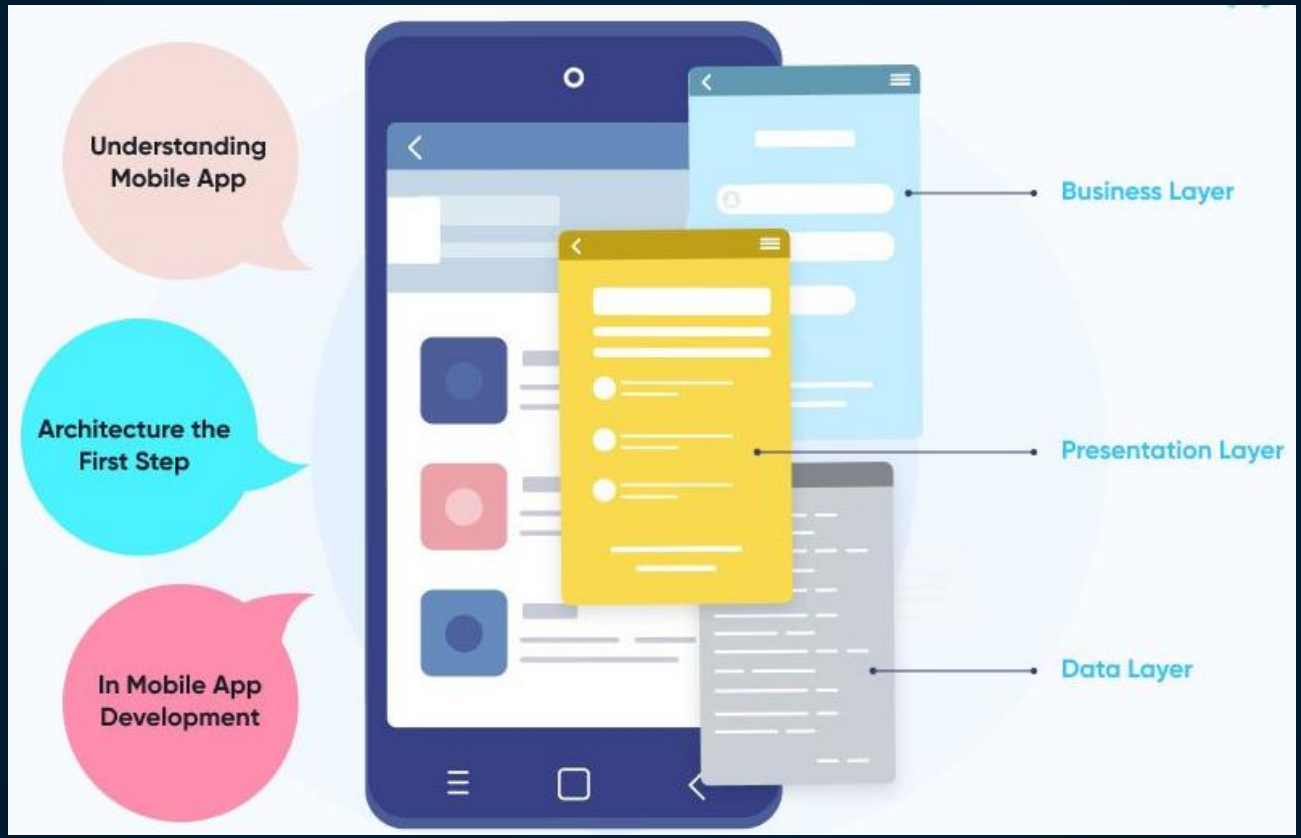
1. There are over 2.22 million apps on the Apple app store and 3.48 million apps on the Google Play store.
2. Mobile app architecture refers to a set of rules, techniques, processes, and patterns to develop a mobile application.
3. Mobile app architecture is often used interchangeably, although incorrectly, with the mobile tech stack.
4. The mobile technology stack is the set of technologies and technical frameworks that make up the front and back-end of a mobile or web app (the what of the app), but is less concerned with the business / customer requirements (the why of the app) or the development process (the how of creating the app)
5. The mobile app architecture is made up of all the parts of the app – all the questions about why, what, how – including what data is collected, how the data moves, what the app looks like, for what platform, using what tech stack

Mobile Apps: Considerations

Mahesha
BR Pandit

1. Choose the platform (iOS, iPadOS, Android, Windows, Cross-platform)
2. Consider the different models of smartphones! (Screen size and DPI, Screen resolution, CPU, RAM)
3. Consider Development frameworks – Front-end: Bootstrap, Foundation, React, Angular, Vue, and Backbone. Back-end: Ruby on Rails, Flask, Django, Laravel, Swift, Xamarin, React Native and Flutter
4. Consider Bandwidth
5. Pay attention to UI and UX
6. Pay attention to navigation – Hamburger menu (three line menu), Search, Bars, rails, drawers, or tabs, Icons, Labels, Gestures, Scrolling, thumb zone navigation
7. Push notifications (90% opening rate, 88% engagement, 70% acceptance, 40% work on it within 1 hour, Android Automatic, iOS opt-in)

Typical 3 Layered Architecture

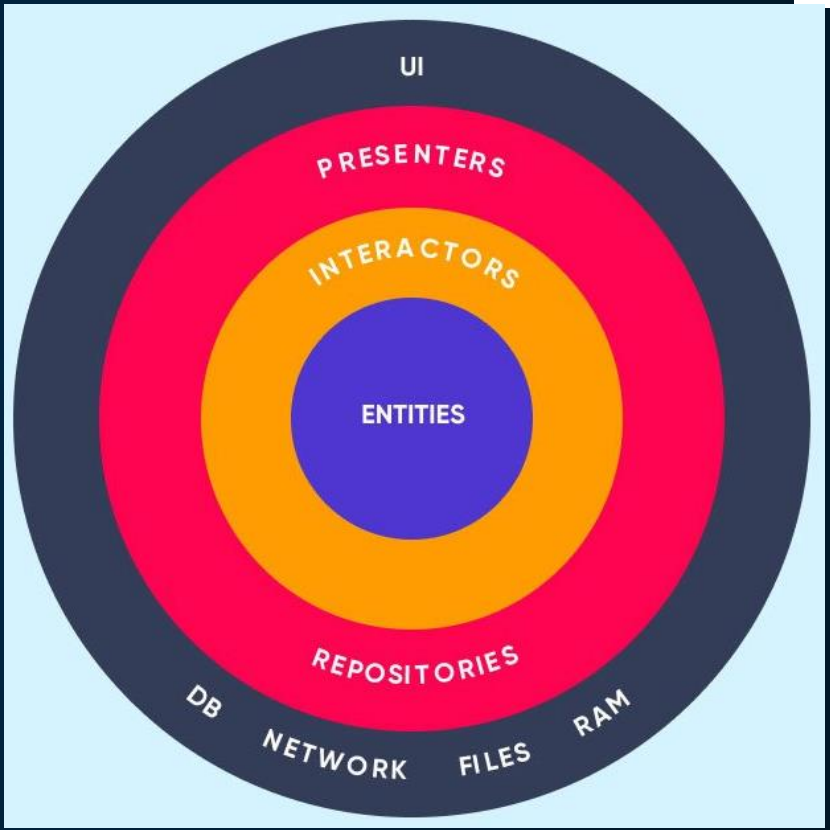


Types of Mobile App Architectures

1. Native Apps	App specifically developed for mobile
2. Web Apps	Web app adapting to mobile device
3. Hybrid Apps	Combination of Native and Web Apps
4. Cross-Platform	Single codebase used across mobile platforms

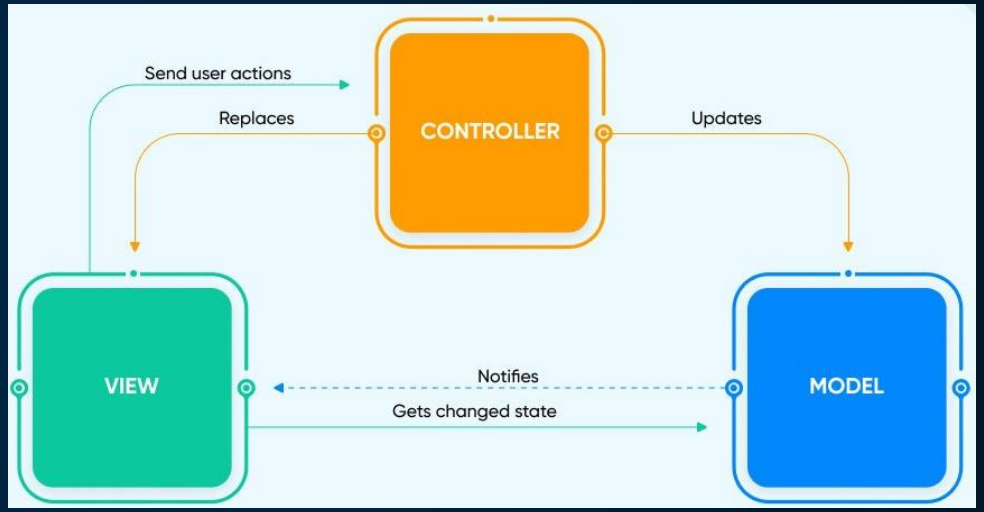
Android Mobile Application Architecture

- Support Android languages (Kotlin and Java)
- No single architecture
- Clean Architecture
- In Clean, the architecture is built on the principles of layers and inversion of control
- Sort of onion architecture



iOS Mobile Application Architecture

- Objective-C and Swift
- MVC model (Model-View-Controller)
- Model – The data layer (persistence, model objects, parsers, managers, networking code).
- View – Similar to the presentation layer, a re-usable layer that represents the app to the user.
- Controller – A mediator level that communicates with an abstraction via a protocol

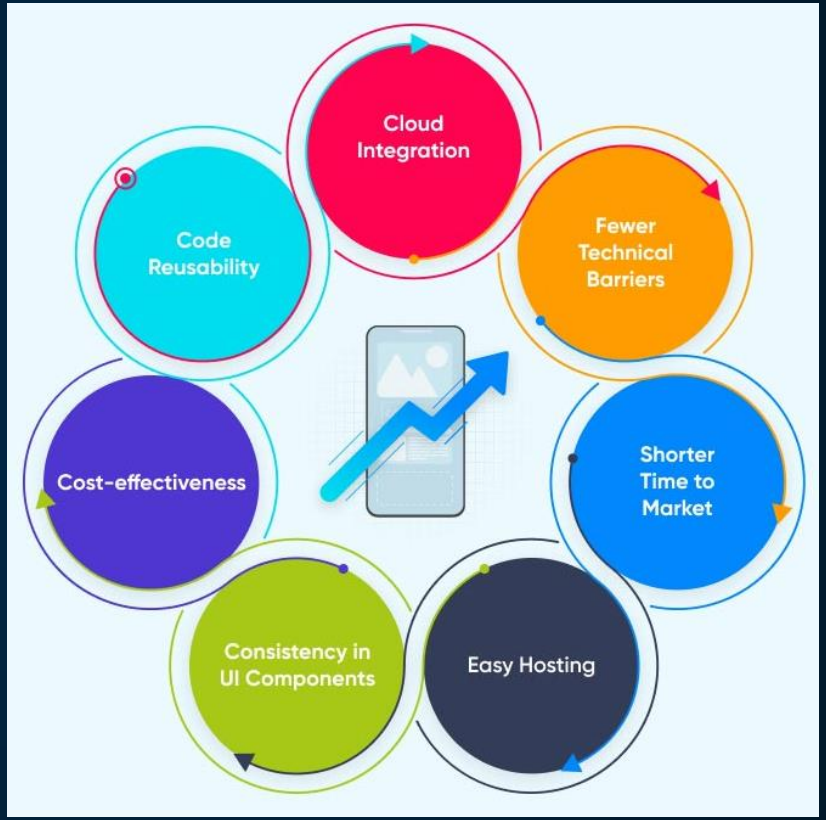


Hybrid Mobile Application Architecture

- Both native and web solutions
- Hybrid apps use native apps as “shells” for the back-end
- JavaScript, HTML and CSS for the front-end.
- Plugins such as Apache Cordova or Ionic Capacitor to access native platform features

Cross Platform Mobile Application Architecture

- Common Code Base
- Works across Android and iOS
- Rely on frameworks, rather than a language
- React Native, Flutter, and Xamarin



How to choose the architecture?

- Budget
- Intended users
- Key Features
- Platform choice
- Development time

Embedded Systems Architecture

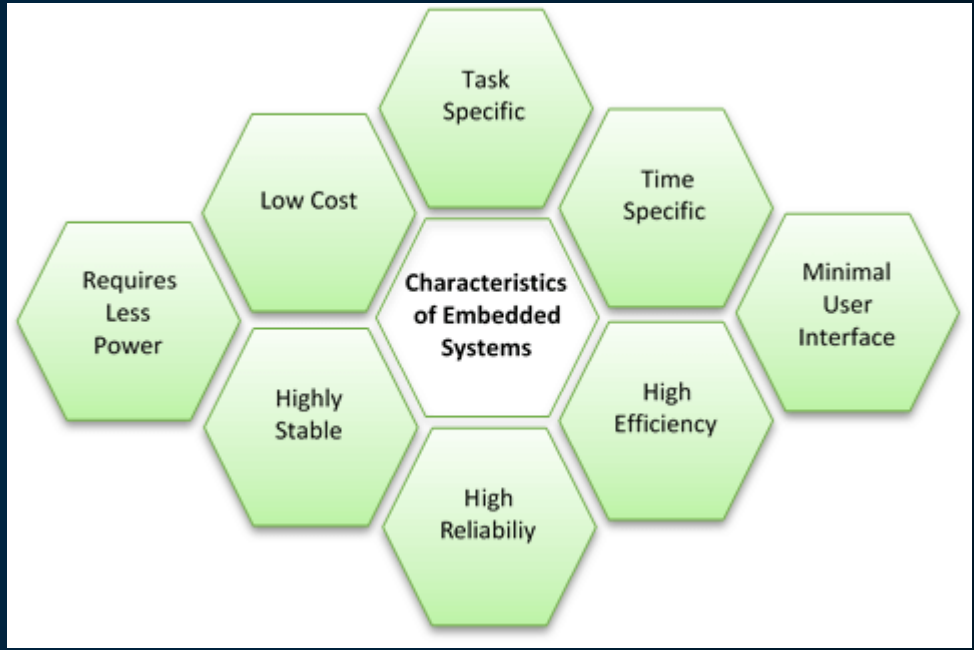
Unit 2.5

Dr. Mahesha BR Pandit

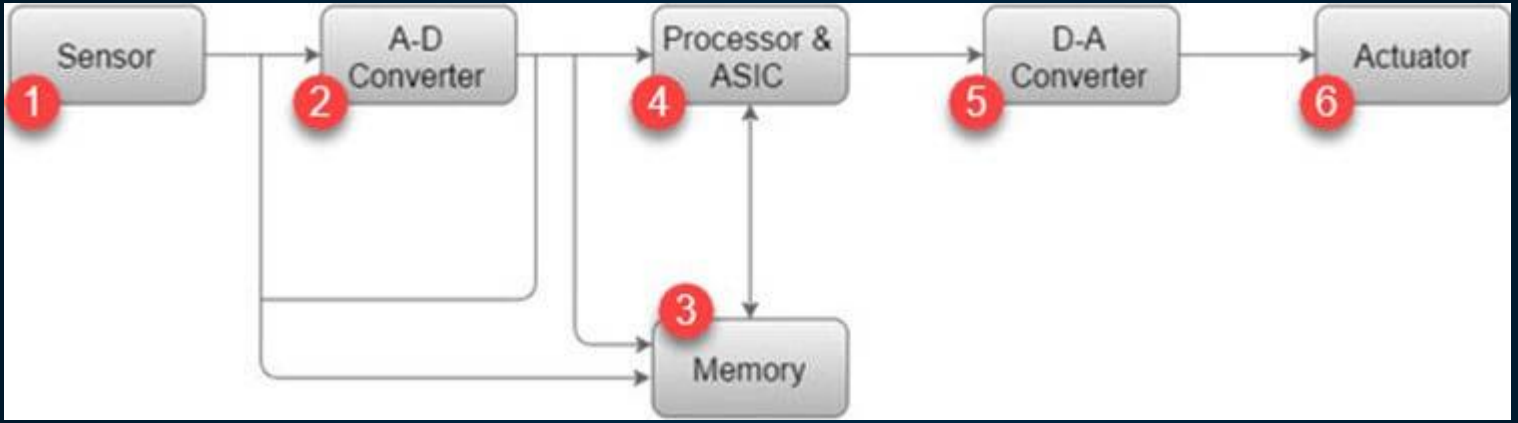
12-Nov-21 Version 1.0

Embedded Systems: Overview

- Combination of computer software and hardware
- Examples: Fire Alarms, Laser Printer, Car, ATM, Drones, Mobiles, TVs
- Real time performance with high availability and reliability.
- Designed for one specific task
- Connected with peripherals
- Needed minimal user interface
- Limited memory, low cost, fewer power consumption

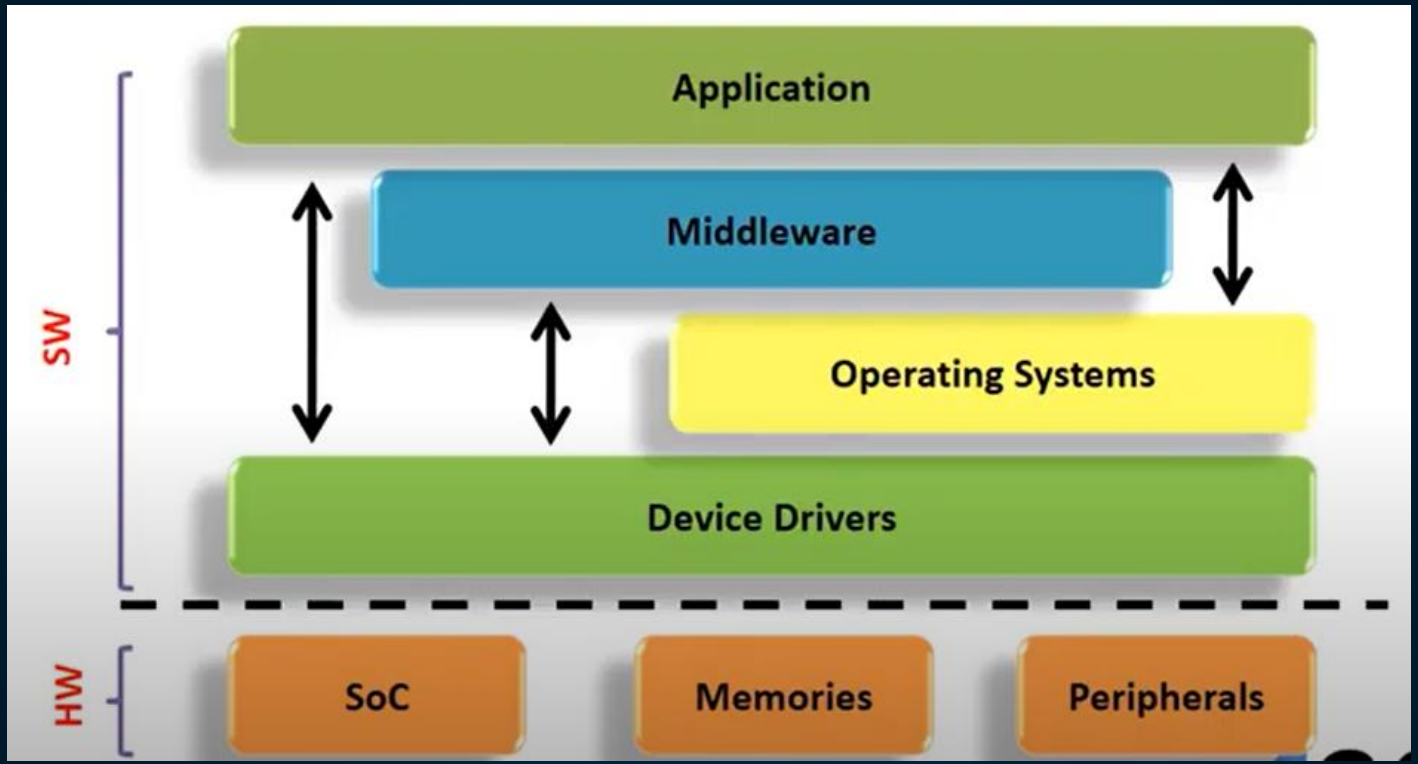


Embedded Systems: Architecture



1. Sensor: Measures the physical quantity and converts it to an electrical signal
2. A-D converter converts an analog signal sent by the sensor into a digital signal.
3. Memory is used to store information (Volatile or Non-Volatile)
4. Processor & ASICs (Application Specific IC): Process the data
5. D-A converter converts the digital data fed by the processor to analog data.
6. An actuator produces the necessary output

Embedded Systems: Architecture 2



SoC = System on a Chip: Example: A smartwatch SoC, includes a primary CPU, graphics processor, DAC, ADC, flash memory, and voltage regulator. These components fit on a single chip roughly the size of a 5 Rs coin

Applications

1. Robotic science: Ground Vehicles, Drones, Underwater Vehicles, Industrial Robots
2. Medical: Dialysis Machine, Infusion Pumps, Cardiac Monitor, Prosthetic Device
3. Automotive: Engine Control, Ignition System, Brake System
4. Networking: Router, Hubs, Gateways, Electronics Instruments
5. Home Devices: TVs, Digital Alarm, Air Conditioner, DVD Video Player, Cameras
6. Automobiles: Fuel Injection, Lighting System, Door Locks, Air Bags, Windows, Parking Assistant System, Anti-stealing Alarms
7. Industrial Control: Robotics, Control System, Missiles, Nuclear Reactors, Space Stations, Shuttles

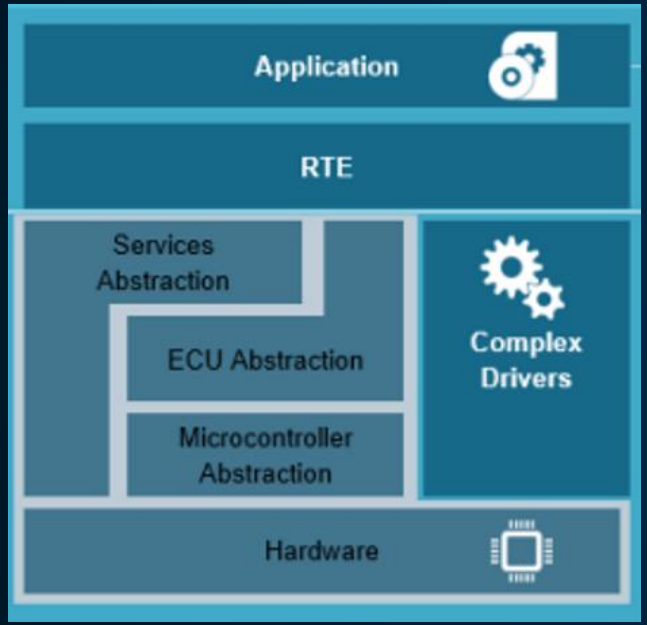
Advantages and Disadvantages

- Wide usage
- Good error control
- Simplifies hardware
- Reduces costs overall.
- Enhanced performance
- Mass production.
- Highly reliable.
- Few interconnections.
- Small in size.
- Low power consumption

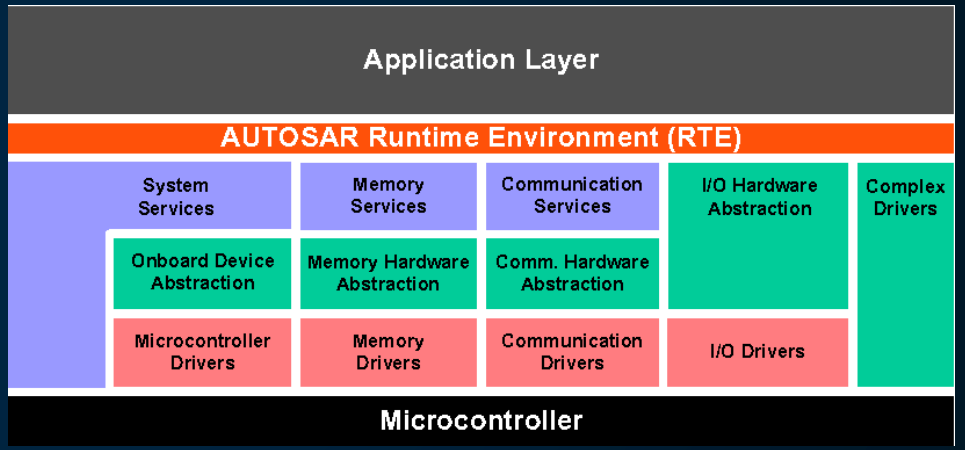
- High development effort.
- Long time to market.
- Embedded systems do a very specific task, so it can't be programmed to do different things.
- Very limited resources for memory.
- It is difficult to backup of embedded files.

Architectures

Siemens

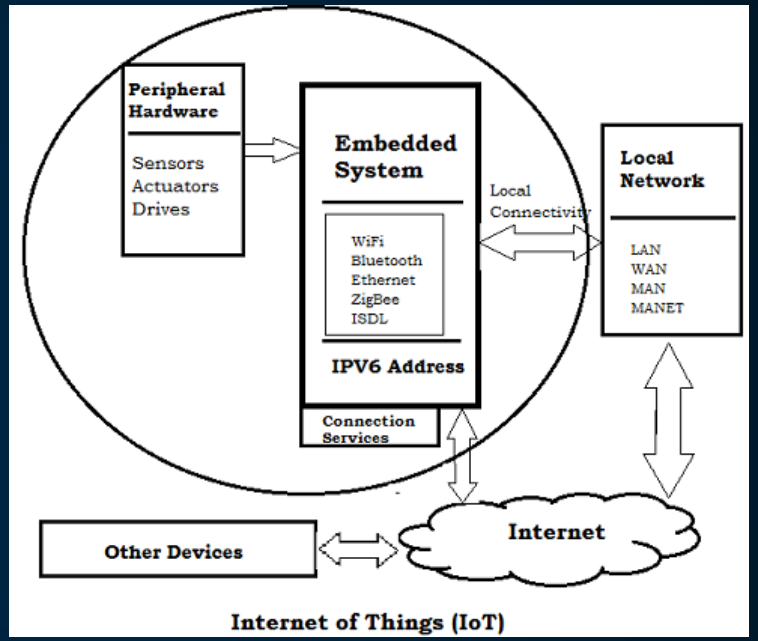


AUTomotive Open System ARchitecture



Embedded Systems versus IoT

The Internet of Things (IoT) is defined as a process in which objects are equipped with sensors, actuators, and processors that involve hardware board design and development, software systems, web APIs, and protocols, which together create a connected environment of embedded systems.



- Embedded System is a subset of IoT
- IoT is made up of “Things” and those things = Embedded Systems

AI and ML Systems Architecture

Unit 2.6

Dr. Mahesha BR Pandit

13-Nov-21 Version 1.0

AI and ML Systems: Overview

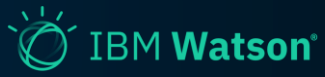
- AI projects are iterative
- AI projects have multiple-stages
- AI projects are data-driven processes
- AI projects require specialized knowledge, skills
- Results of AI projects are only as good as the input data
- AI model development is dependent upon data
- Data should be in the format expected by the deep learning framework.
- It is also iterative; repeatedly looping through data sets and tunings to develop accurate models, then comparing new data in the model to the original business or technical requirements to refine the approach.

Framework – Model and Platform

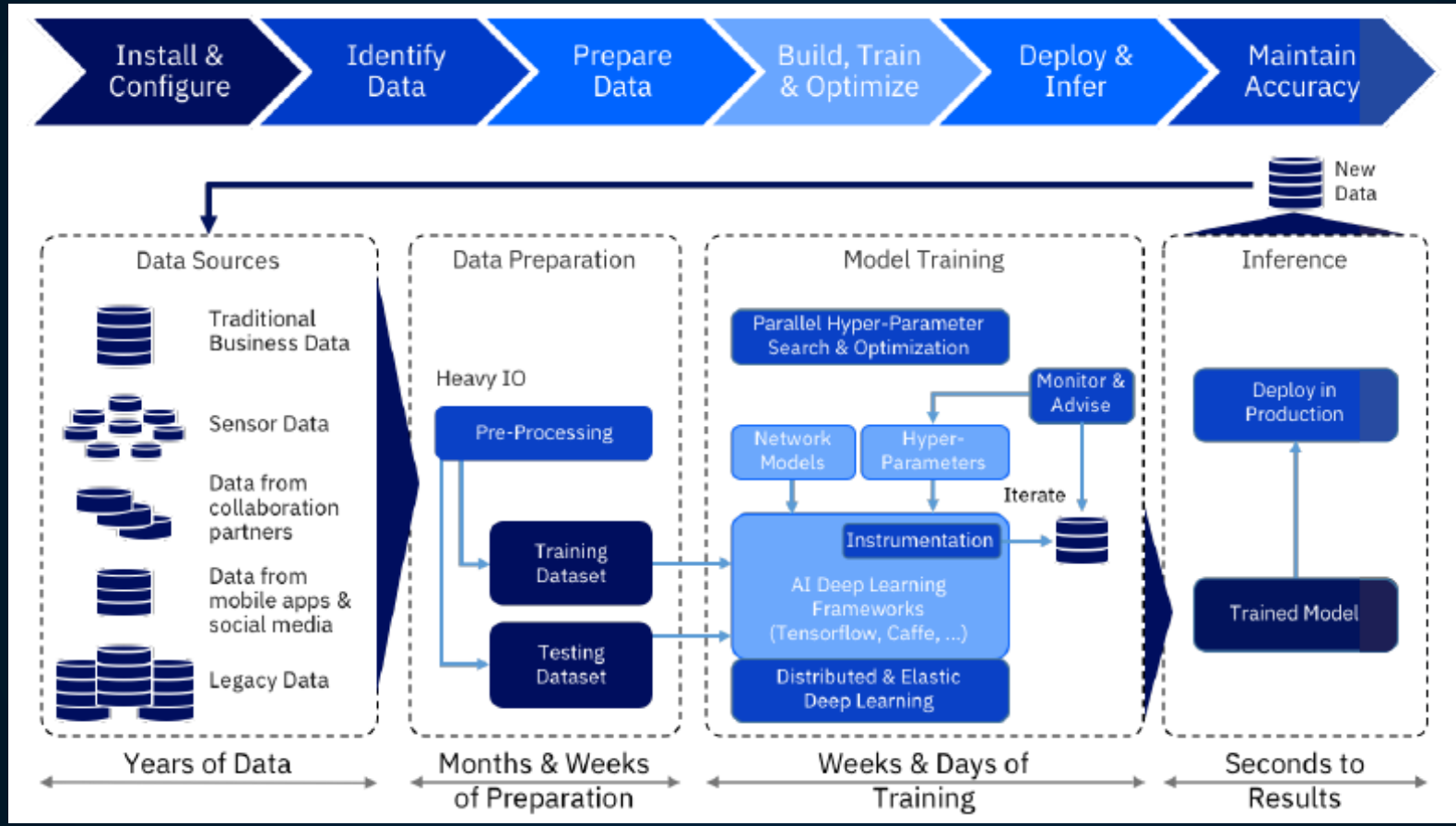
- AI frameworks provide the building blocks for data scientists and developers to design, train and validate AI models through a high-level programming interface and without getting into the details of the of the underlying algorithms.
- AI models are the work products produced by the AI frameworks, these are the objects that get trained using data and then used to analyze new data.
- AI platform consists of AI frameworks, the software used to build a multitenant data science environment, the distributed computing, training and inference environment, and a tiered data management environment

Google Cloud AI

Microsoft Azure AI



AI Workflow


Install & Configure
Identify Data
Prepare Data
Build, Train & Optimize
Deploy & Infer
Maintain Accuracy
New Data
Data Sources
Traditional Business Data
Sensor Data
Data from collaboration partners
Data from mobile apps & social media
Legacy Data
Data Preparation
Heavy IO
Pre-Processing
Training Dataset
Testing Dataset
Model Training
Parallel Hyper-Parameter Search & Optimization
Monitor & Advise
Network Models
Hyper-Parameters
Iterate
Instrumentation
AI Deep Learning Frameworks (Tensorflow, Caffe, ...)
Distributed & Elastic Deep Learning
Inference
Deploy in Production
Trained Model
Years of Data
Months & Weeks of Preparation
Weeks & Days of Training
Seconds to Results

Confusion

- Causation
- Correlation

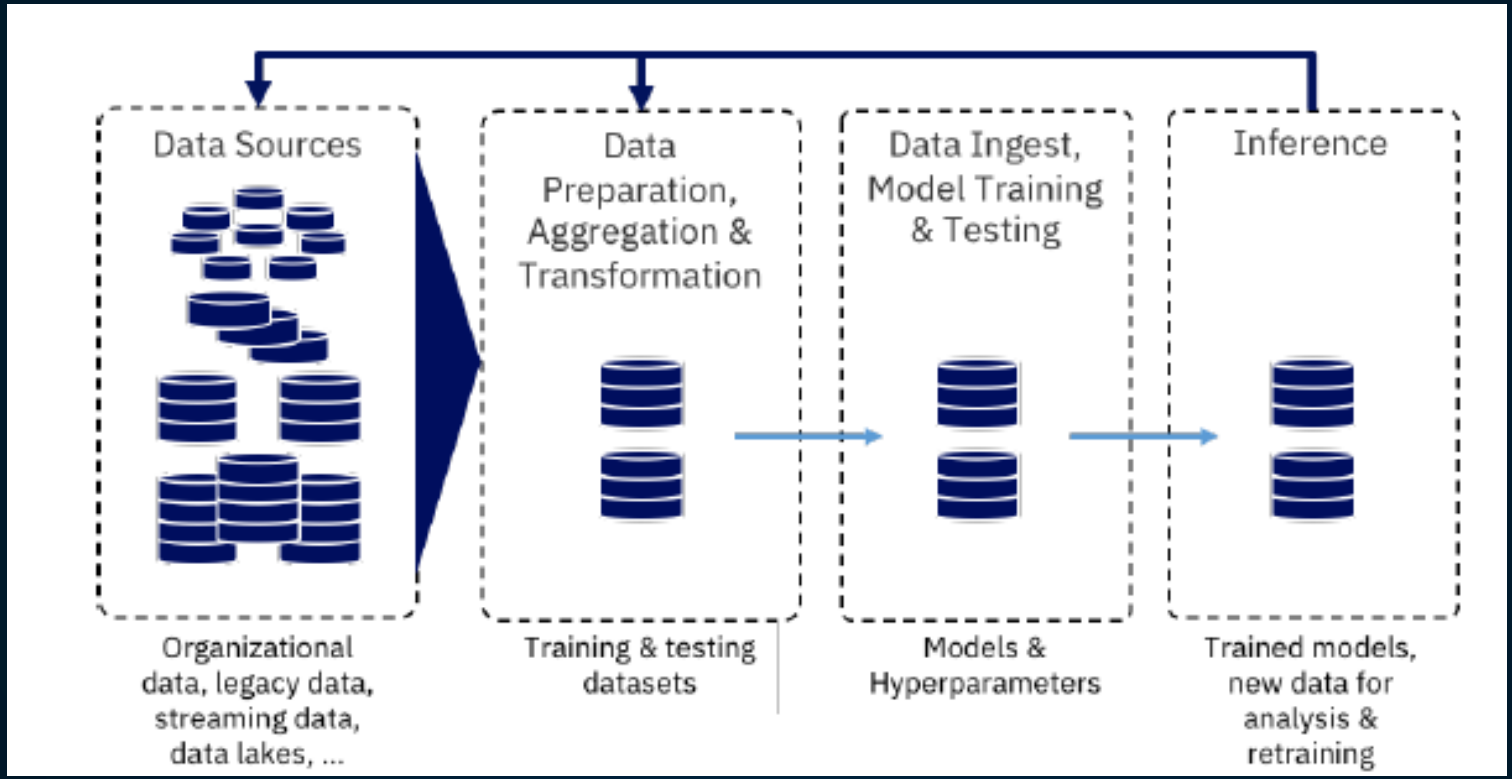
AI Architecture – Description – 1

- AI workflow is a process cycle: Understand the problem – find the data needed to build a model to solve the problem – build, train, retrain
- *Champion challenger* approach – new models challenge existing ones – model with the best results becomes the new champion
- Quality of training data influences the outcome of the AI model
- Data is separated into a few broad sets – training, validation, testing, new data
- Many sources exist for data – traditional ERP systems, databases, data lakes, sensors, collaborators and partners, public data, mobile apps, social media, and legacy data
- Data may be structured and unstructured in many formats such as file, block, object and Hadoop Distributed File Systems (HDFS)
- Many AI projects begin as a big data problem. Regardless of the source, a large volume of data is needed, and it inevitably needs preparation, transformation and manipulation.
- Preparing the data is often one of the largest organizational challenges - highly manual and serial set of steps: identifying and connecting to data sources, extracting to a staging server, tagging the data, using tools and scripts to manipulate the data (e.g., removing extraneous elements, breaking large images down to 'tile' size so they will fit in GPU memory, etc.)

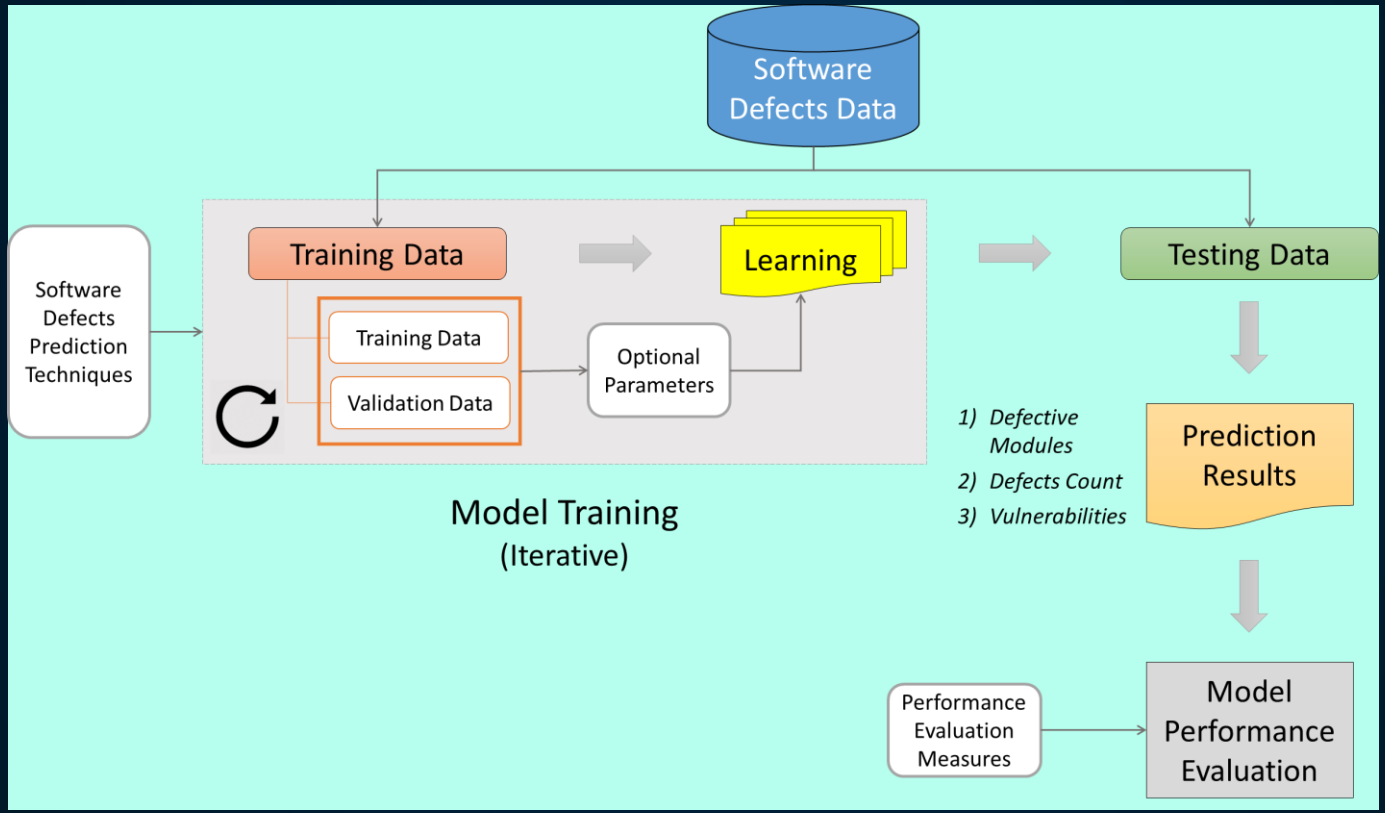
AI Architecture – Description – 2

- Modeling is both art and science
- Compute heavy
- Parameters define model's characteristics (number of leaves and depth of a tree, the number of hidden layers, and other factors)
- Parameters to be chosen before training
- Trial and error
- Open-source frameworks are rigid and fragile
- AI models are trained initially on historical data and are then tested and deployed and run as an inference service and used to analyse new and real time data
- Completing the AI cycle, the newly analyzed data is fed back into the process to retrain and improve the model or used to build new models

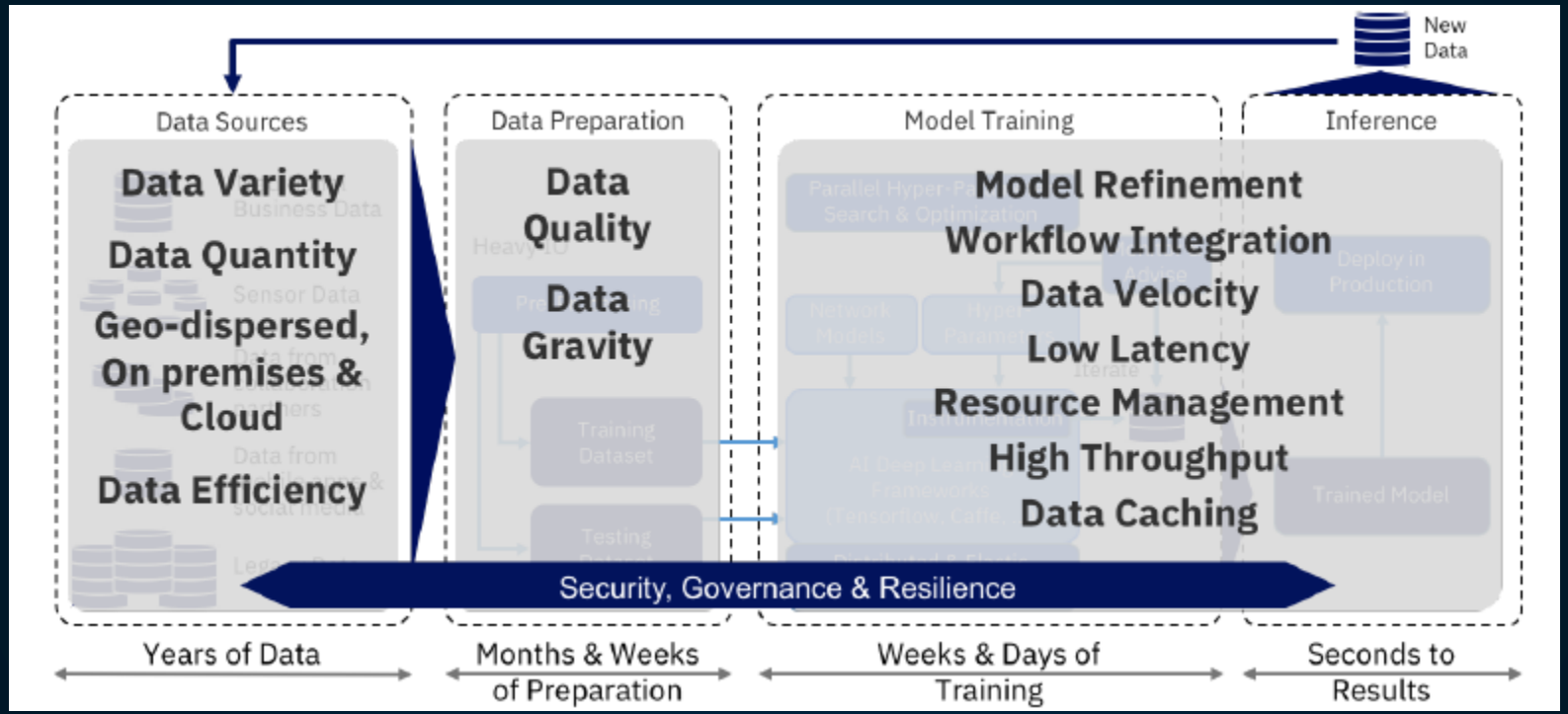
Data Flow


Data Sources
Data Preparation, Aggregation & Transformation
Data Ingest, Model Training & Testing
Inference
Organizational data, legacy data, streaming data, data lakes, ...
Training & testing datasets
Models & Hyperparameters
Trained models, new data for analysis & retraining

AI Model for Defect Prediction



Data Pipeline



Technology Stacks

Unit 3.1

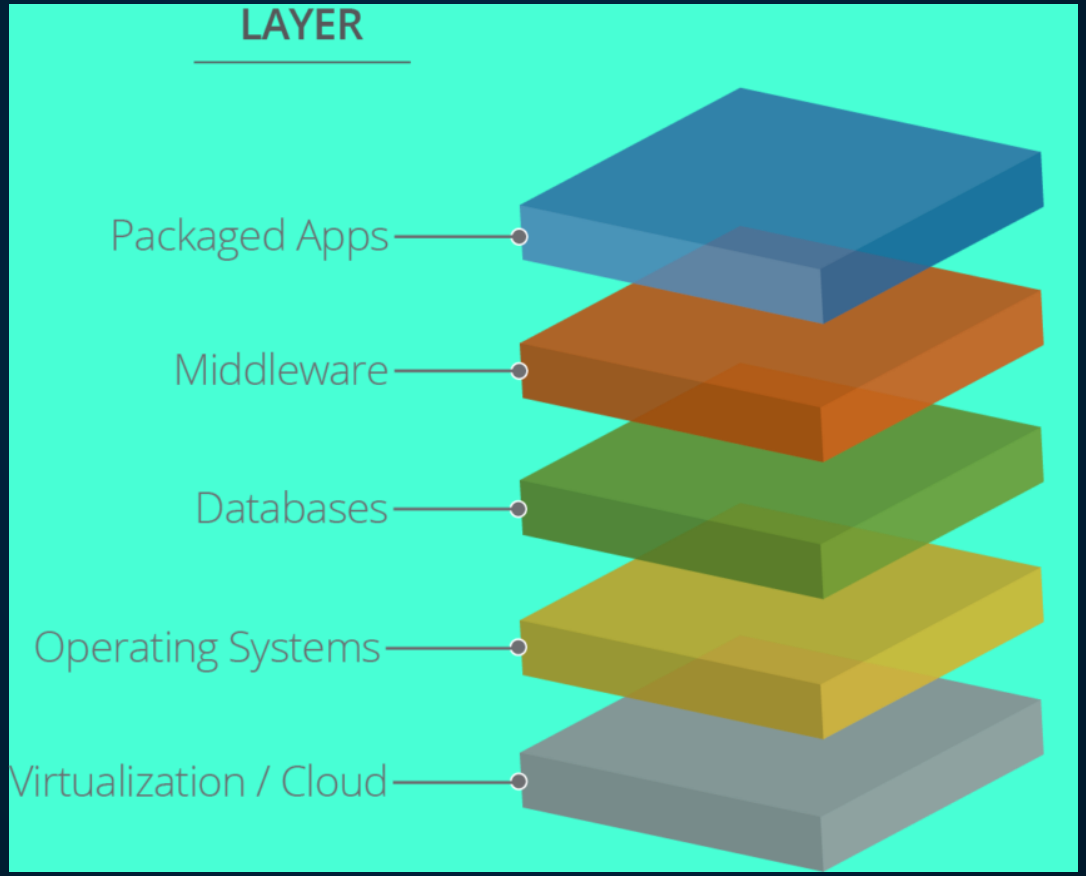
Dr. Mahesha BR Pandit

23-Nov-21 Version 1.0

Technology Stacks

- A technology stack is simply a collection of different technologies that work together for some purpose
- A foundation for systems such as web applications, SaaS, mobile apps, and standalone desktop software
- E.g., LAMP – Linux, Apache, MySQL and PHP
- There are several types of technology stacks – Front End, Back End (or Server Side), Full Stack, Business Stacks
- Hardware can be a part of the technology stack
- Network infrastructure can also be a part of the technology stack
- Aka “Solution Stack”

Technology Stack



Technology Stacks: How to Select?

- Speed
- Throughput
- Scalability

Technology Stacks: Popular Choices

- HTML, CSS, JavaScript
- LAMP
- ASP.NET: ASP.NET MVC (Standard model-view-controller framework), IIS (Microsoft's web server), Angular frontend framework with TypeScript, SQL Server – Microsoft's enterprise database, Microsoft Azure – A favorite of most .NET developers
- MEAN: MongoDB (NoSQL database), Express.js backend web framework, Angular.js frontend framework, Node.js server side JavaScript
- MERN: Replace Angular with React.js in MEAN
- MEVN: Replace Angular with Vue.js in MEAN
- Ruby on Rails
- Java: Java, Spring, Wildfly, Linux, NGINX
- Python : Python, Django, Flask, any RDBMS or NoSQL, Any Server

Technology Stacks: Examples

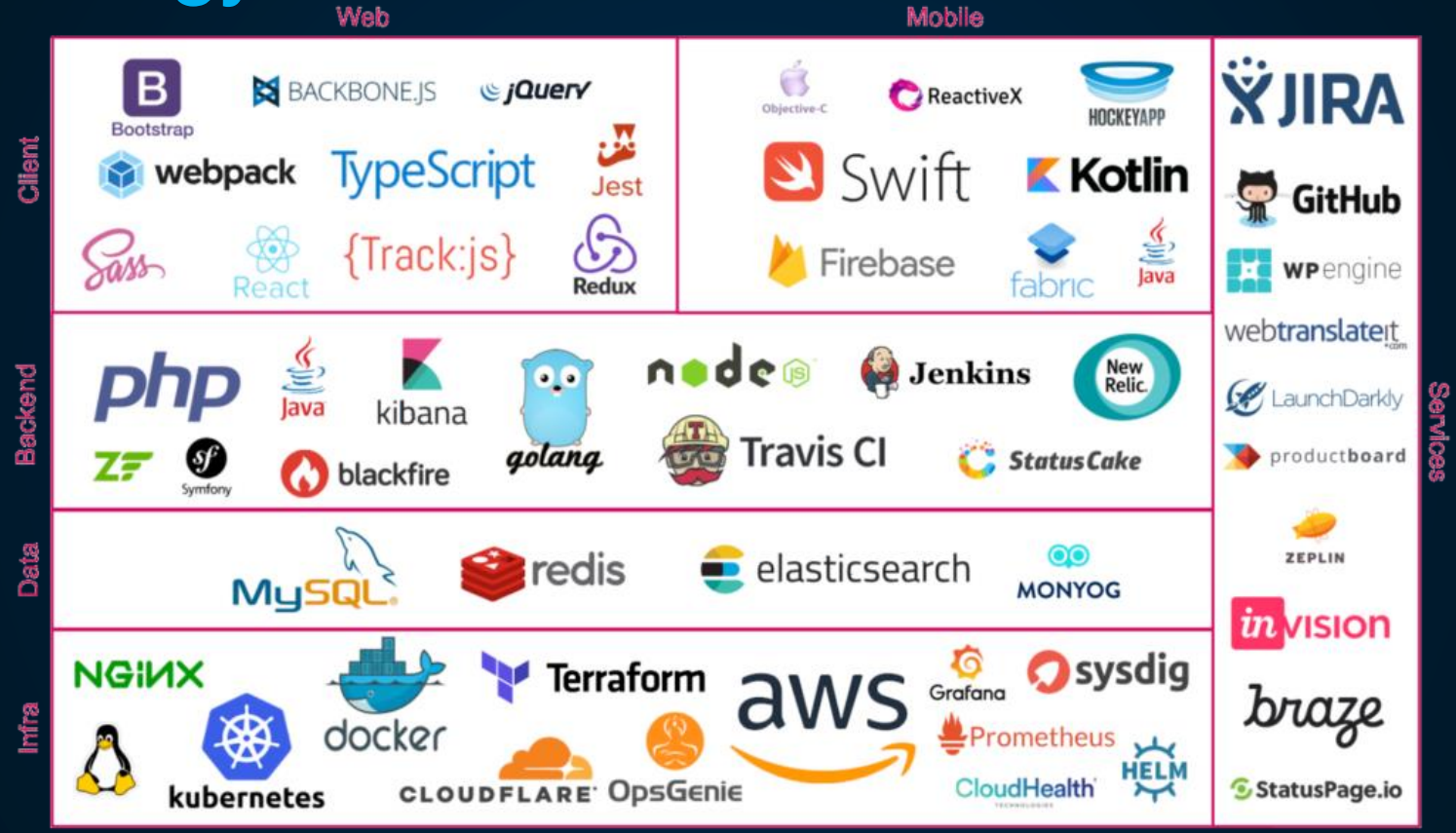
	Airbnb	Facebook	Pinterest	Uber
Programming Languages	JavaScript, Ruby	PHP, GraphQL, Hack	Python, Java, Go	Python, Java, Go, Objective-C
Framework	Rails	Tornado	Django, Javascript MVC	Node.js, Apache Thrift
Databases	MySQL, Amazon RDS, Hadoop	Cassandra, RocksDB, Beringei, Memcached	MySQL, Hadoop, HBase, Memcached, Redis	MySQL. PostgreSQL, MongoDB, Redis
Server	NGINX	Custom	NGINX	NGINX

Technology Stacks: Examples

- Airbnb Tech Stack: Amazon CloudFront, Google Analytics, Braintree, Twilio, MixPanel
- Uber Tech Stack: Zendesk, PayPal, Twilio, Optimizely, MixPanel
- Netflix Tech Stack: Oracle, Amazon SES, Airship, Falcor, GitHub
- LinkedIn Tech Stack: Oracle, Google Analytics, Adobe Experience Manager, Okta, Unbounce
- Twitter Tech Stack: Oracle Dyn, Fastly, HackerOne, UserTesting, Campaign Monitor
- Pinterest Tech Stack: Amazon CloudFront, BitBar, Zendesk, SparkPost, Amazon Route 53
- Facebook Tech Stack: BitBar, Campaign Monitor, Confluence, Framer, Stetho
- Google Tech Stack: App Annie, Bazel, Android Studio, Kubernetes, EarlGrey
- Spotify Tech Stack: Amazon CloudFront, Google Analytics, Optimizely, Twilio SendGrid, Lookback

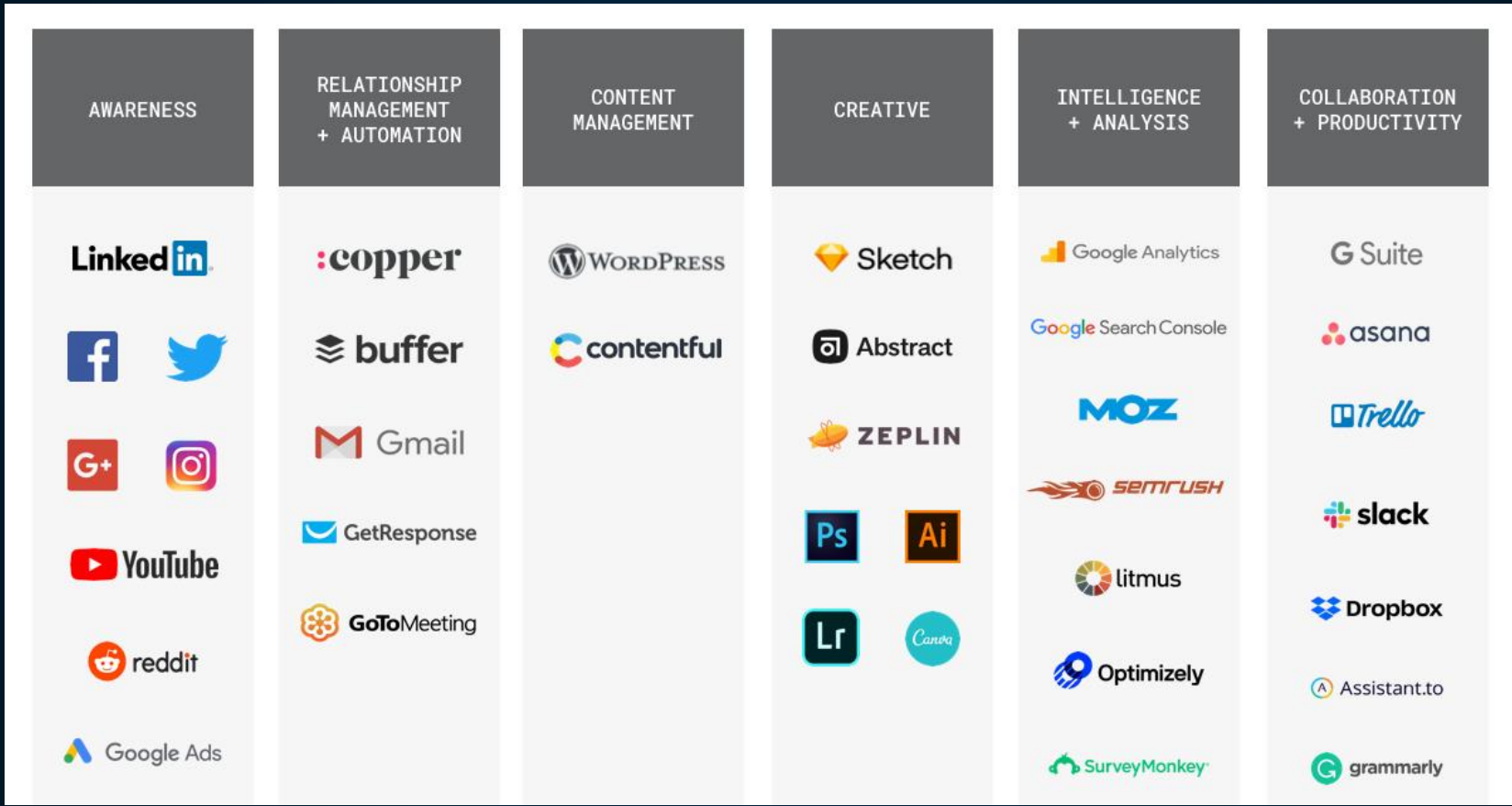
https://blog.hubspot.com/customers/auditing-your-companys-tech-stack-or-platform-apps

Technology Stacks



https://tech.foodora.com/tech-stack/

Technology Stacks: Creative Agencies

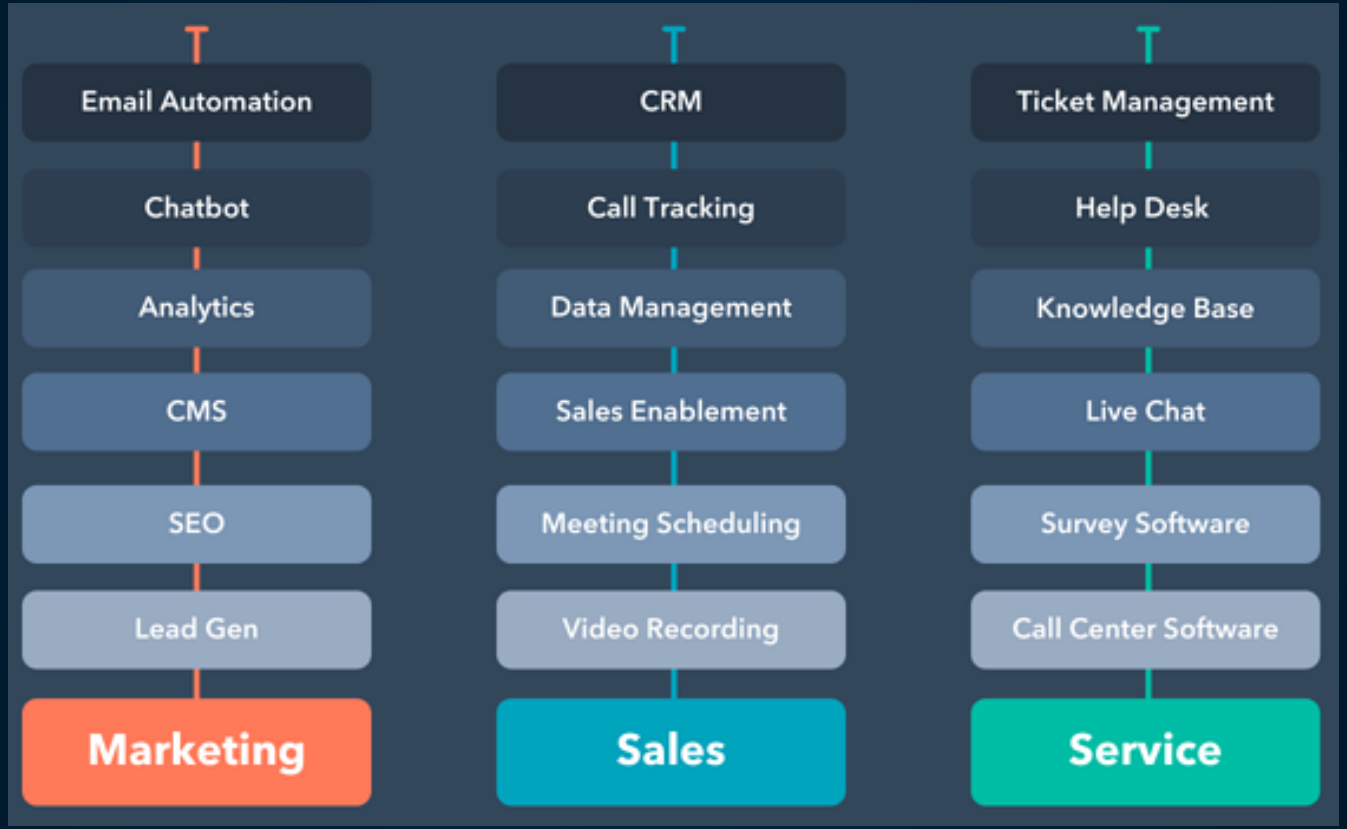


Technology Stacks: Android

https://android.stackexchange.com/questions/218078/how-much-privileged-gms-core-is-in-the-android-stack

Technology Stacks: Business

Technology Stacks: Marketing, Sales, Service



https://blog.hubspot.com/customers/auditing-your-companys-tech-stack-or-platform-apps

Technology Stacks: How to Track?

- Look at this example: https://chiefmartec.com/2018/04/marketing-technology-landscape-supergraphic-2018/

https://blog.hubspot.com/customers/auditing-your-companys-tech-stack-or-platform-apps

Assignments

- 3A: Describe the technology stack of LinkedIn, Twitter and NetFlix
- 3B: Describe the technology stack of iOS (refer slide 10 for Android)
- 3C: Describe the difference between *sales* and *marketing*

https://blog.hubspot.com/customers/auditing-your-companys-tech-stack-or-platform-apps

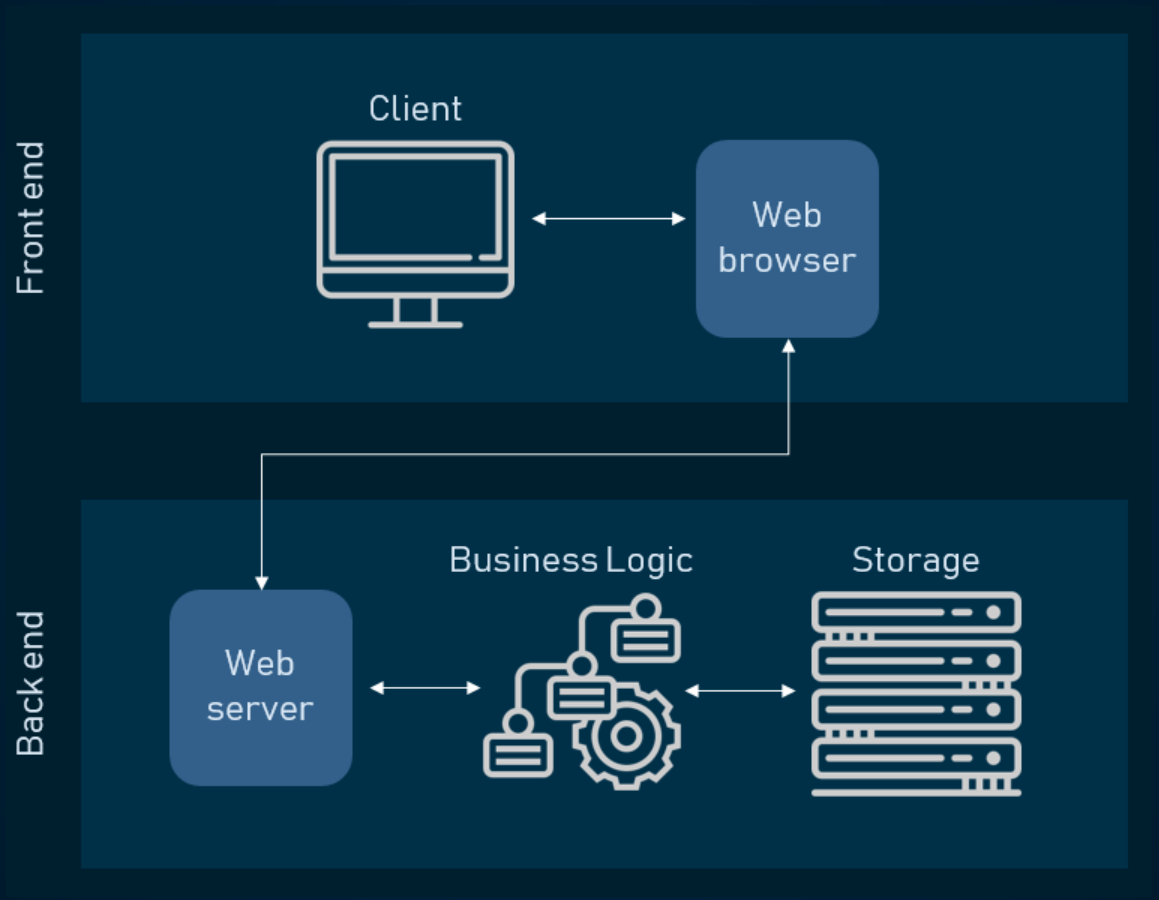
Frontend Technologies

Unit 3.2

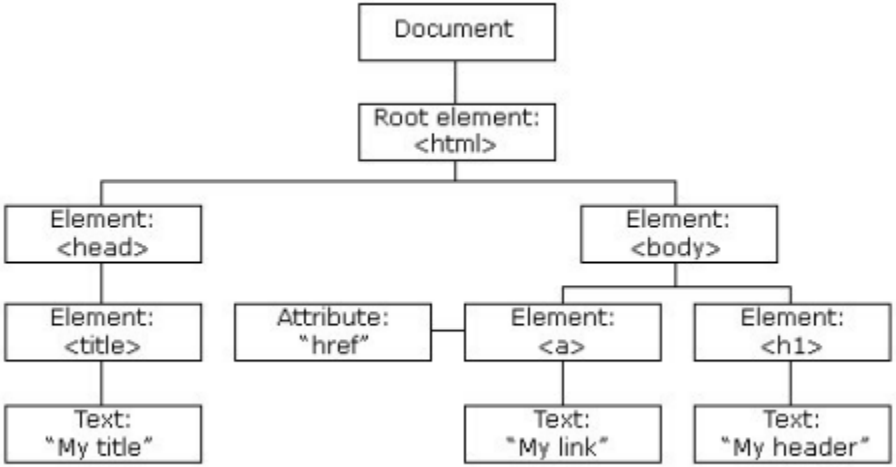
Dr. Mahesha BR Pandit

26-Nov-21 Version 1.0

Frontend


Front end
Client
Web browser
Back end
Business Logic
Storage
Web server

The DOM


Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
"href"
Element:
<a>
Element:
<h1>
Text:
"My title"
Text:
"My link"
Text:
"My header"

Frontend Technologies

- Minimum: HTML, CSS, DOM, JavaScript, Ajax
- Full set: Core, Text Editors, UI Frameworks, Concepts, Templating, Browser Refreshing, CSS Preprocessors, OOCSS & Style Guides, Version Control, Package Managers, Front-End Performance checkers, JS Frameworks, JS Preprocessors, Process Automation, Code Quality, Build Tools and Testing

Frontend Technologies: Full Set

- Concepts: Responsive Web Design, Progressive Enhancement, Accessibility
- Core: HTML (HTML5), CSS (CSS3), JavaScript (ES6), Web Components
- Text Editors: Atom, Sublime Text, WebStorm
- UI Frameworks: Bootstrap, Ionic, Foundation
- Templating: Handlebars, Haml, Jade
- Browser Refreshing: LiveReload, Guard
- CSS Preprocessors: Sass, Less
- OOCSS & Style Guides: MVCSS, SMACSS, BEM, Inuitcss, KSS, Pattern Lab
- Version Control: Git, Subversion, Mercurial
- Package Mangers: npm, Bower, CodeKit
- Front-End Performance: WebPagetest, mod_pagespeed, PerfBudget, CriticalCSS, Picturefill
- JS Frameworks: jQuery, Backbone, Ember, Angular, React, Vue, Polymer, D3
- JS Preprocessors: CoffeeScript, TypeScript, Babel
- Process Automation: Grunt, Gulp, Broccoli
- Code Quality: JSCS, ESLint
- Build Tools: RequireJS, Browserify, Webpack
- Testing: Jasmine, Mocha, Protractor, Karma

Frontend Technology wheel

Mahesha
BR Pandit

Popular CSS Frameworks

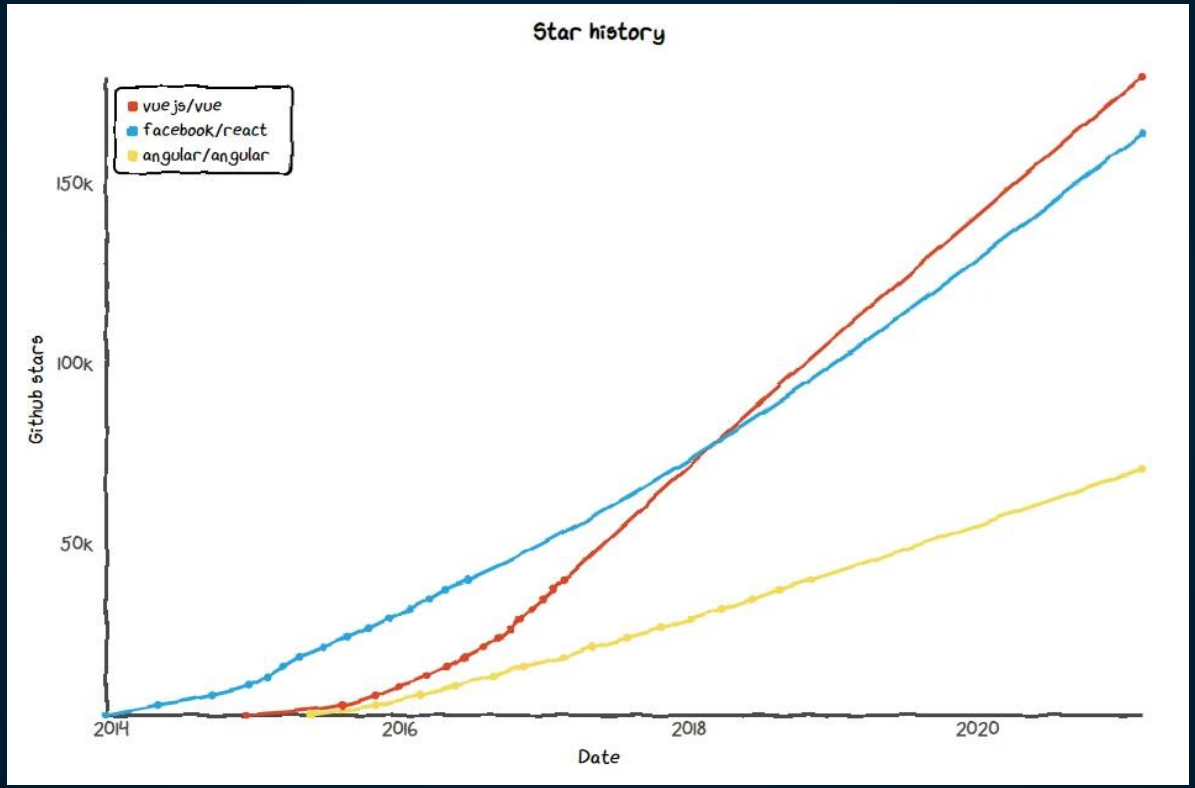
	Full-featured					Material-based		Very Lightweight
	Bootstrap	Foundation	Semantic UI	UIkit	Bulma	Materialize	Material Design Lite	Pure
Principles	RWD, Mobile first	RWD, Mobile first Semantic	Semantic Tag, Ambivalence, Responsive	RWD, Mobile first	RWD, Mobile first, Modern free	Responsive, Web design, UX-focused	Cross-device use	SMACSS, Minimalism
Size(zip folder)	592 KB	233 KB	1.8 MB	347 KB	113 KB	162 KB	205 KB	72 KB
Preprocessors	Sass	Sass	Less	Less, Sass	Sass	Sass	Sass	-
Unique Features	Jumbotron, Card component, Responsive navbar	Icon Bar, Clearing Lightbox, Flex video, Keystrokes, Joyride, Pricing tables	Divider, Flag, Rail, Reveal, Step, Advertisement, Card, Feed, Item, Statistic, Dimmer, Rating,	Article, Comments, Placeholder Flex, Cover, HTML editor	-	ScrollFire, ScrollSpy, Wave behaviors, Slide-out drawer menu, Toast, Parallax	Tooltips, Spinners	-
Flexbox	+	- (some key components can be converted)	+	+ (The Flex component)	+	-	+	-
Grid	up to 12 columns	XY 12-column grid: float grid empowered by Flexbox features	default theme: 16 columns	components: grid, flex and width; offers a border between grid columns	Simple building of columns layout	a standard 12-column fluid responsive grid system	12 -desktop; 8 – tablet; 4 - phone	a 5ths and 24ths unit-based grid
Documentation	Excellent	Good	Excellent	Good	Good	Good	Excellent	Good
JS skills required	+ -	+	+	-	-	+	-	-
Learning curve	Mild	Steep	Moderate	Moderate	Mild	Mild	Mild	Mild

Popular JS Frameworks

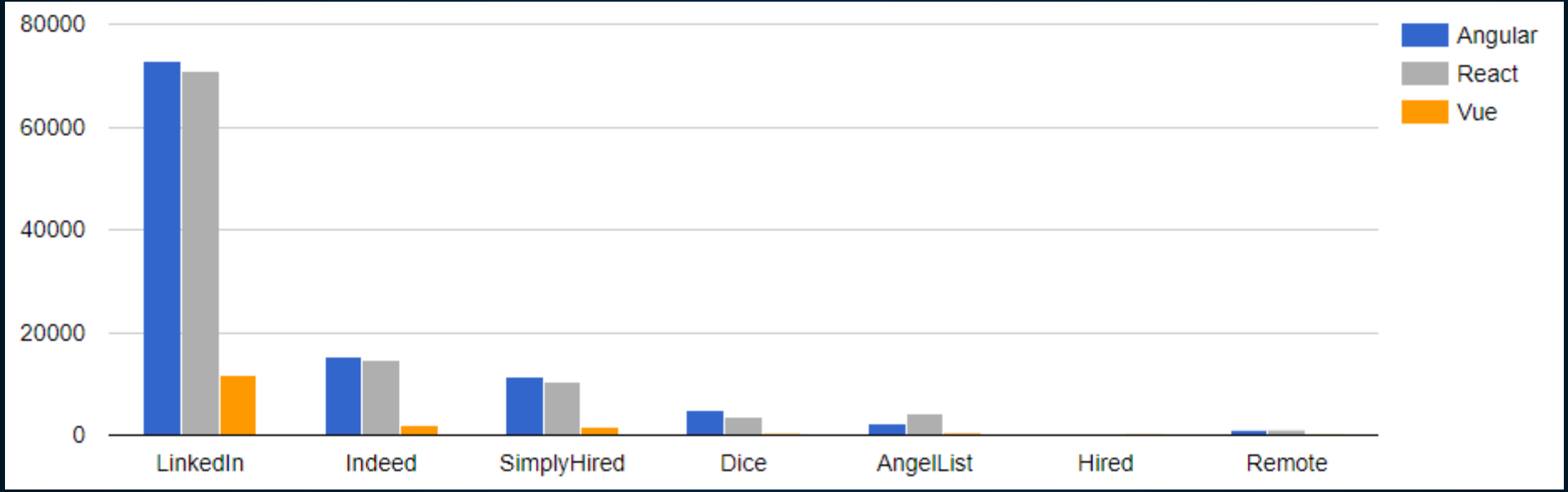
Comparison of Frameworks

	Angular	React	Vue
Initial release	2010	2013	2014
Official site	angular.io	reactjs.org	vuejs.org
Current version	11	17.x	3.0.x
Used by	Google, Wix	Facebook, Uber	Alibaba, GitLab

Comparison of Frameworks



Comparison of Frameworks



Comparison of Frameworks

	Angular	React	Vue
# Watchers	3.2k	6.7k	6.3k
# Stars	70.9k	164k	200.8k
# Forks	18.6k	32.9k	31.7k
# Contributors	1,352	1,533	382

Comparison of Frameworks

- Angular: most mature, good from contributors and is a complete package. However, the learning curve is steep.
- React is just old enough to be mature and has a huge number of contributions from the community. It has gained widespread acceptance. The job market for React is really good, and the future for this framework looks bright. The ability to integrate with other frameworks seamlessly gives it a great advantage for those who would like some flexibility in their code.
- Vue is newest to the arena, without the backing of a major company. However, it has done really well in the last few years to come out as a strong competitor for Angular and React, and especially so with the release of Vue 3.0. Vue should be your choice if you prefer simplicity, but also like flexibility.

Communication between Front & Backend

- REST: Representative State Transfer - lightweight – doesn't remember the client – aka "API" – Five verbs: GET, PUT, POST, DELETE and PATCH.
- "Restful web services" = they use REST APIs
- GraphQL: Graph Query Language – new – can query the database directly from the client side
- Pass message between front and backend
- Old technique: SOAP – envelope – carried state information
- RPC = Remote Procedure Call – call a known procedure

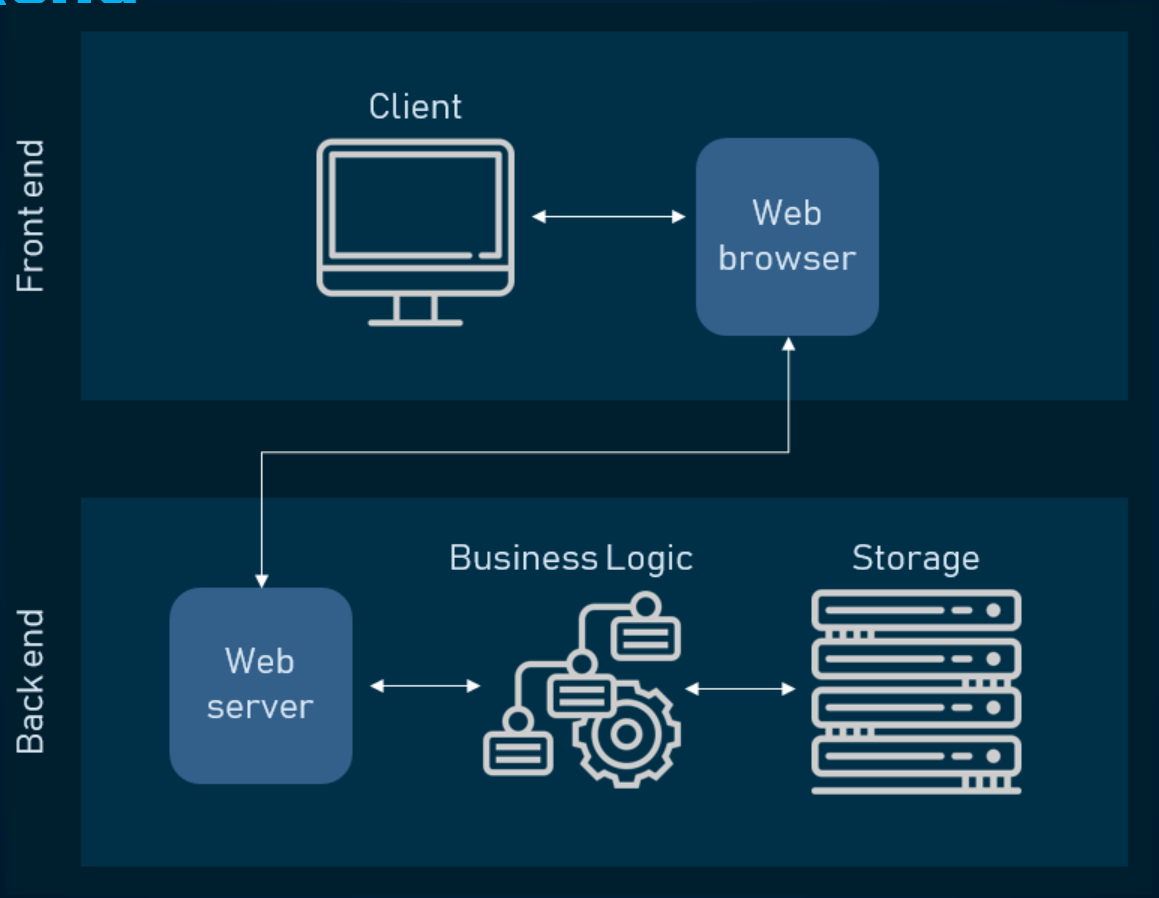
Backend Programming Languages

Unit 3.3

Dr. Mahesha BR Pandit

27-Nov-21 Version 1.0

The Backend


Front end
Client
Web browser
Back end
Web server
Business Logic
Storage

Types

Language classes	Categories	Languages
Programming paradigm	Imperetive procedural	C, C++, C#, Objective-C, Java, Go
	Imperetive scripting	CoffeeScript, JavaScript, Python, Perl, Php, Ruby
	Functional	Clojure, Erlang, Haskell, Scala
Type checking	Static	C, C++, C#, Objective-C, Java, Go, Haskell, Scala
	Dynamic	CoffeeScript, JavaScript, Python, Perl, Php, Ruby, Clojure, Erlang
Implicit type conversion	Disallow	C#, Java, Go, Python, Ruby, Clojure, Erlang, Haskell, Scala
	Allow	C, C++, Objective-C, CoffeeScript, JavaScript, Perl, Php
Memory class	Managed	Others
	Unmanaged	C, C++, Objective-C

We omit TypeScript from language classification as it allows both explicit and implicit type conversion.

Characteristics

- A programming language must be simple, easy to learn and use, have good readability, and be human recognizable.
- Abstraction is a must-have Characteristics for a programming language in which the ability to define the complex structure and then its degree of usability comes.
- A portable programming language is always preferred.
- Programming language's efficiency must be high so that it can be easily converted into a machine code and executed consumes little space in memory.
- A programming language should be well structured and documented so that it is suitable for application development.
- Necessary tools for the development, debugging, testing, maintenance of a program must be provided by a programming language.
- A programming language should provide a single environment known as Integrated Development Environment(IDE).
- A programming language must be consistent in terms of syntax and semantics

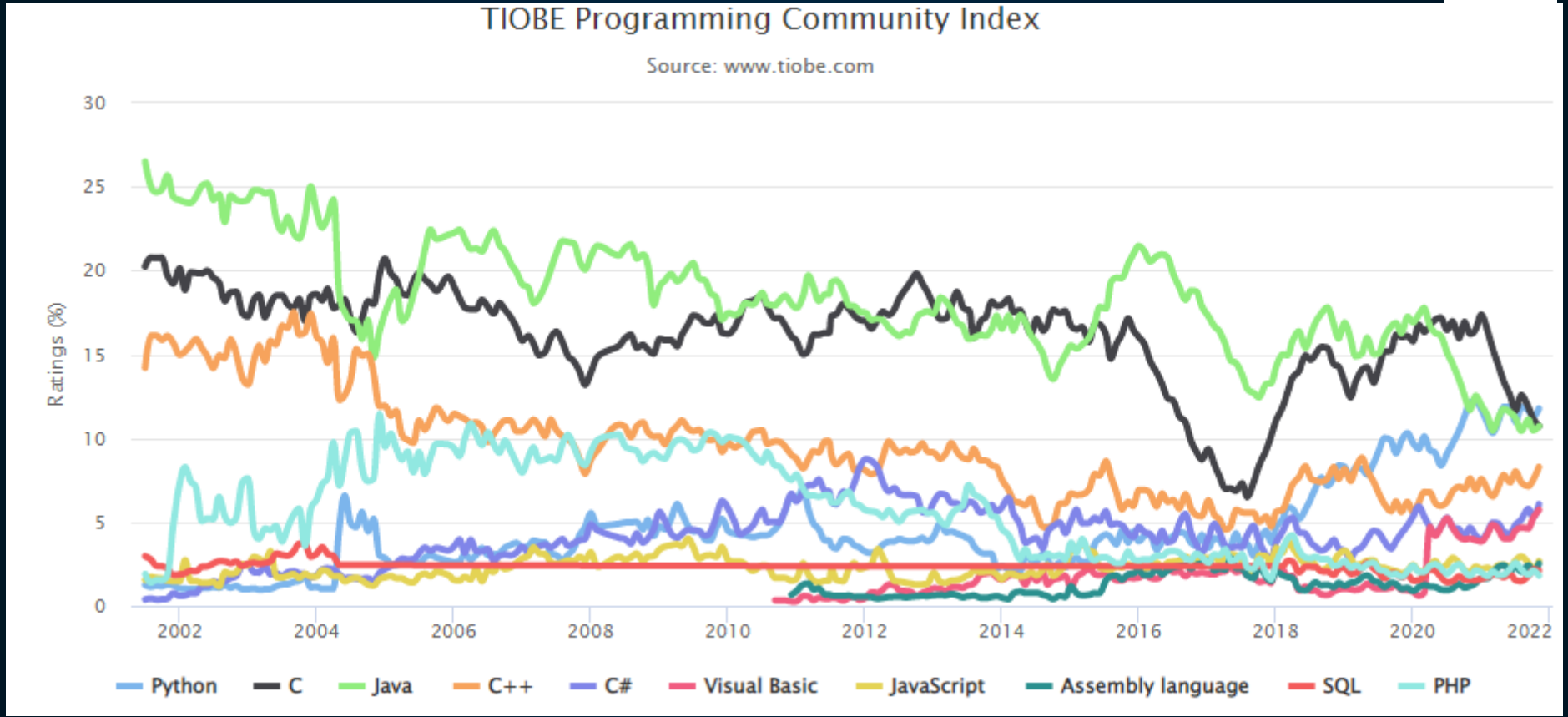
Top 10 Languages as of November 2021

Nov 2021	Nov 2020	Change	Programming Language	Ratings	Change
1	2	↑	Python	11.77%	-0.35%
2	1	↓	C	10.72%	-5.49%
3	3		Java	10.72%	-0.96%
4	4		C++	8.28%	+0.69%
5	5		C#	6.06%	+1.39%
6	6		Visual Basic	5.72%	+1.72%
7	7		JavaScript	2.66%	+0.63%
8	16	⇈	Assembly language	2.52%	+1.35%
9	10	↑	SQL	2.11%	+0.58%
10	8	↓	PHP	1.81%	+0.02%

Long Term History

Programming Language	2021	2016	2011	2006	2001	1996	1991	1986
C	1	2	2	2	1	1	1	1
Python	2	5	6	8	20	27	-	-
Java	3	1	1	1	2	14	-	-
C++	4	3	3	3	3	2	2	5
C#	5	4	4	7	12	-	-	-
Visual Basic	6	13	-	-	-	-	-	-
JavaScript	7	7	10	9	9	20	-	-
PHP	8	6	5	5	10	-	-	-
Assembly language	9	10	-	-	-	-	-	-
SQL	10	-	-	-	36	-	-	-
Fortran	20	25	29	21	25	5	3	8
Ada	28	29	18	16	18	10	5	2
Lisp	31	27	13	13	16	7	4	3
(Visual) Basic	-	-	8	4	4	3	8	6

Community Index



Hall of Fame

Year	Winner
2020	Python
2019	C
2018	Python
2017	C
2016	Go
2015	Java
2014	JavaScript
2013	Transact-SQL
2012	Objective-C
2011	Objective-C
2010	Python

Popular Programming Languages

- Javascript — Netflix, Candy Crush, Linkedin
- Python — Instagram, Spotify, Instacart
- Ruby — Fiverr, Github, Sendgrid
- PHP — WordPress, Mailchimp, Yahoo
- Java — Wikipedia Search, Minecraft, Twitter
- C# — Microsoft Visual Studio, Windows Installer, NMath
- Perl — IMDB, BBC, Duckduckgo
- C++ — Adobe Photoshop, Youtube, Firefox
- Kotlin — Square, Trello, Slack
- Scala — Akka, Spark, Slick

JavaScript

- Most popular
- Node.js with Express Framework
- Quick development, Lenient backend, any middleware, handle the I/O requests, Open-Source, Fewer scripting overheads
- The event-driven functions are very complicated.
- Most of the programmers using JavaScript are unable to understand the middleware used with it.
- JavaScript requires MySQL for database services that are very complex and outdated.
- Most developers do not prefer the freedom that the JavaScript backend framework offers.

Python

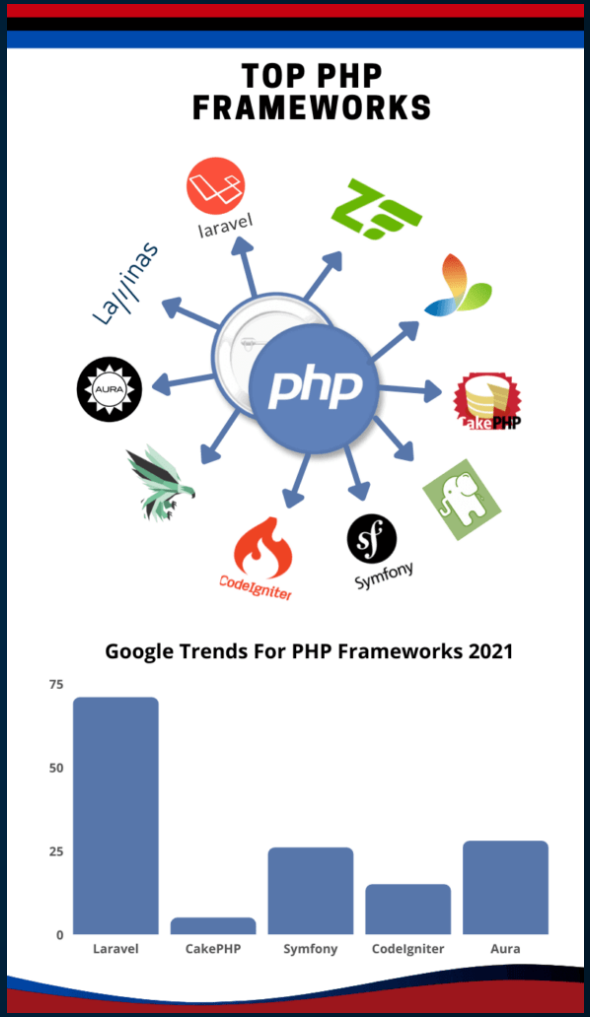
- Came up in 1991
- Multi-purpose, Object-oriented, clean, many libraries
- Easy to learn, many libraries, IoT features, Embedded Code, Cost Efficient
- The database access layer is not as well developed as other languages.
- The programs developed on Python require a lot of testing and debugging.
- Python is highly dependent on 3rd party frameworks and libraries.
- If the application is interrupted, the execution time becomes very slow.

Ruby

- Came up in 1990
- Syntax similar to Java and Python
- Great automation capabilities
- Productivity, Meta Programming, Testing Features, Reliable and Quick
- The runtime of Ruby is comparatively slower than other languages.
- The number of libraries and sources available for Ruby is not great.
- As Ruby is a new language, the current developers find it a bit difficult to learn.
- The programs coded in Ruby are difficult to debug.

PHP

- Developed in 1994
- World's best server-side development language
- About 79% of the world's websites are made using PHP
- Easy to use, open-source, versatile, secure, helps automation, community driven
- Many PHP Frameworks
- Not popular now
- Lacks many features



Java

- Developed in 1991, officially published in 1995
- One of the best programming languages in the world.
- Second-best programming language in the 2021
- Scalable, Easy to understand, multi-threading, open-source, secure, WORA
- The programming language Java is time-consuming.
- There is no low-level programming done in Java.
- There are no commands for garbage data collection.
- Java is a language that runs better on expensive or high-end systems. It makes Java development very expensive.
- In the GUI section of the Java development applications, many necessary things like trendy interface objects and tools are missing.

C#

- One of the most popular languages for making the backend
- Automation on Windows Servers
- Very Fast
- Game development
- Cross platform development, compatibility, good garbage collection
- Cannot directly interact with hardware
- Only MS Windows
- Runs only on .NET framework

Perl

- 30 years of history
- Compatible with many platforms, Open source, Extensible, Good text processing
- The processing or programs developed in the Perl language are not as good as some other languages.
- The libraries available in Pearl lack some of the most necessary features.
- The process of bug fixing in Perl is not as easy, and it is very challenging.
- If your code is extensive, then it will not only be challenging to handle it, but it will also be very challenging for the system to handle the code.
- Scalability is not an option available when you use Perl.
- The speed of application developed on Perl is not that great.
- It is a costly language because it is an older technology, and there are not many developers available. Additionally, this language takes a lot of time to learn and get a grip on.
- The libraries available in Perl are not free as you have to pay copying charges for them.

C++

- Advanced version of the C language, supports OO, good for data science
- Low level language, can interact with hardware
- Portable, good memory management
- Poor garbage collection
- Poor hardware security

Kotlin

- Created in 2011
- Overtaking Java
- Supports cross-framework development
- Less code, Java compatible, Easy to manage
- There are not a lot of primitive variable types available in Kotlin. It makes managing variables throughout your code somewhat difficult. It can also cause output issues sometimes.
- The functions here also have primitive types like traditional languages. It makes it difficult for the developers who are used to other languages.
- The compilation for android applications is slower if we compare it to that of Java.

Scala

- Functional programming + object-oriented programming
- Concise code, easy coding, Java compatibility
- Unconventional coding – longer to learn

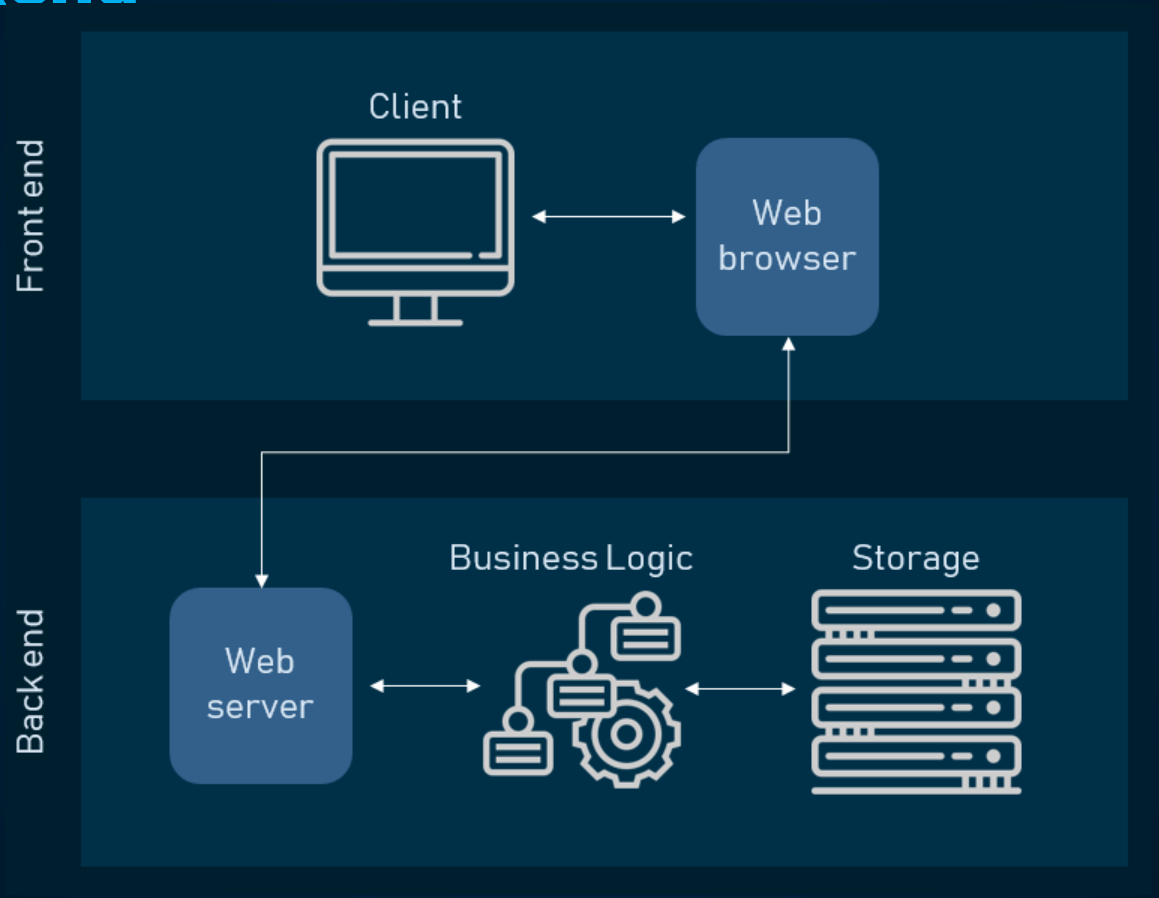
Database Technologies

Unit 3.4

Dr. Mahesha BR Pandit

30-Nov-21 Version 1.0

The Backend


Front end
Client
Web browser
Back end
Web server
Business Logic
Storage

Databases: Overview

- A database is a collection of information stored within a computer.
- Databases allow computers to store essential information in an organized and easily searchable way.
- Database technology has improved over the years
- There are many types of databases
- Centralized database
- Cloud database
- Commercial database
- Distributed database
- End-user database
- Graph database
- NoSQL database
- Object-oriented database
- Open-source database
- Operational database
- Personal database
- Relational database

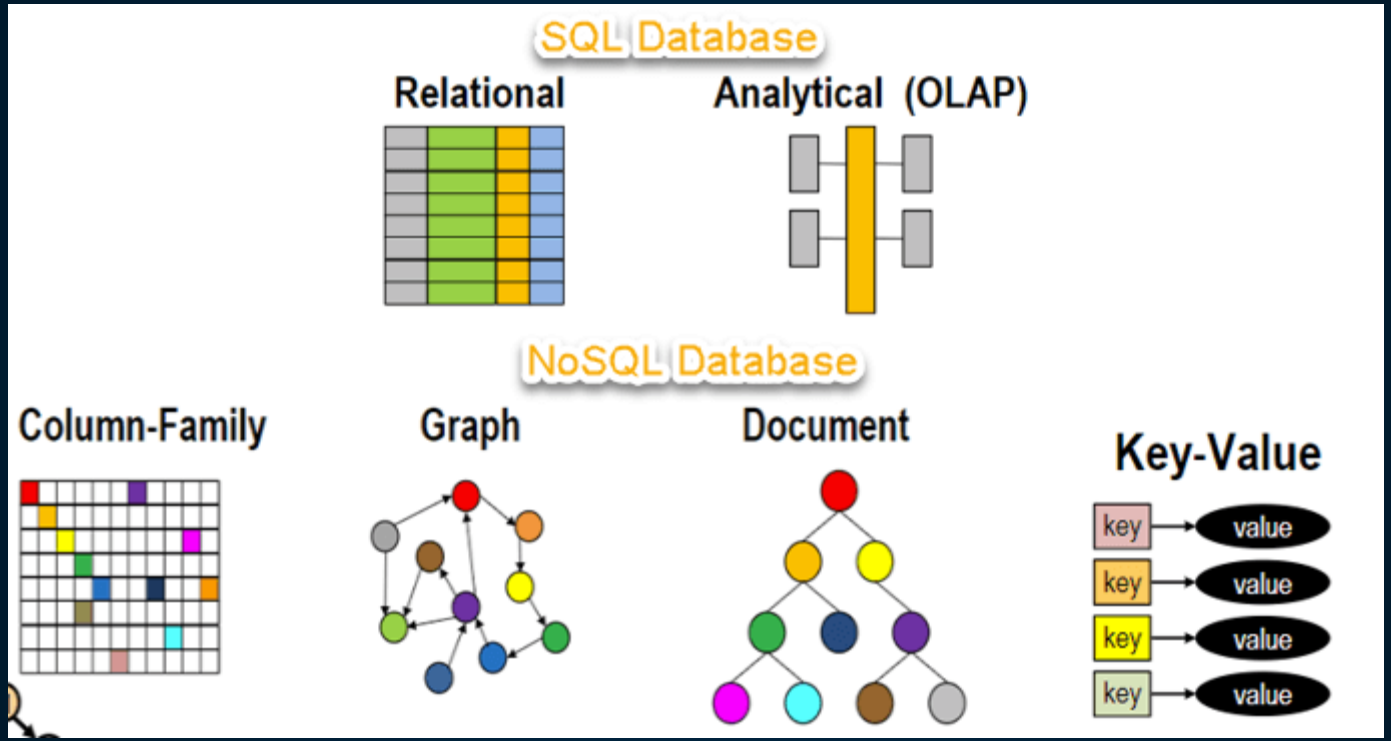
Usage Types

- Centralized: Single location, located on a central computer, local usage
- Cloud/Multi Tenant: Hosted on cloud, access via the Internet, encrypted
- Commercial: Paid, feature-rich, backed-up, legal implications, expensive
- Distributed: Multiple locations, located on multiple computers, sharded
- End User Databases: Personal, located on laptops/desktops, like MS Excel
- Graph Databases: Data + Relation between Data, Open Model, Fast
- NOSQL Databases: Unstructured, non-relational
- Object Oriented: Data is represented as classes and objects, relational, large size, fast processing
- Open Source: Wide public usage, often free, modifiable, inexpensive
- Operational: Real time operations, ideal for data-warehousing
- Personal Databases: Single machine, simple design, personal use
- Relational: Structured, High integrity, rigid schema

Database Technologies

- 4D (4th Dimension)
- ADABAS
- Altibase
- AmazonRDS
- Cloudera
- CouchDB
- Couchbase
- DbVisualizer
- FileMaker
- Hadoop HDFS
- IBM DB2
- Informix
- Informix Dynamic Server
- ManageEngine Applications Manager
- MariaDB
- Microsoft Access
- Microsoft SQL Server
- MongoDB
- MySQL
- Neo4j
- Oracle RDBMS
- OrientDB
- PostgresSQL
- Redis
- Robomongo
- SAP Sybase ASE
- SQL Developer
- SQLite
- Seqel PRO
- SolarWinds Database Performance Analyzer
- Teradata
- Toad
- phpMyAdmin

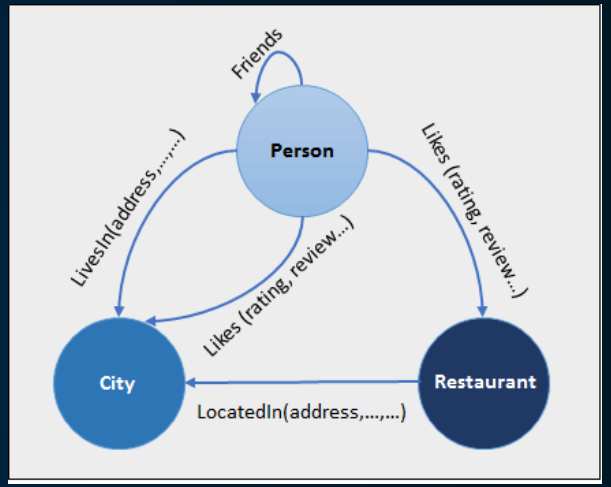
Types


SQL Database
Relational
Analytical (OLAP)
NoSQL Database
Column-Family
Graph
Document
Key-Value
key
value
key
value
key
value
key
value

Types - Illustrated

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value



Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

Document 1

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

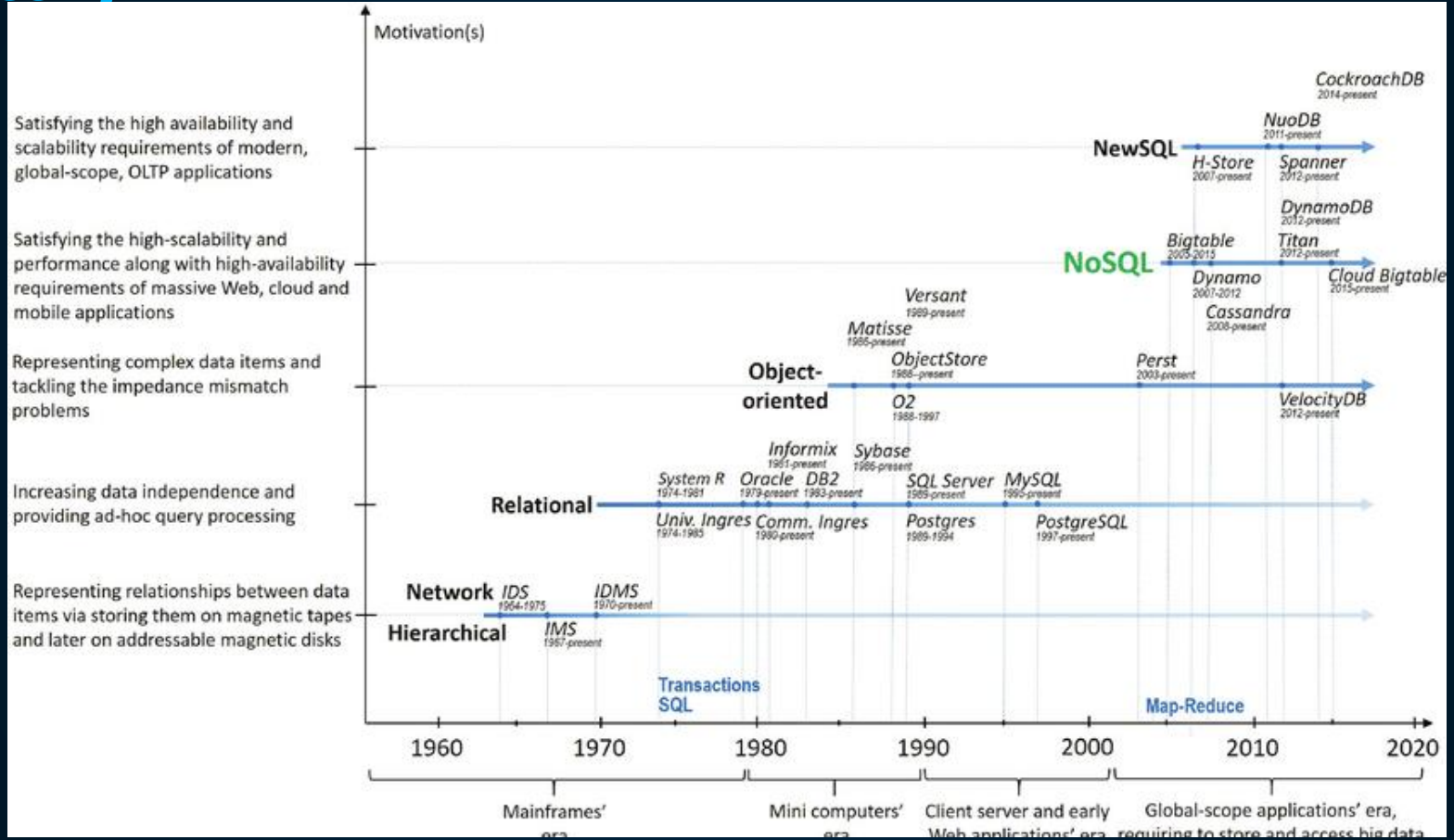
Document 2

```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

Document 3

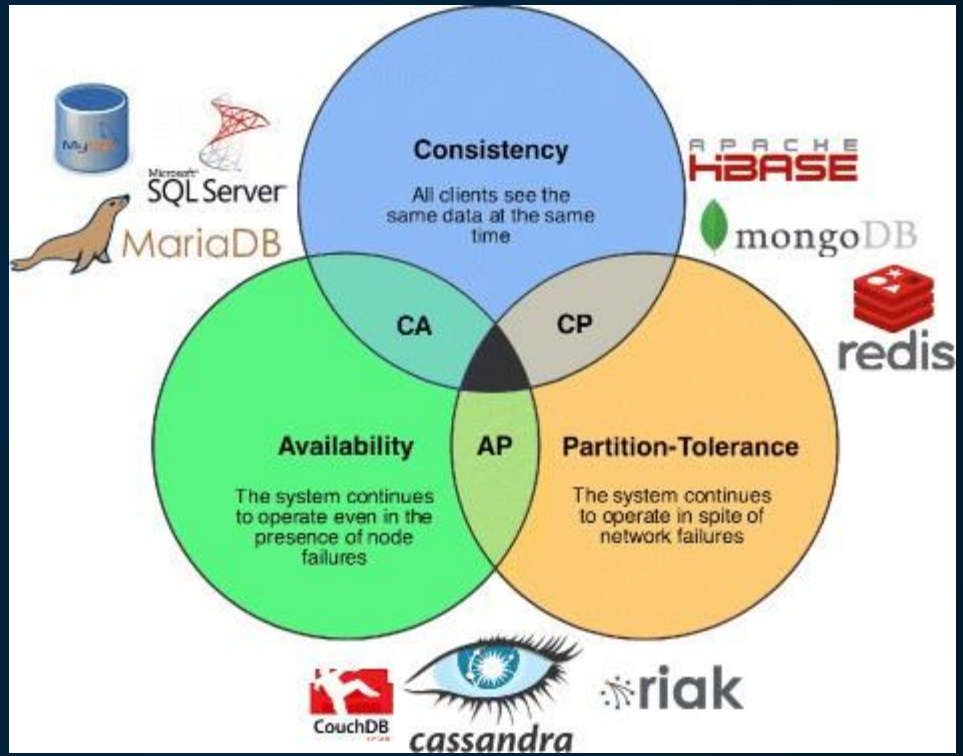
```
{
  "prop1": data,
  "prop2": data,
  "prop3": data,
  "prop4": data
}
```

History


Motivation(s)
Satisfying the high availability and scalability requirements of modern, global-scope, OLTP applications
Satisfying the high-scalability and performance along with high-availability requirements of massive Web, cloud and mobile applications
Representing complex data items and tackling the impedance mismatch problems
Increasing data independence and providing ad-hoc query processing
Representing relationships between data items via storing them on magnetic tapes and later on addressable magnetic disks
NewSQL
NoSQL
Object-oriented
Relational
Network
Hierarchical
CockroachDB 2014-present
NuoDB 2011-present
H-Store 2007-present
Spanner 2012-present
DynamoDB 2012-present
Bigtable 2005-2015
Titan 2012-present
Dynamo 2007-2012
Cloud Bigtable 2015-present
Cassandra 2008-present
Versant 1989-present
Matisse 1985-present
ObjectStore 1988-present
O2 1988-1997
Perst 2003-present
VelocityDB 2012-present
Informix 1981-present
Sybase 1986-present
System R 1974-1981
Oracle 1979-present
DB2 1983-present
SQL Server 1989-present
MySQL 1995-present
Univ. Ingres 1974-1985
Comm. Ingres 1990-present
Postgres 1989-1994
PostgreSQL 1997-present
IDS 1964-1975
IDMS 1970-present
IMS 1967-present
Transactions SQL
Map-Reduce
1960
1970
1980
1990
2000
2010
2020
Mainframes' era
Mini computers'
Client server and early Web applications' era
Global-scope applications' era, requiring to store and access big data

CAP Theorem

- Also known as Brewer's theorem
- Pick any two
- Only two options when the data is partitioned – AP and CP
- AP(Availability and Partition tolerance)
- CP(Consistency and Partition tolerance)



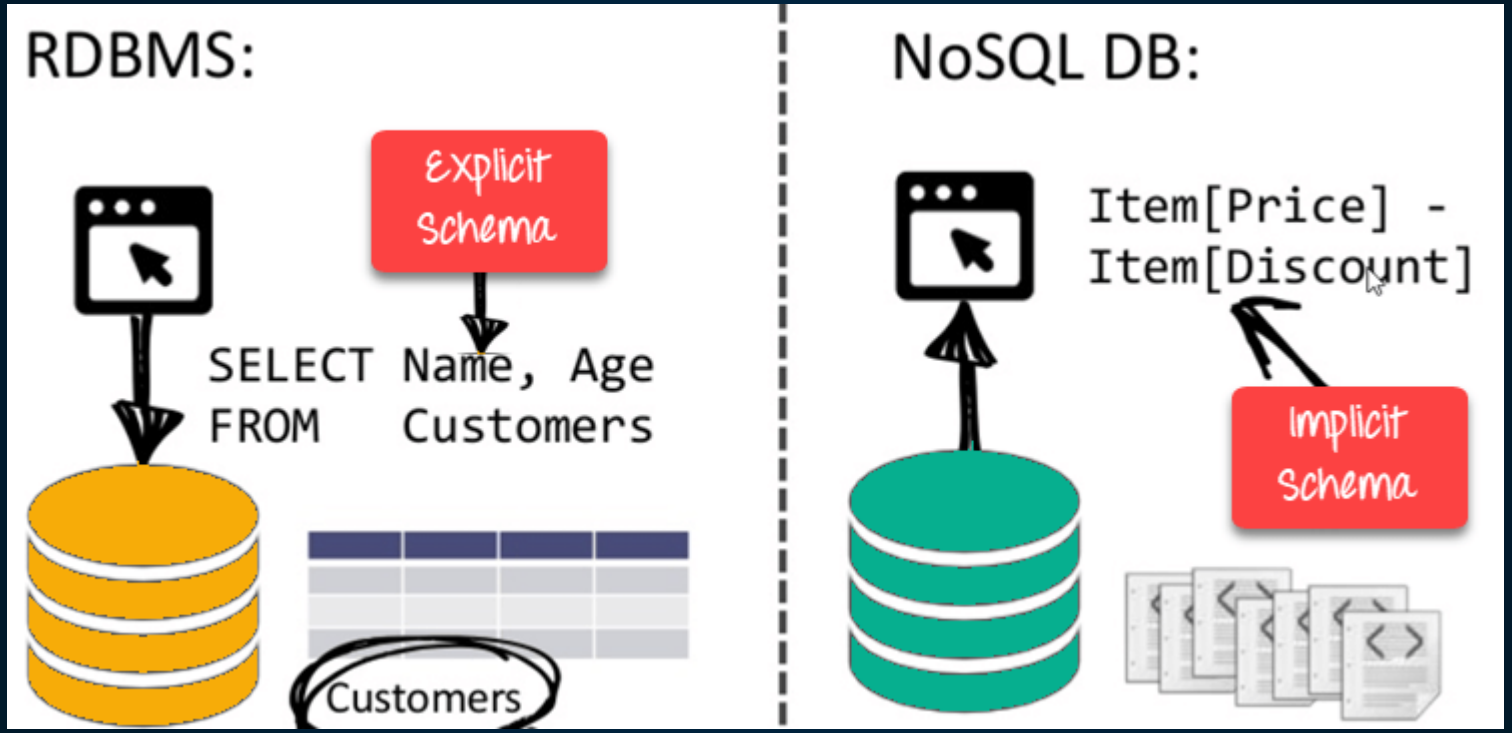
Relational Databases

- Since the 1970s
- Data is stored in multiple, related tables.
- Within the tables, data is stored in rows and columns.
- The relational database management system (RDBMS) is the program that allows you to create, update, and administer a relational database.
- Structured Query Language (SQL) is the most common language for reading, creating, updating and deleting data.
- Relational databases are very reliable.
- They are compliant with ACID (Atomicity, Consistency, Isolation, Durability), which is a standard set of properties for reliable database transactions.
- Relational databases work well with structured data.
- Organizations that have a lot of unstructured or semi-structured data should not be considering a relational database.
- Examples: Microsoft SQL Server, Oracle Database, MySQL, PostgreSQL and IBM DB2

NoSQL Databases

- Not Only SQL
- Any database that doesn’t use SQL as its primary data access language.
- Non-relational databases.
- Doesn’t have to conform to a pre-defined schema,
- Great for organizations seeking to store unstructured or semi-structured data.
- Can make changes to the database on the fly, without affecting applications that are using the database.
- Examples: Apache Cassandra, MongoDB, CouchDB, and CouchBase

RDBMS versus NoSQL



Types of NoSQL Databases

Mahesha
BR Pandit

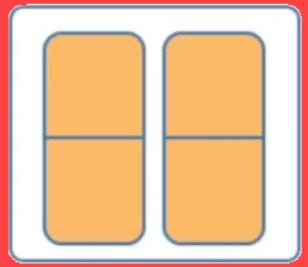
Key Value

Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

Document-Based

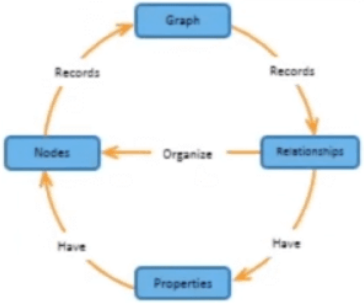
Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-Based



Example:
BigTable, Cassandra, Hbase, Hypertable

Graph-Based



Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Cloud, Multi-Tenant Databases

- Any database that’s designed to run in the cloud.
- Offer flexibility and scalability, along with high availability.
- Often low-maintenance, since many are offered via a SaaS model.
- Support multiple tenants
- Encrypted
- Examples: Microsoft Azure SQL Database, Amazon Relational Database Service, Oracle Autonomous Database.

Columnar Databases

- Also referred to as column data stores
- Store data in columns rather than rows.
- Often used in data warehouses because they’re great at handling analytical queries.
- When you are querying a columnar database, it essentially ignores all data that doesn’t apply to the query, because you can retrieve the information from only the columns you want.
- Examples: Google BigQuery, Cassandra, HBase, MariaDB, Azure SQL Data Warehouse

Wide Column Databases

- Also known as wide column stores
- Schema-agnostic.
- Data is stored in column families, rather than in rows and columns.
- Highly scalable
- Can handle petabytes of data
- Ideal for supporting real-time big data applications.
- Examples: BigTable, Apache Cassandra and Scylla

Object Oriented Databases

- Based on object-oriented programming
- Data and all its attributes, are tied together as an object.
- Managed by object-oriented database management systems (OODBMS).
- Work well with object-oriented programming languages, such as C++ and Java.
- Like relational databases, object-oriented databases conform to ACID standards.
- Examples: Wakanda, ObjectStore

Key-Value Databases

- One of the simplest types of NoSQL databases
- Key-value databases save data as a group of key-value pairs made up of two data items each.
- Also referred to as a key-value store.
- Highly scalable and can handle high volumes of traffic,
- Ideal for processes such as session management for web applications, user sessions for massive multi-player online games, and online shopping carts.
- Examples: Amazon DynamoDB, Redis

Hierarchical Databases

- Use a parent-child model to store data.
- Looks like a family tree, with one object on top branching down to multiple objects beneath it.
- The one-to-many format is rigid
- Child records can’t have more than one parent record.
- Originally developed by IBM in the early 1960s
- Commonly used to support high-performance and high availability applications.
- Examples: IBM Information Management System (IMS), Windows Registry

Document Databases

- Also known as document stores
- Use JSON-like documents to model data instead of rows and columns.
- Also referred to as document-oriented databases
- Designed to store and manage document-oriented information, semi-structured data.
- Simple and scalable
- Useful for many modern applications, even mobile apps that need fast iterations.
- Examples: MongoDB, Amazon DocumentDB, Apache CouchDB

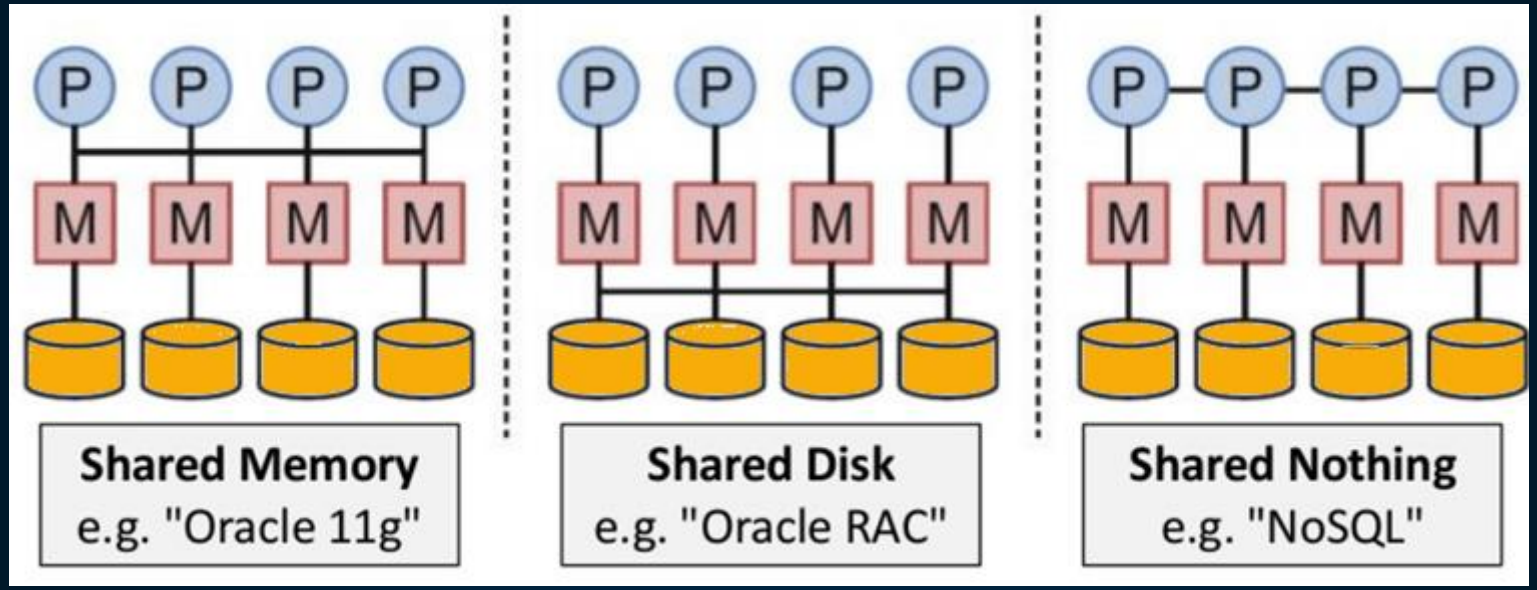
Graph Databases

- A type of NoSQL database
- Based on graph theory.
- Graph-Oriented Database Management Systems (DBMS) software is designed to identify and work with the connections between data points. Therefore
- Often used to analyze the relationships between heterogeneous data points, such as in fraud prevention or for mining data about customers from social media.
- Examples: Datastax Enterprise Graph, Neo4J

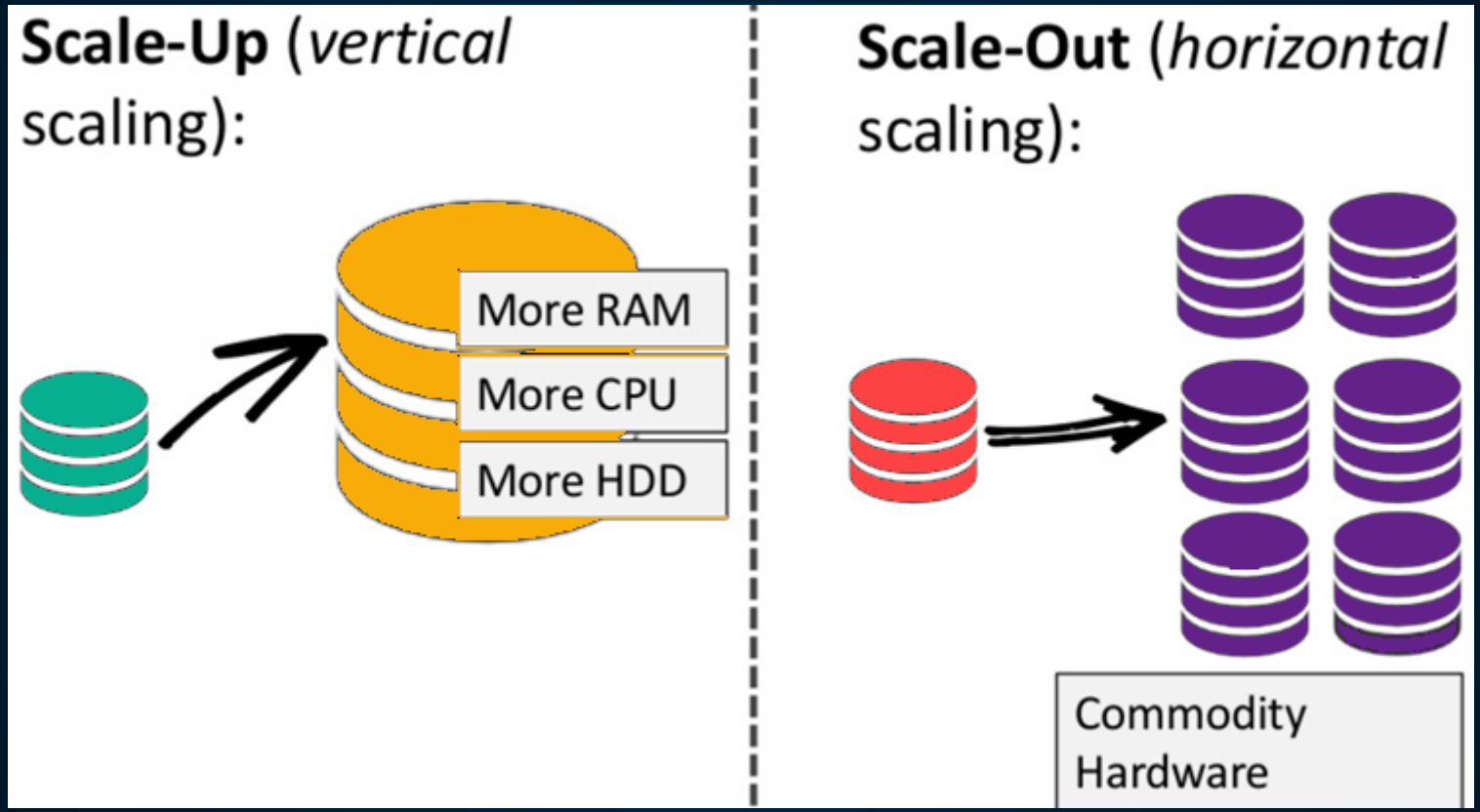
Time series Databases

- Optimized for time-stamped, or time series, data.
- Examples of this type of data include network data, sensor data, and application performance monitoring data.
- Internet of Things sensors put out a constant stream of time series data.
- Examples: Druid, eXtremeDB, InfluxDB

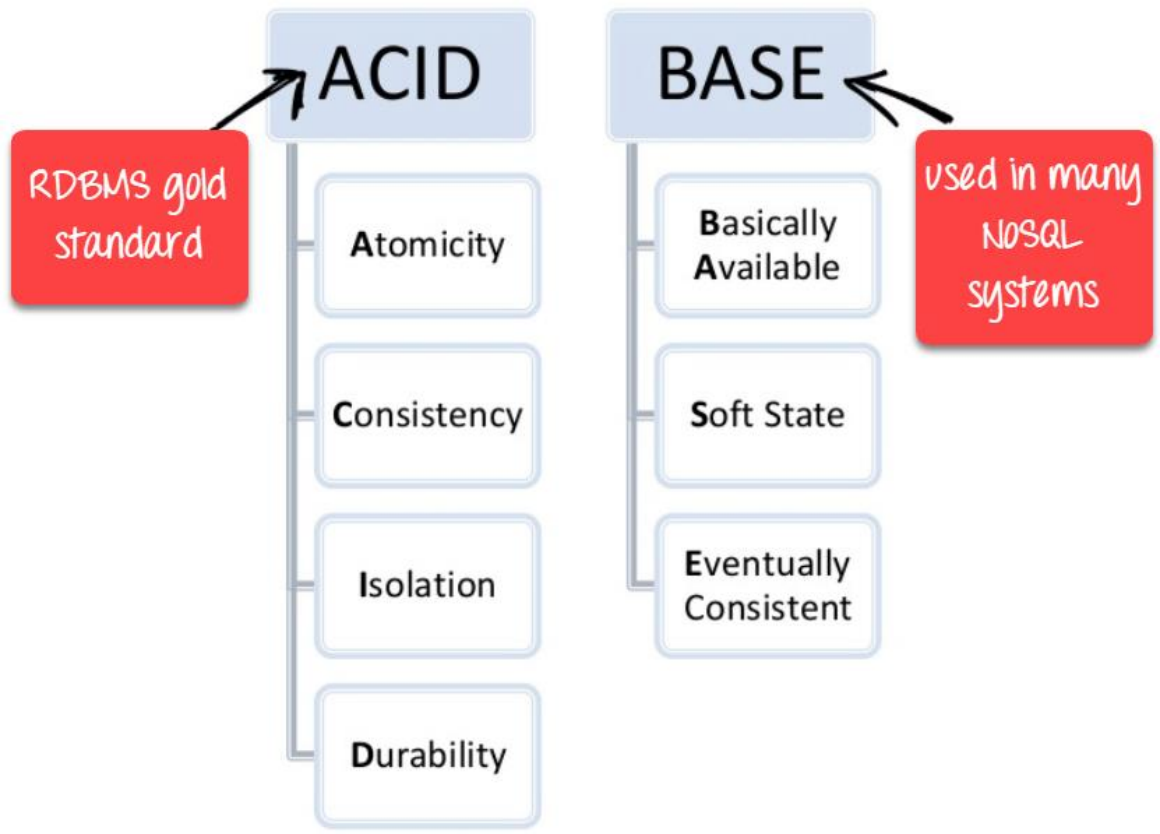
Sharing Architectures



Database Scalability



Standard Features



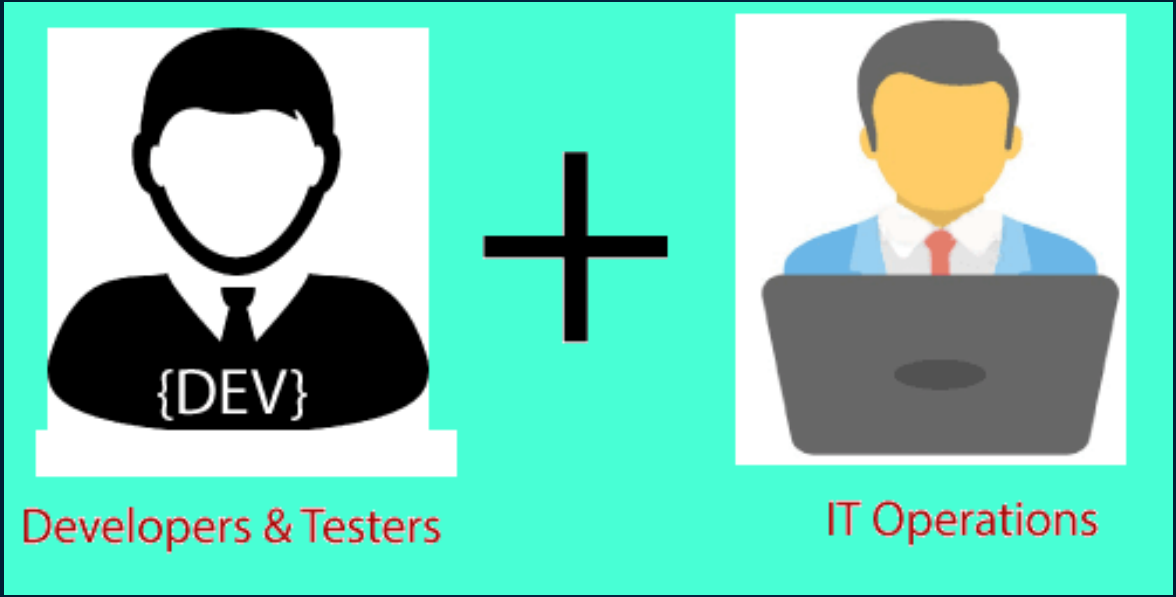
DevOps Technologies

Unit 3.5

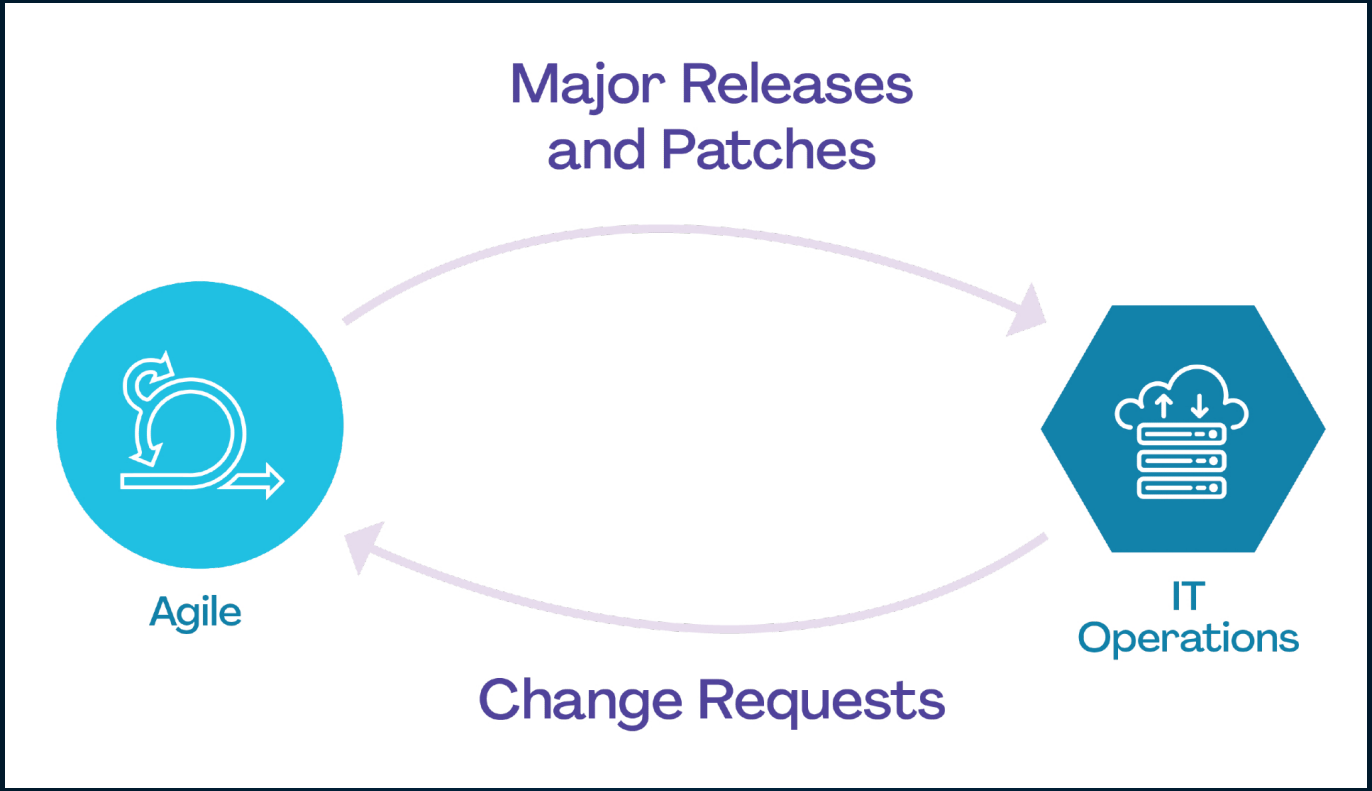
Dr. Mahesha BR Pandit

03-Dec-21 Version 1.0

DevOps = Dev + Ops



Dev and Ops Interaction



Why do we need DevOps?

- The operation and development team worked in complete isolation.
- After the design-build, the testing and deployment are performed respectively. That's why they consumed more time than actual build cycles.
- Without the use of DevOps, the team members are spending a large amount of time on designing, testing, and deploying instead of building the project.
- Manual code deployment leads to human errors in production.
- Coding and operation teams have their separate timelines and are not in synch, causing further delays.

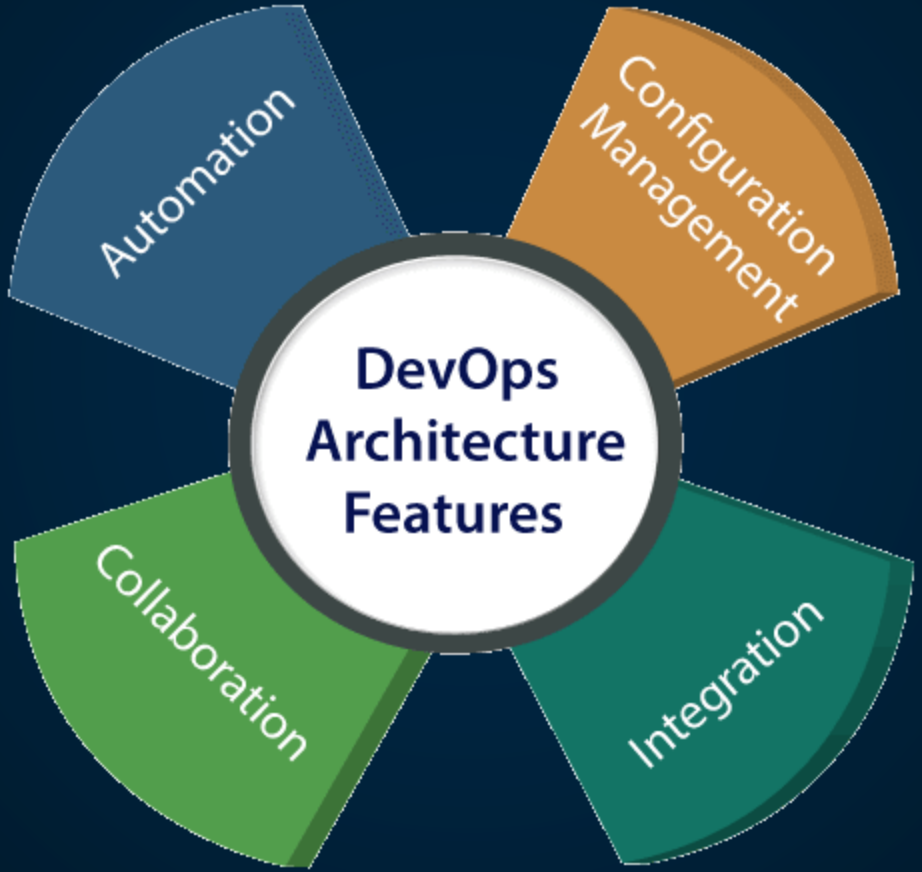
DevOps Principles

- Continuous delivery, automation, and fast response to feedback.
- End to End Responsibility: Provide performance support until end of life of product. It enhances the responsibility and the quality of the products engineered.
- Continuous Improvement: Minimize waste. It continuously speeds up the growth of products or services offered.
- Automate Everything: Both software development and entire infrastructure landscape.
- Custom Centric Action: Do everything for the customer
- Monitor and test everything: Identify and remove defects
- Work as one team: Designers, developers, and testers to work as one team with complete collaboration.

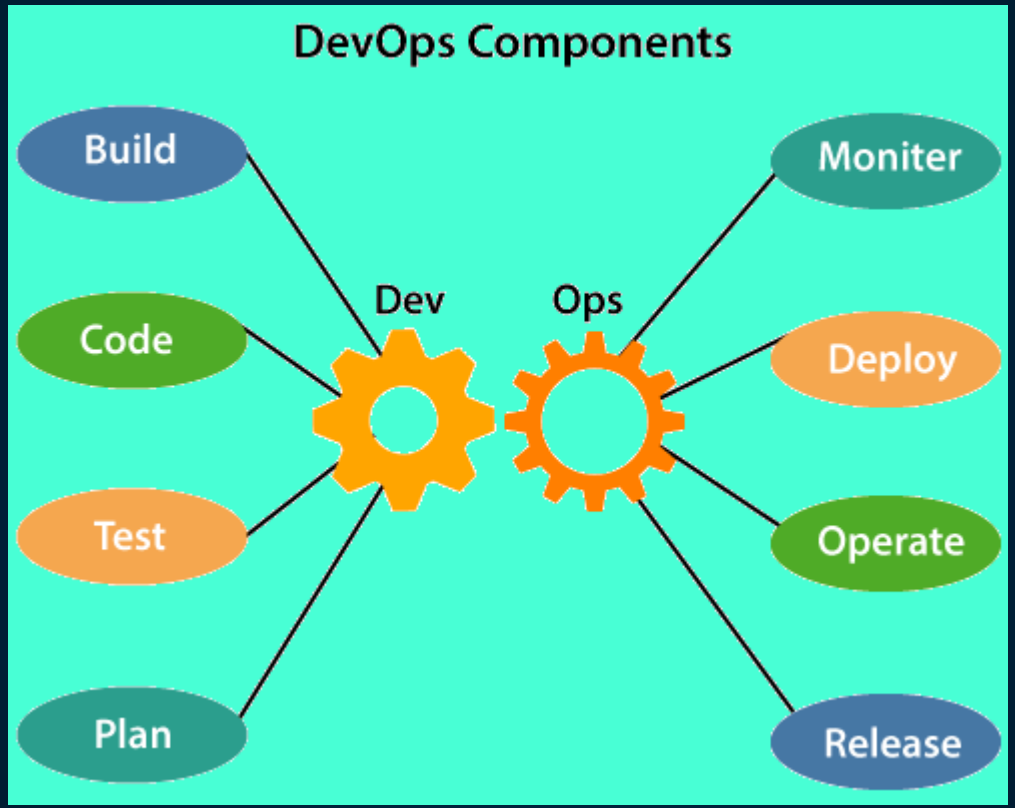
What happens in DevOps?

- Self-service configuration
- Continuous build
- Continuous integration
- Continuous delivery
- Incremental testing
- Automated provisioning
- Automated release management

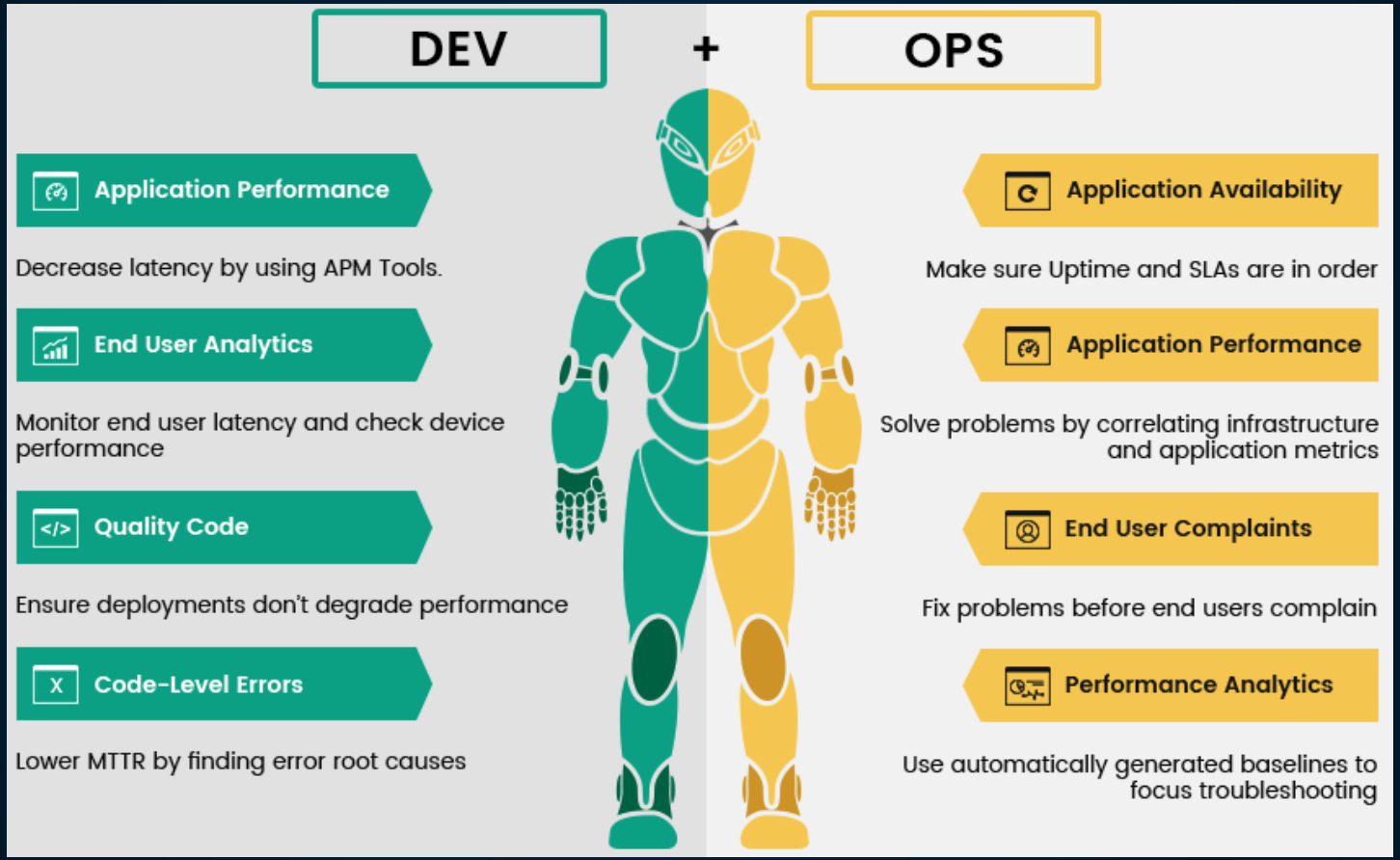
DevOps Architecture



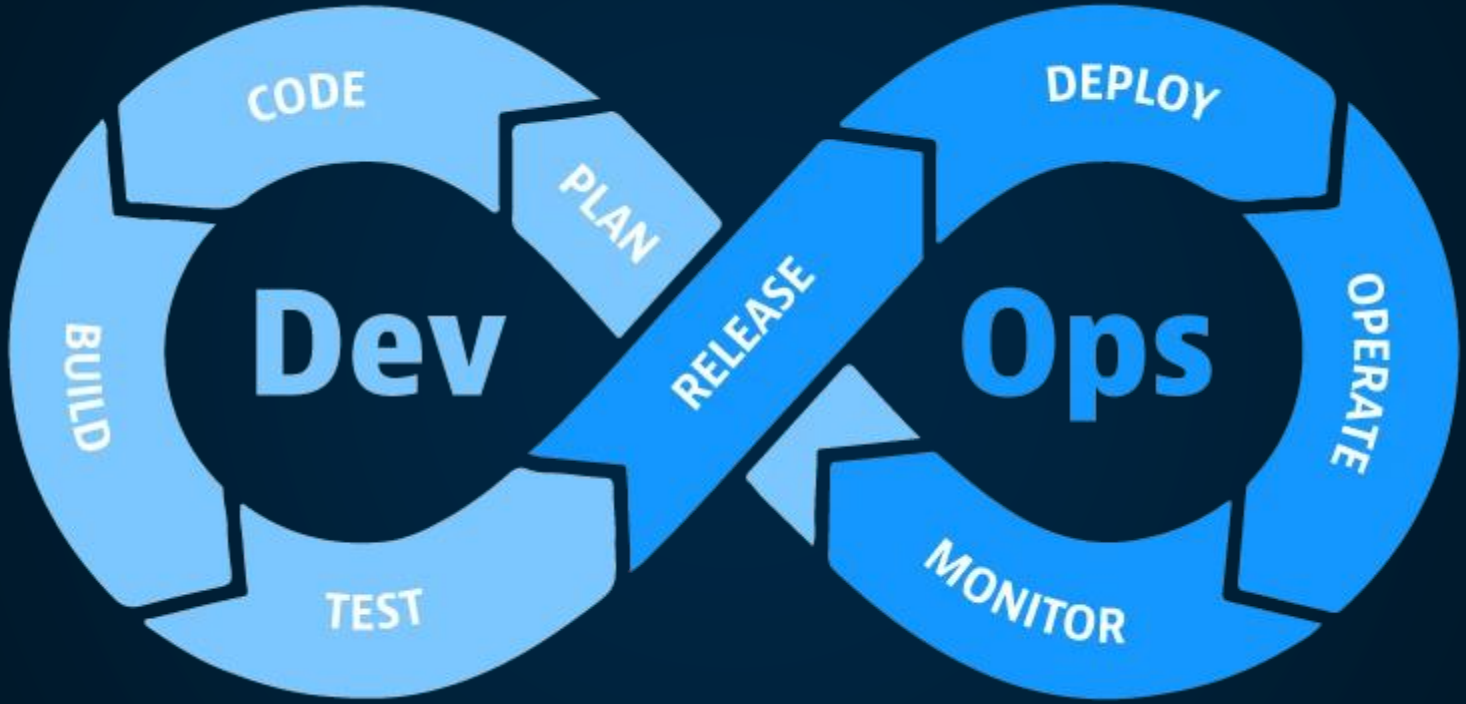
DevOps Components



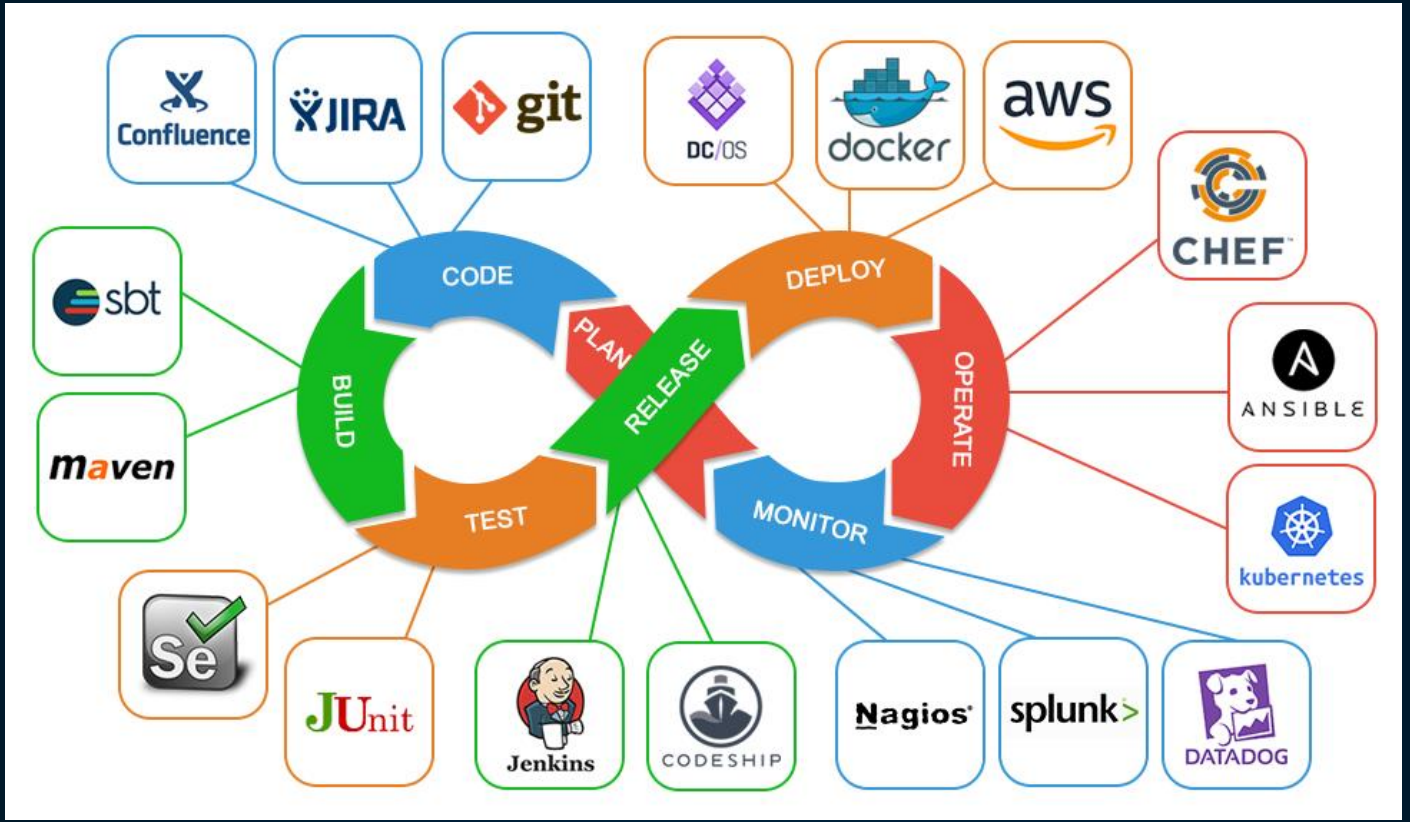
Work Split


DEV
+
OPS
Application Performance
Decrease latency by using APM Tools.
End User Analytics
Monitor end user latency and check device performance
Quality Code
Ensure deployments don't degrade performance
Code-Level Errors
Lower MTTR by finding error root causes
Application Availability
Make sure Uptime and SLAs are in order
Application Performance
Solve problems by correlating infrastructure and application metrics
End User Complaints
Fix problems before end users complain
Performance Analytics
Use automatically generated baselines to focus troubleshooting

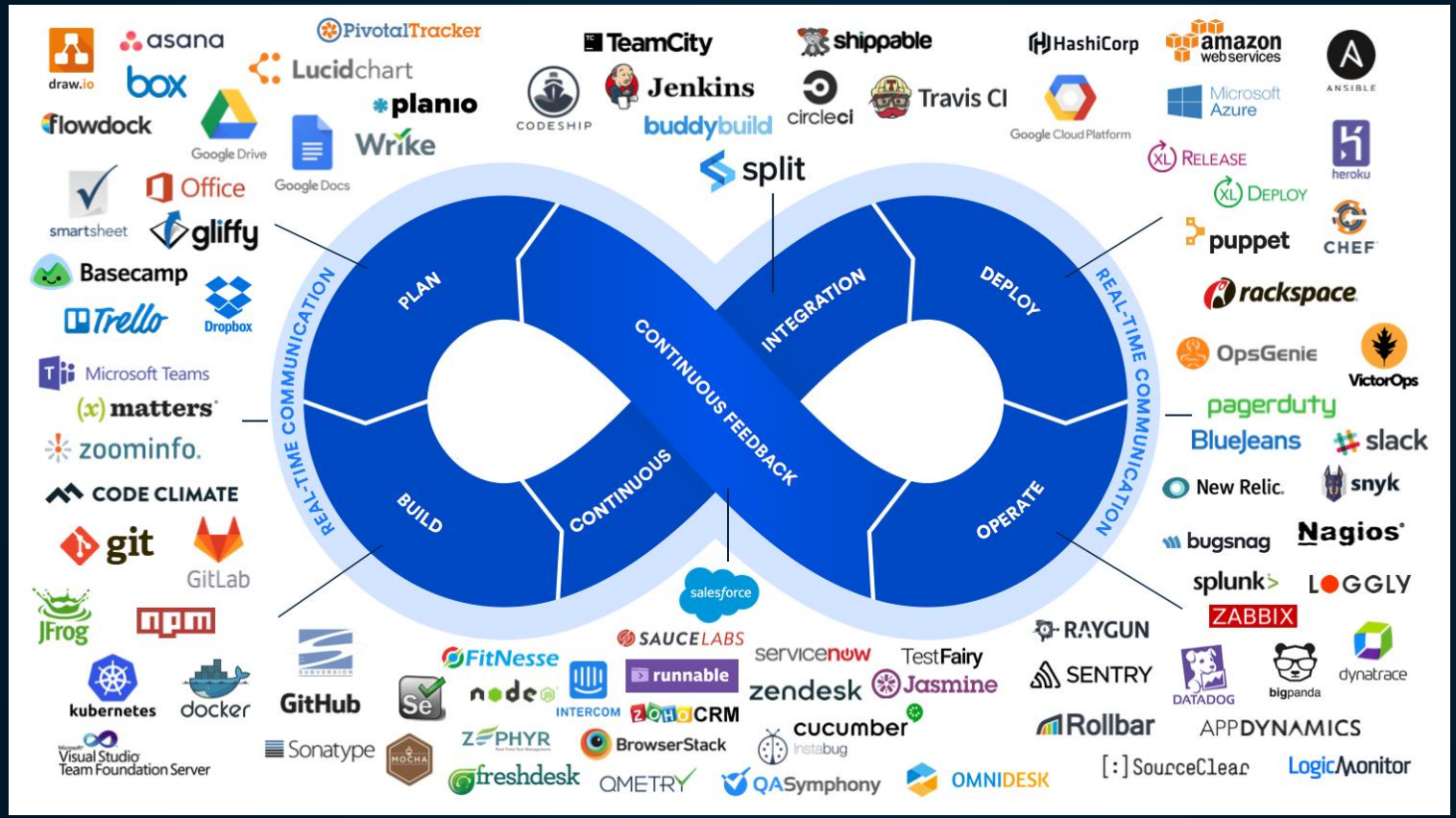
Infinite DevOps Cycle



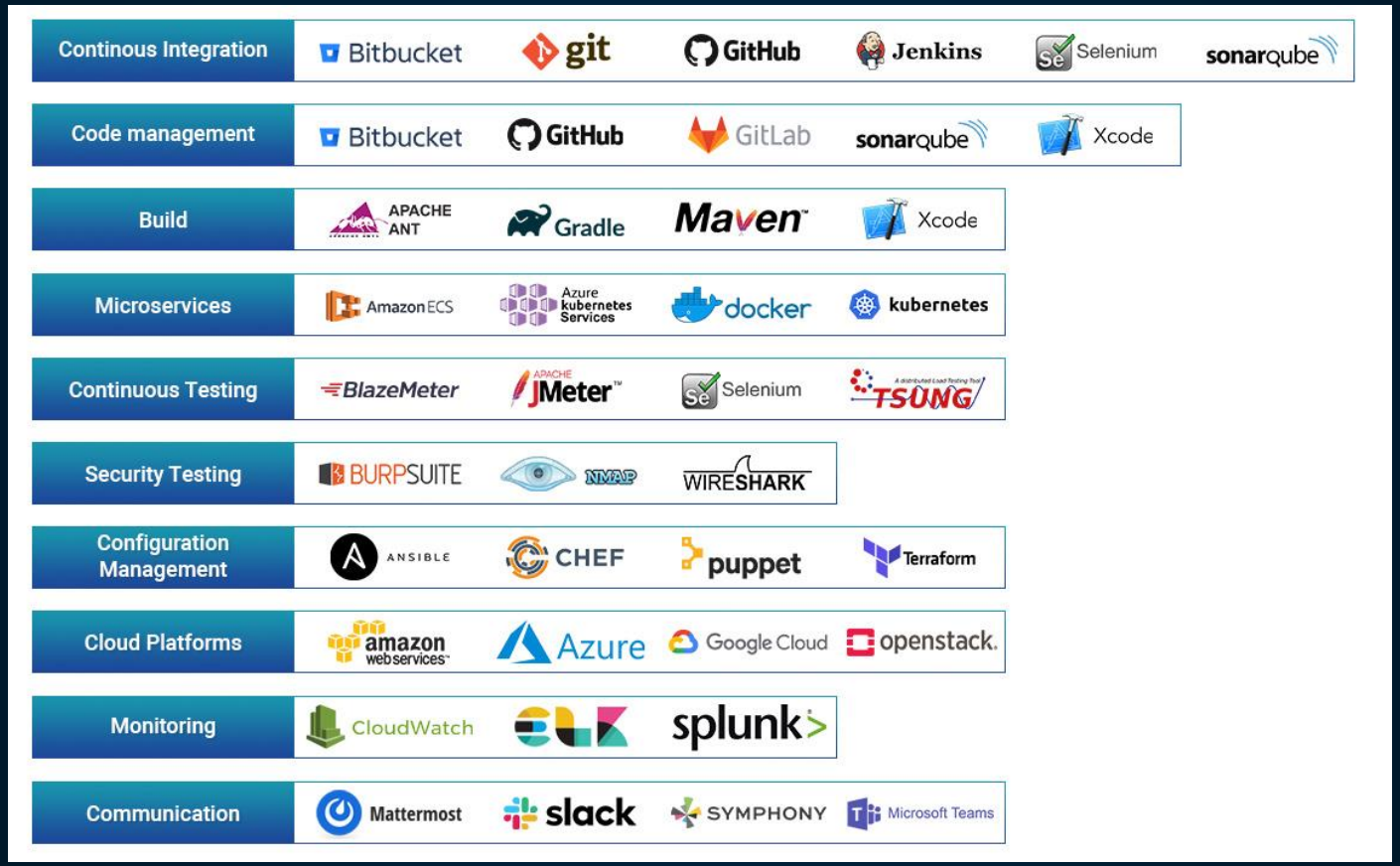
DevOps Technologies



DevOps – Extended Technologies

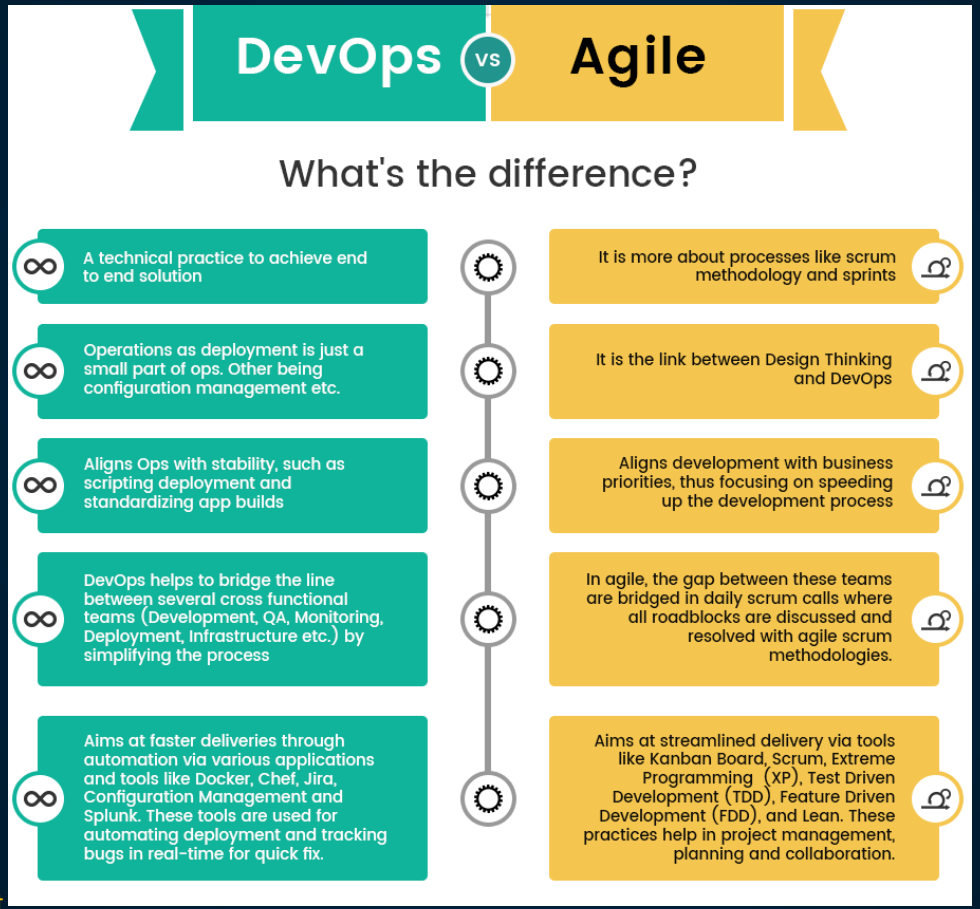


DevOps Technologies

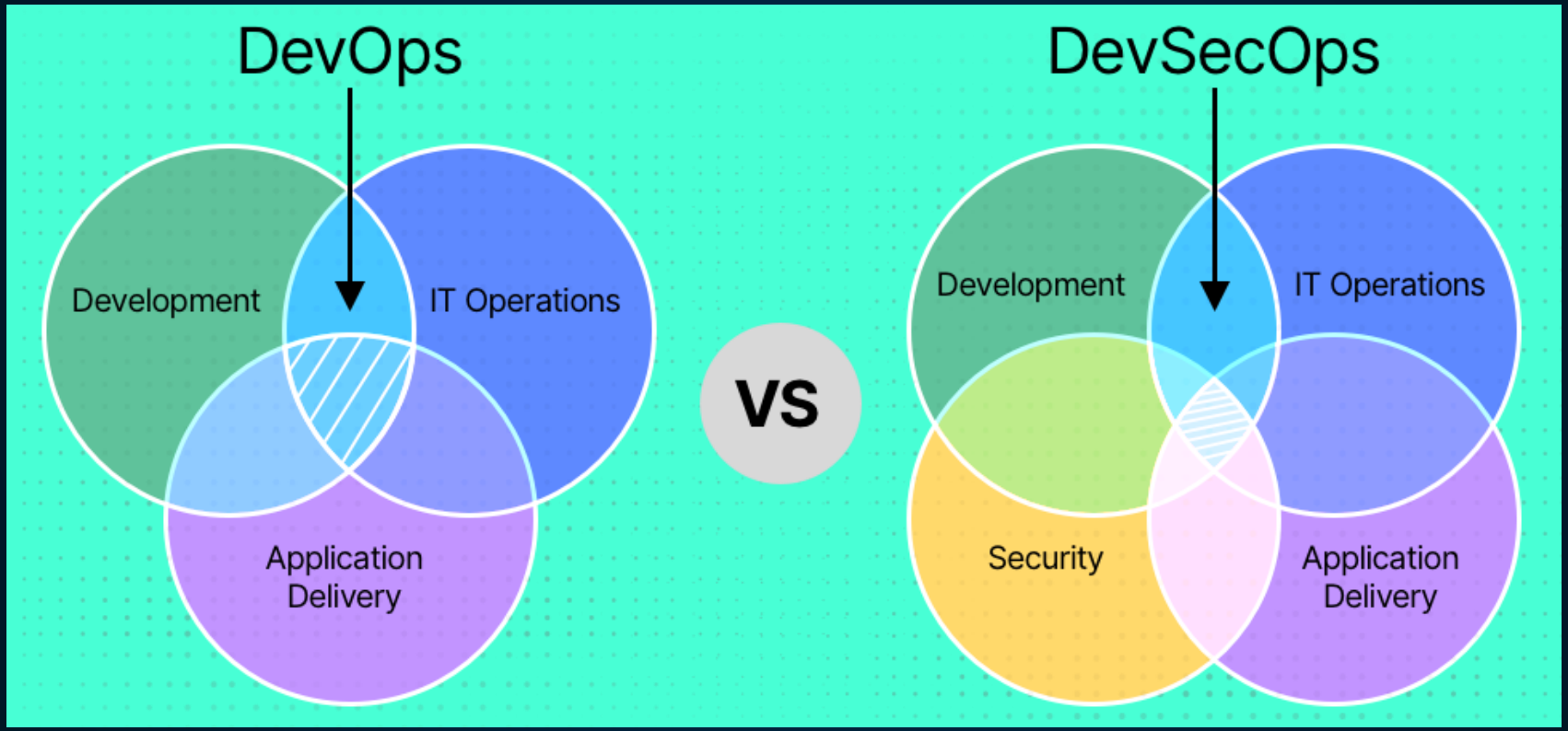


DevOps Tools

DevOps is not Agile!



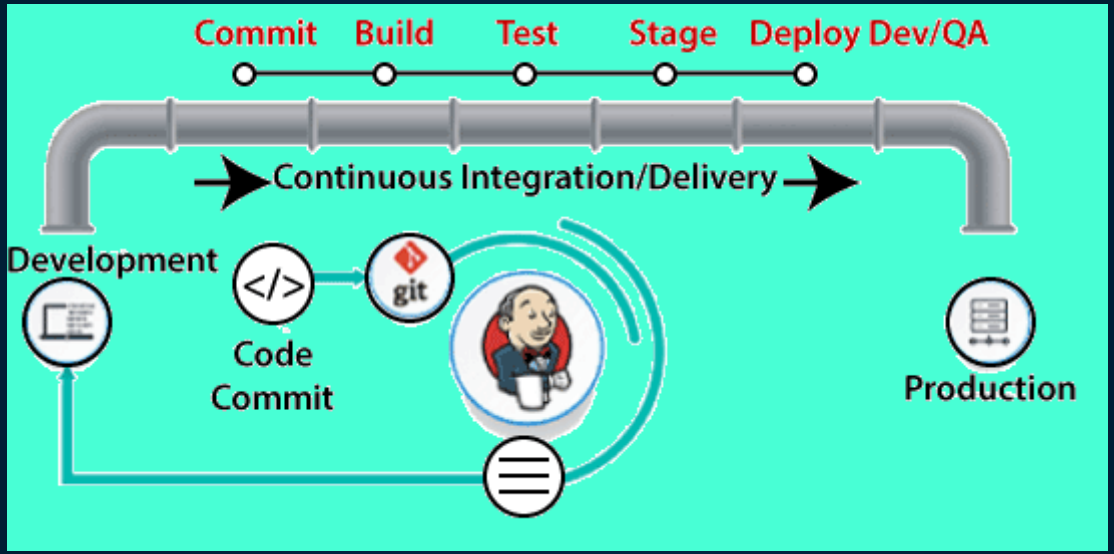
DevOps and DevSecOps



Continuous Things

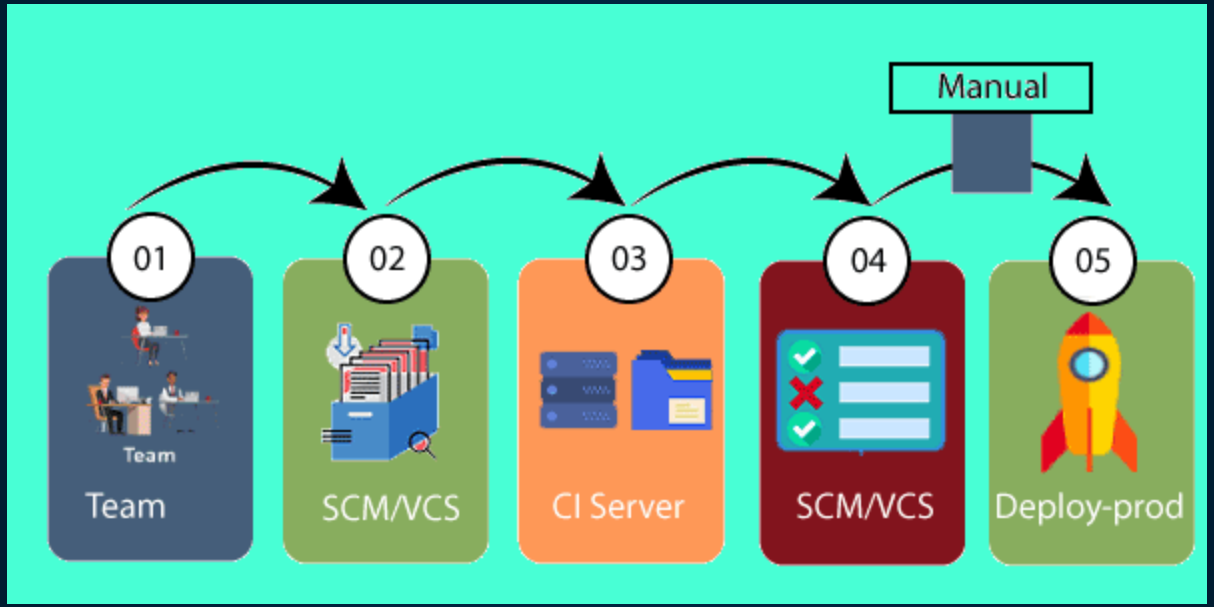
- Continuous Development
- Continuous Integration
- Continuous Testing
- Continuous Monitoring
- Continuous Feedback
- Continuous Deployment
- Continuous Operations

Continuous Integration

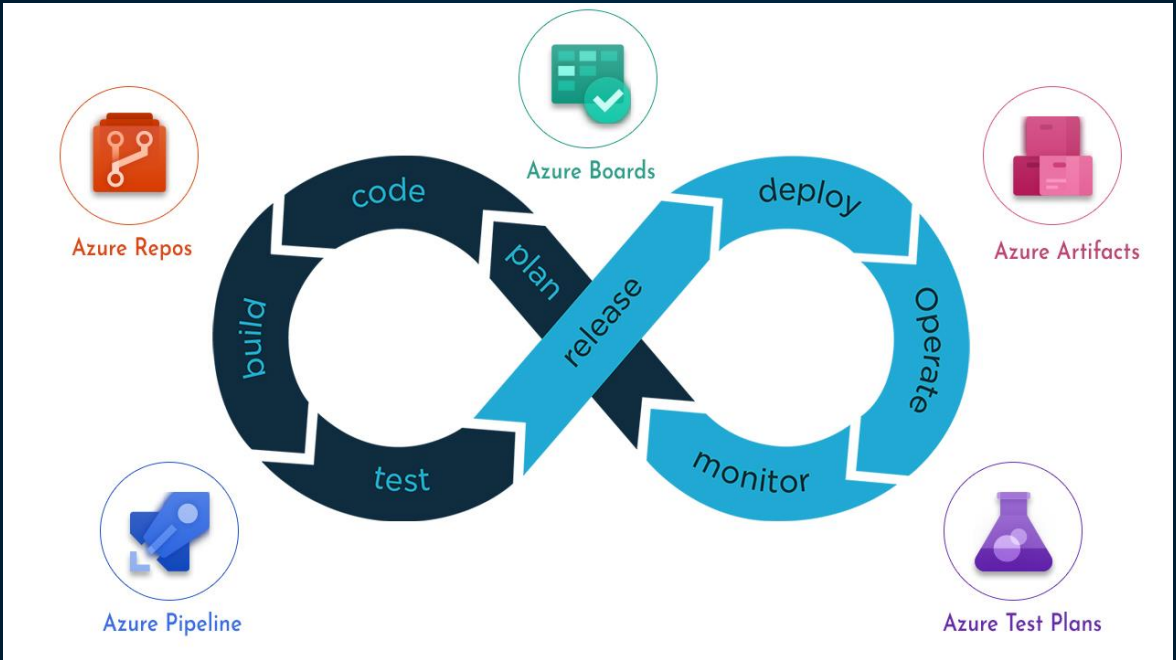


Continuous Testing

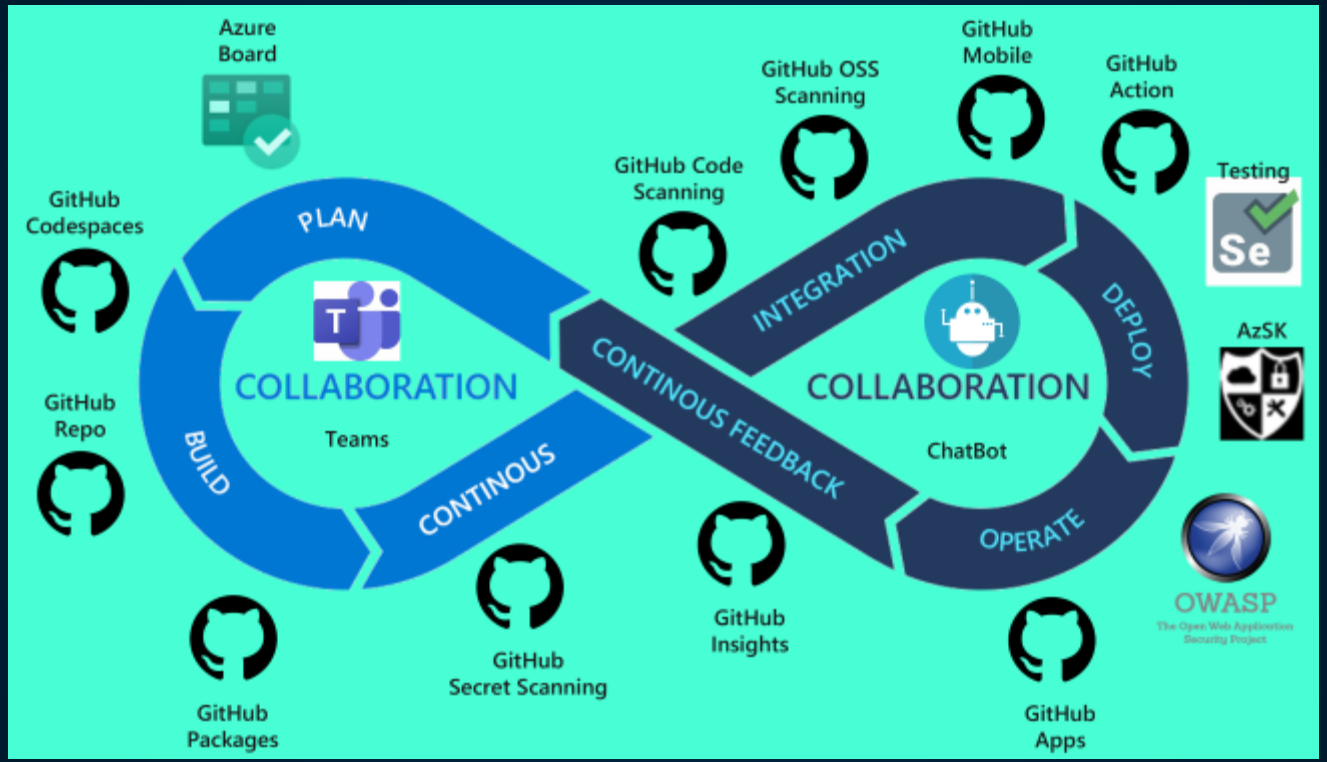
Continuous Deployment



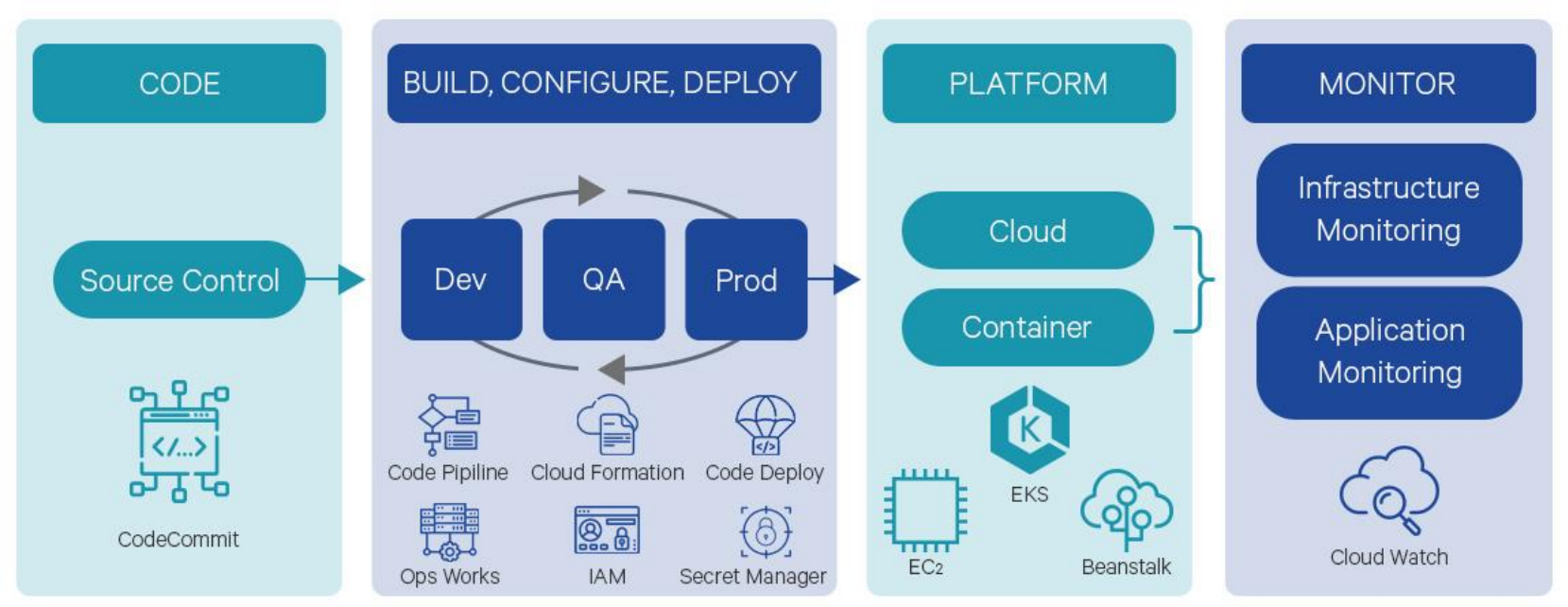
Azure DevOps



DevOps with GitHub



AWS DevOps


CODE
Source Control
CodeCommit
BUILD, CONFIGURE, DEPLOY
Dev
QA
Prod
Code Pipiline
Cloud Formation
Code Deploy
Ops Works
IAM
Secret Manager
PLATFORM
Cloud
Container
EKS
EC2
Beanstalk
MONITOR
Infrastructure Monitoring
Application Monitoring
Cloud Watch

DevOps: Advantages

- DevOps is an excellent approach for quick development and deployment of applications.
- It responds faster to the market changes to improve business growth.
- DevOps escalate business profit by decreasing software delivery time and transportation costs.
- DevOps clears the descriptive process, which gives clarity on product development and delivery.
- It improves customer experience and satisfaction.
- DevOps simplifies collaboration and places all tools in the cloud for customers to access.
- DevOps means collective responsibility, which leads to better team engagement and productivity.

DevOps: Disadvantages

- DevOps professional or expert's developers are less available.
- Developing with DevOps is so expensive.
- Adopting new DevOps technology into the industries is hard to manage in short time.
- Lack of DevOps knowledge can be a problem in the continuous integration of automation projects.

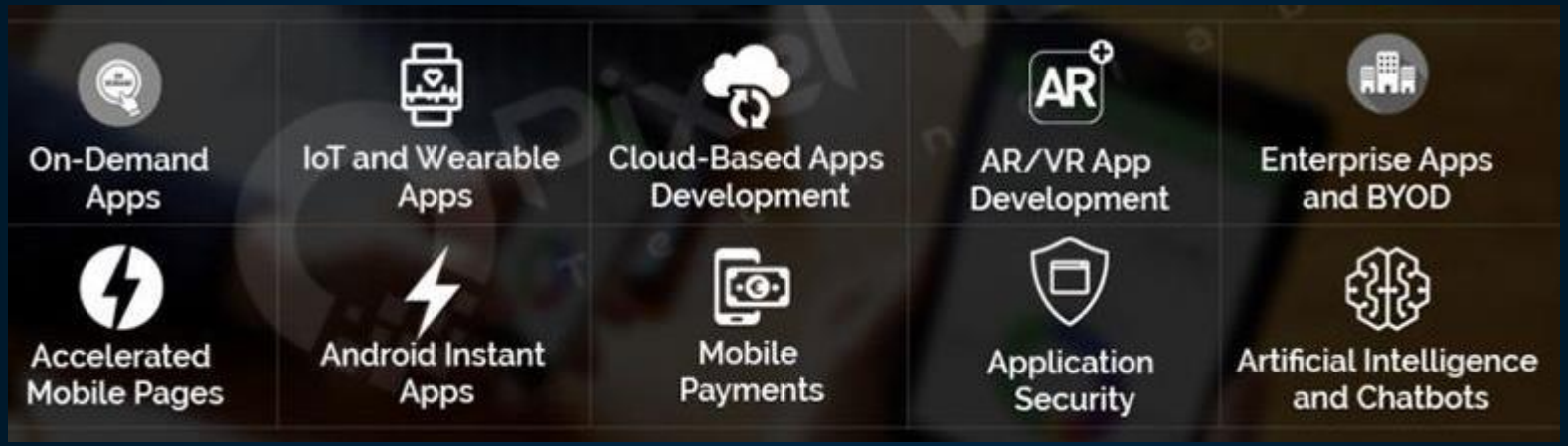
Mobile App Development Technologies

Unit 3.6

Dr. Mahesha BR Pandit

04-Dec-21 Version 1.0

Mobile App Development Trends



Choices and comparison

Decision Criterion	Native Approach	Cross Platform Approach
Quality of UX	Excellent	Not as good as native apps
Quality of apps	High	Medium to low
Potential users	Limited to a particular mobile platform	Large - as it reaches to users of different platforms
App development cost	High	Medium to low
Time-to-market	High	Short

Native App Development

- Most wished
- Mobile application fits only a particular and single platform.
- Use programming language native to the operating system.
- Java or Kotlin for Android, and Swift/Objective-C for iOS.
- Superior & rich performance user-experience that can be tailored to a single platform.
- Expensive
- At least two code bases

Native


Native App Development Tools
XCode
APPCode
Android Studio

Xcode

- Xcode acquaints a way to design and develop mobile applications.
- Swift is a creative new programming language for Cocoa and Cocoa Touch and, when joined with Xcode tools, makes programming a superbly live encounter.
- Live rendering inside Interface Builder shows your manually written UI code inside the plan canvas, in a flash reflecting changes you type in code.
- Xcode incorporates everything developers need to make applications for Mac, iPhone, iPad, Apple TV, and Apple Watch.
- Xcode gives developers a brought together work process for UI configuration, coding, testing, and troubleshooting.

AppCode

- AppCode is an IDE for iOS/macOS advancement.
- Notwithstanding working with Objective-C, Swift and C/C++ programming dialects, it underpins web innovations, for example, JavaScript, HTML, XML, CSS, and then some.
- It gives an assortment of significant incorporations including among different CocoaPods director and implicit Reveal support.
- AppCode provides the developers with many features such as sparing their time on automating routine tasks, finding and fixing mistakes, exploiting advantages of the IDE, and expanding their general efficiency, it may be a similarly important resource for the business.

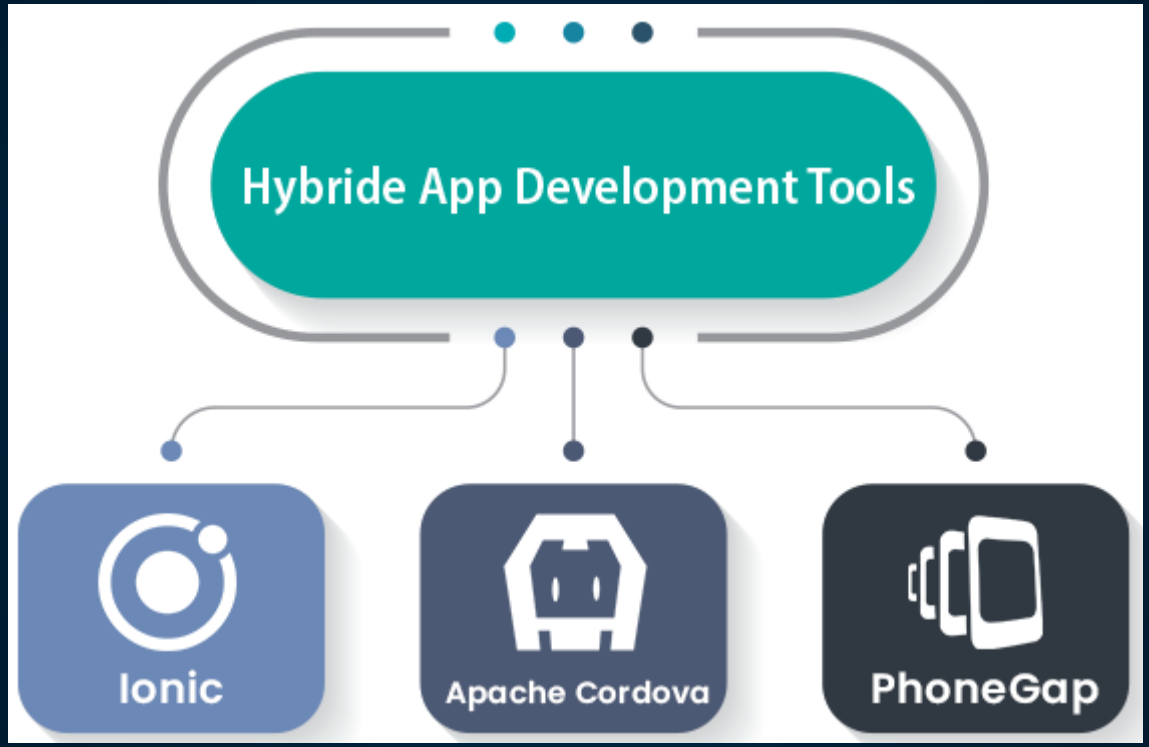
Android Studio

- Android Studio is an Android development Software developed by Google.
- Its implementation editor is extremely valuable for Android developers.
- Android studio gives alternate ways to coding and planning and its format architect makes it simple to utilize, which diminishes time spent on coding.
- Android Studio likewise gives intuitive highlights to plan the design of your projects.

Hybrid App Development

- Better than Native Applications in terms of user experience.
- Requires less time for development
- Code sharing
- Imperfect user experience.
- Combination of web components with mobile-based components.
- Wrap web component inside a native container – the WebView.
- The content within WebView is rendered as a plain website.
- Difficult to access phone features like camera, GPS, and so on

Hybrid


Hybride App Development Tools
Ionic
Apache Cordova
PhoneGap

Ionic

- Ionic is an easily adaptable and usable mobile app development tool that takes into account quick prototyping with an interactive CLI.
- Coordination with Angular makes for a pleasurable domain to code in.
- Utilizing standard web technologies, Ionic assists communities to develop and deliver effective cross-platform hybrid and progressive web apps.
- The Ionic framework is a free, open-source mobile UI software development framework used for developing cross-platform applications for native iOS, Android, and the web—all from a single codebase.

Cordova

- Apache Cordova is an open-source mobile app development tool that empowers web developers to utilize their HTML, CSS, and JavaScript substance to make a native application for an assortment of mobile platforms.
- Cordova takes your web application and renders it inside a native WebView.
- A WebView is an application part (like a catch or a tab bar) that is utilized to show web content inside a native application.
- You can think about a WebView as an internet browser with no of the standard UI components, for example, a URL field or status bar.

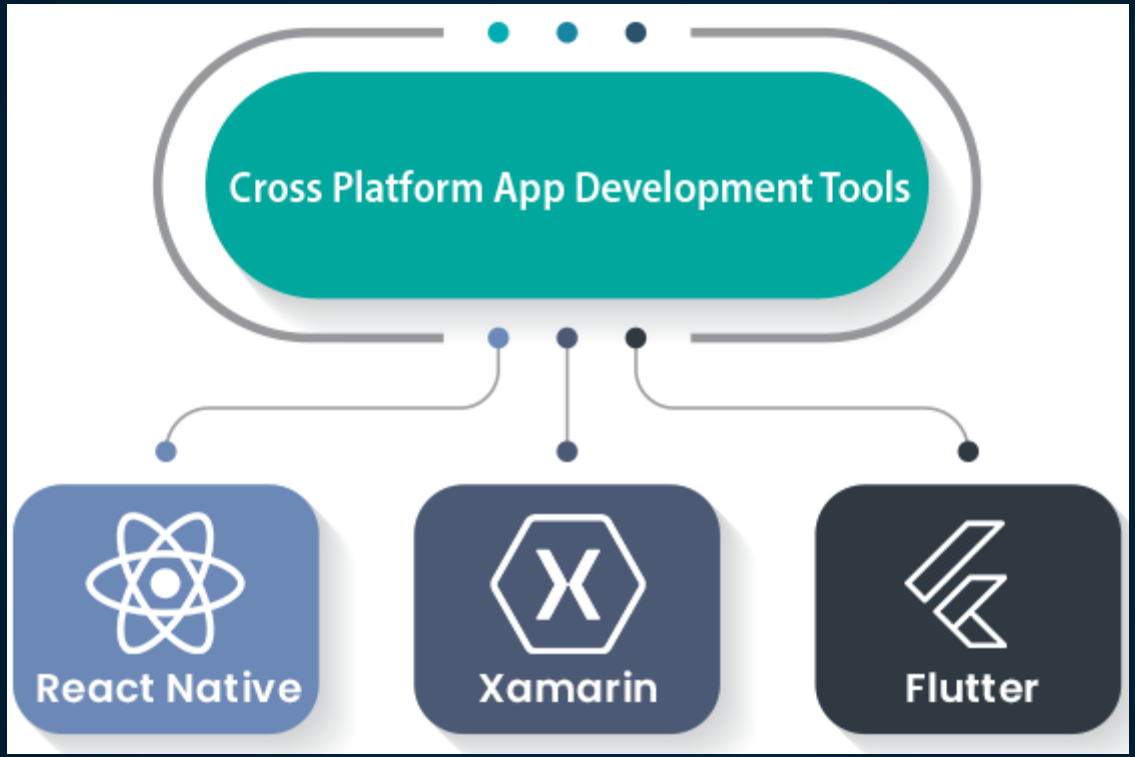
PhoneGap

- Sponsored by Adobe, PhoneGap is a distribution of Cordova systems.
- This open-source mobile application development structure is normally viewed as the top and the most mainstream tool for Hybrid solutions not least in light of its usability.
- The device permits composing applications in HTML5, CSS3 and JavaScript. For every mobile platform, it gives native modules and containers, empowering developers to make more highlights and to get to gadget usefulness, for example, camera, microphone, accelerometer, compass, file management system and a lot more.
- After incorporating these features, applications can effectively be executed in a WebView program inside a native compartment on the proper platform.

Cross Platform App Development

- Hybrid and cross-platform applications are not like each other.
- A common & fundamental component between both types of applications is of their code shareability.
- Fast development
- Cross-platform dev approach utilizes a native rendering engine.
- The codebase written in JS interfaces with native segments through the bridges. This gives the close-to-native UX.
- Cross-platform applications are liberated from platform hooks.
- Consistent usefulness, simple execution, and cheaper solution. However, performance is lower than native apps.
- Needs a lot of customization

Cross Platform



ReactNative

- Created by the Facebook community, React Native is a cross-platform system dependent on JavaScript innovation.
- It empowers developers to develop effective and native-like applications utilizing a single codebase.
- The point of this system is to assemble top-notch native application encounters utilizing JavaScript and React.
- This exceptional JavaScript innovation makes it conceivable to build up a cross-platform mobile application that closely resembles a native one however it is coded by a team of designers just utilizing React and JavaScript.

Xamarin

- Xamarin is a mobile application development platform that helps architects and designers fabricate native iOS, Android and Windows applications utilizing a single shared .NET codebase.
- An Integrated Development Environment (IDE), Xamarin uses Visual Studio Tools to create native mobile applications.
- The platform offers a large group of highlights including code altering, refactoring, debugging, testing and cloud-distributing.
- Moreover, the platform gives access to on-request content from Xamarin’s college educational plan and month to month Azure credit.

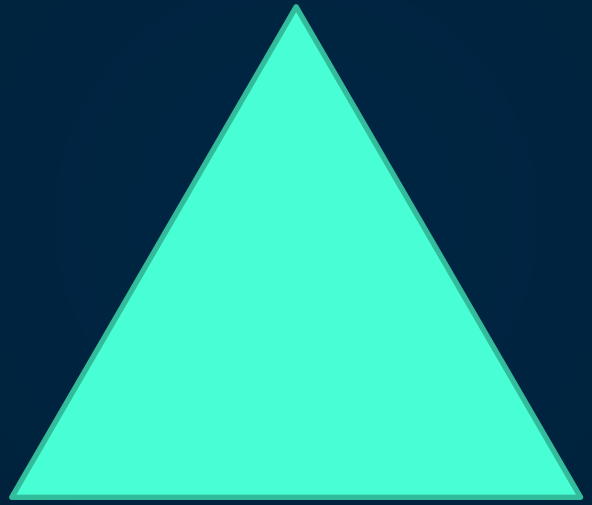
Flutter

- Flutter is a software development unit (SDK) for mobile applications, created by Google.
- It has been created to create mobile applications for the major application development platforms, similar to Android and iOS.
- As an open-source structure, Flutter is free and it offers a total bundle of development tools, features and systems.
- This empowers the application designers to show signs of development experience, backing off the procedure.
- Despite the fact that Flutter is a nearly new framework, organizations like Tencent and Alibaba have just utilized this framework in their processes.
- Moreover, Flutter is being utilized by Google in the application called ‘Google Ads’.
- In this manner, Flutter has just demonstrated its significance in creating native-like applications on Android and iOS, utilizing a similar codebase.

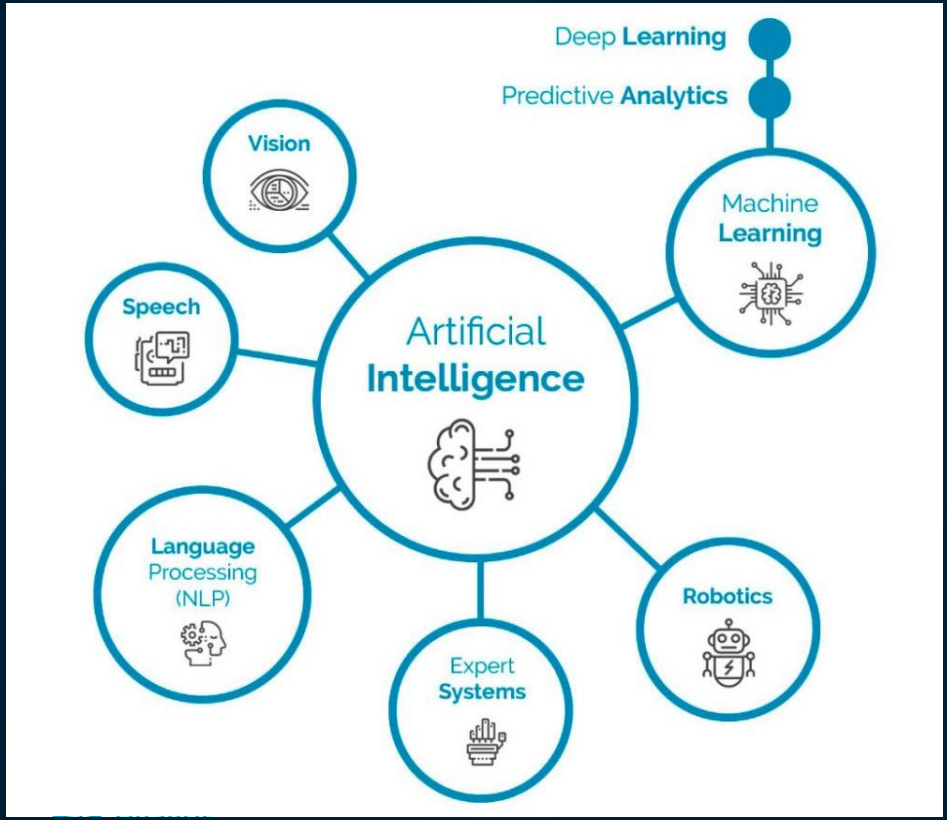
AI and ML Technologies

Unit 3.7
Dr. Mahesha BR Pandit
07-Dec-21 Version 1.0

Three Points of AI



Use Case Classes



Algorithms: Minimum Set

1. Classification Algorithms

a) Naive Bayes

b) Decision Tree

c) Random Forest

d) Support Vector Machines

e) K Nearest Neighbours

2. Regression Algorithms

a) Linear regression

b) Lasso Regression

c) Logistic Regression

d) Multivariate Regression

e) Multiple Regression Algorithm

3. Clustering Algorithms

a) K-Means Clustering

b) Fuzzy C-means Algorithm

c) Expectation-Maximisation (EM) Algorithm

d) Hierarchical Clustering Algorithm

Classification Algorithms – 1

- Implement supervised learning. Divide the subjected variable into different classes and then predict the class for a given input. For example: Spam or Not, Defect or Not etc.
- Naive Bayes algorithm works on Bayes theorem and takes a probabilistic approach. The algorithm has a set of prior probabilities for each class. Computes posterior probability to predict whether the input belongs to a given list of classes or not.
- The decision tree algorithm is more of a flowchart like an algorithm where nodes represent the test on an input attribute and branches represent the outcome of the test.
- Random forest works like a group of trees. The input data set is subdivided and fed into different decision trees. The average of outputs from all decision trees is considered. Random forests offer a more accurate classifier as compared to Decision tree algorithm.

Classification Algorithms – 2

- Support Vector Machines algorithm classifies data using a hyperplane, making sure that the distance between the hyperplane and support vectors is maximum.
- K Nearest Neighbours algorithm uses a bunch of data points segregated into classes to predict the class of a new sample data point. It is called “lazy learning algorithm” as it is relatively short as compared to other algorithms.

Regression Algorithms – 1

- Supervised machine learning algorithms.
- Can predict the output values based on input data points fed in the learning system.
- The main application of regression algorithms includes predicting stock market price, predicting weather, etc.
- Linear regression: Simplest of all regression algorithms but can be implemented only in cases of linear relationship or a linearly separable problem. Draws a straight line between data points called the best-fit line or regression line and is used to predict new values.
- Lasso regression algorithm works by obtaining the subset of predictors that minimizes prediction error for a response variable. This is achieved by imposing a constraint on data points and allowing some of them to shrink to zero value.

Regression Algorithms – 2

- Logistic regression is mainly used for binary classification. This method allows you to analyze a set of variables and predict a categorical outcome. Its primary applications include predicting customer lifetime value, house values, etc
- Multivariate Regression algorithm has to be used when there is more than one predictor variable. This algorithm is extensively used in retail sector product recommendation engines, where customers preferred products will depend on multiple factors like brand, quality, price, review etc.
- Multiple Regression Algorithm uses a combination of linear regression and non-linear regression algorithms taking multiple explanatory variables as inputs. The main applications include social science research, insurance claim genuineness, behavioural analysis, etc.

Clustering Algorithms – 1

- Clustering is the process of segregating and organizing the data points into groups based on similarities within members of the group.
- This is part of unsupervised learning.
- The main aim is to group similar items.
- For example, it can arrange all transactions of fraudulent nature together based on some properties in the transaction.
- K-Means Clustering: Simplest unsupervised learning algorithm. The algorithm gathers similar data points together and then binds them together into a cluster. The clustering is done by calculating the centroid of the group of data points and then evaluating the distance of each data point from the centroid of the cluster. Based on the distance, the analyzed data point is then assigned to the closest cluster. ‘K’ in K-means stands for the number of clusters the data points are being grouped into.

Clustering Algorithms – 2

- Fuzzy C-means Algorithm works on probability. Each data point is considered to have a probability of belonging to another cluster. Data points don't have an absolute membership over a particular cluster, and this is why the algorithm is called fuzzy.
- Expectation-Maximisation (EM) Algorithm: It is based on Gaussian distribution. Data is pictured into a Gaussian distribution model to solve the problem. After assigning a probability, a point sample is calculated based on expectation and maximization equations.
- Hierarchical Clustering Algorithms sort clusters hierarchical order after learning the data points and making similarity observations. It can be of two types Divisive clustering, for a top-down approach and Agglomerative clustering, for a bottom-up approach

Python Libraries – 1

- Numpy: Handles large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.
- SciPy is a very popular library among Machine Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics. There is a difference between the SciPy library and the SciPy stack. The SciPy is one of the core packages that make up the SciPy stack. SciPy is also very useful for image manipulation.
- Scikit-learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data-mining and data-analysis, which makes it a great tool who is starting out with ML.

Python Libraries – 2

- Theano is a popular python library that is used to define, evaluate and optimize mathematical expressions involving multi-dimensional arrays in an efficient manner. It is achieved by optimizing the utilization of CPU and GPU. It is extensively used for unit-testing and self-verification to detect and diagnose different types of errors. Theano is a very powerful library that has been used in large-scale computationally intensive scientific projects for a long time but is simple and approachable enough to be used by individuals for their own projects.
- TensorFlow is a very popular open-source library for high performance numerical computation developed by the Google Brain team in Google. Tensorflow is a framework that involves defining and running computations involving tensors. It can train and run deep neural networks that can be used to develop several AI applications. TensorFlow is widely used in the field of deep learning research and application.

Python Libraries – 3

- Keras is a very popular Machine Learning library for Python. It is a high-level neural networks API capable of running on top of TensorFlow, CNTK, or Theano. It can run seamlessly on both CPU and GPU. Keras makes it really for ML beginners to build and design a Neural Network. One of the best thing about Keras is that it allows for easy and fast prototyping.
- PyTorch is a popular open-source Machine Learning library for Python based on Torch, which is an open-source Machine Learning library which is implemented in C with a wrapper in Lua. It has an extensive choice of tools and libraries that supports on Computer Vision, Natural Language Processing(NLP) and many more ML programs. It allows developers to perform computations on Tensors with GPU acceleration and also helps in creating computational graphs.

Python Libraries – 4

- Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for groping, combining and filtering data.
- Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar chats, etc,

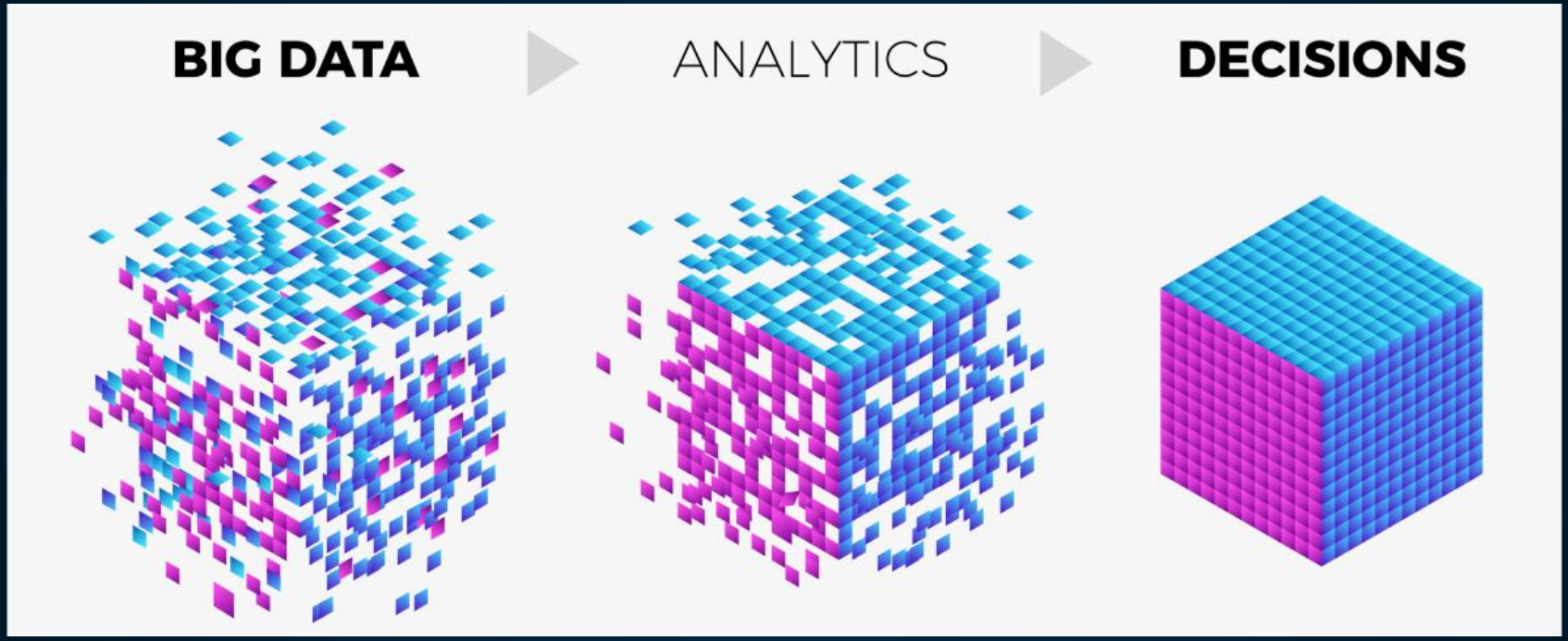
Data Analytics Technologies

Unit 3.8

Dr. Mahesha BR Pandit

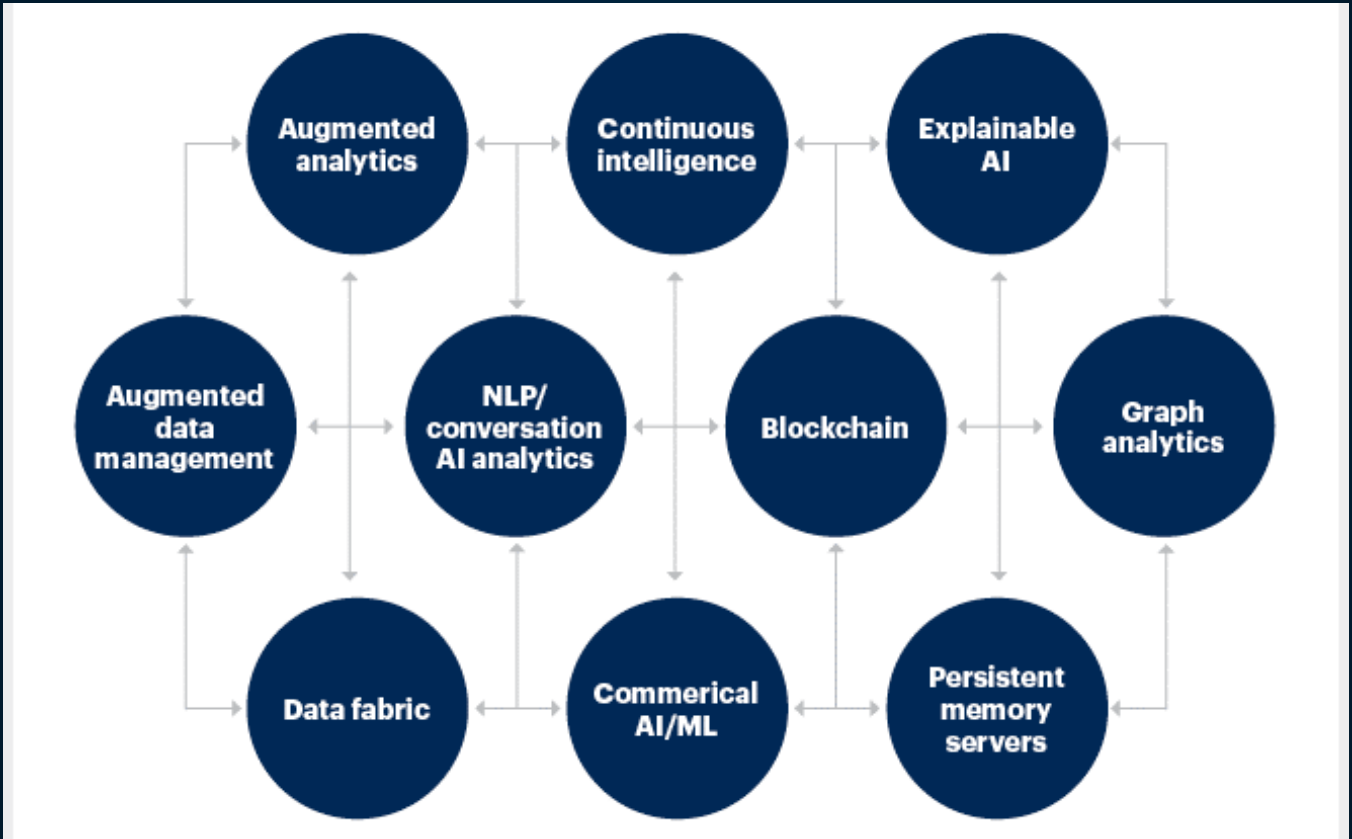
11-Dec-21 Version 1.0

What is What?



Trends – as in 2019

Top 10 Trends – Gartner, 2020



Five Key Focus Areas

DATA SOURCES

Internal data sources such as data from CRM system, ERP system, sales reports, etc.

External data sources such as government statistics and media channels

DATA STORAGE

Big data storage software tools store, manage and retrieve massive amounts of data.

APACHE hadoop

mongoDB

cassandra

DATA MINING

Data mining tools allow businesses to extract usable data from a huge set of raw data to find relationships, patterns, and anomalies.

rapidminer

SPSS Modeler

DATA ANALYTICS

Although data mining tools incorporate data analysis, there are software designed specifically with advanced analytical capabilities.

APACHE Spark

kafka

IBM Analytics

DATA VISUALIZATION

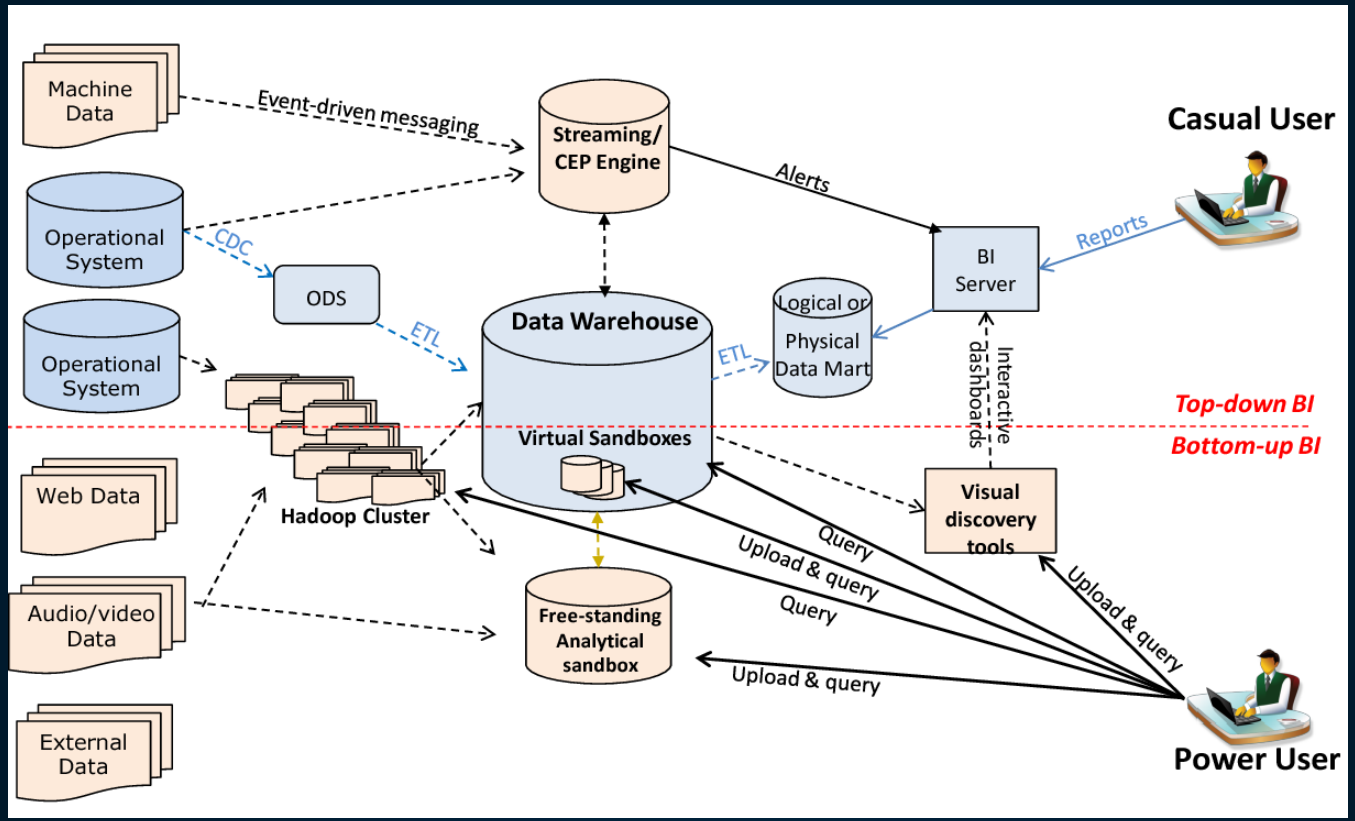
Data visualization software is also a type of data analytics tool. However, they are specifically designed to take the raw data and presenting it with beautiful and easy digestible visuals like graphs and charts.

+ a b | e a u

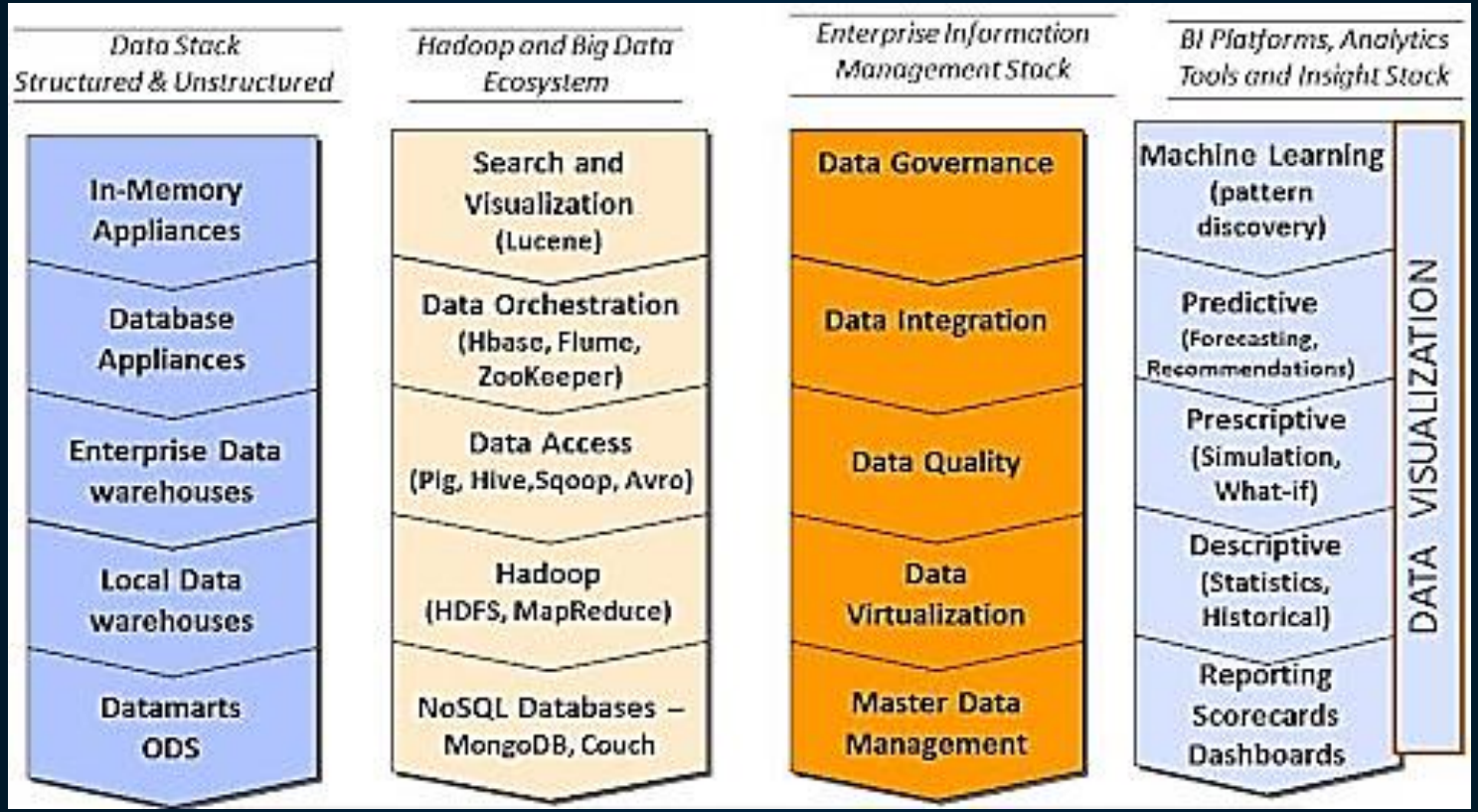
looker

Klipfolio

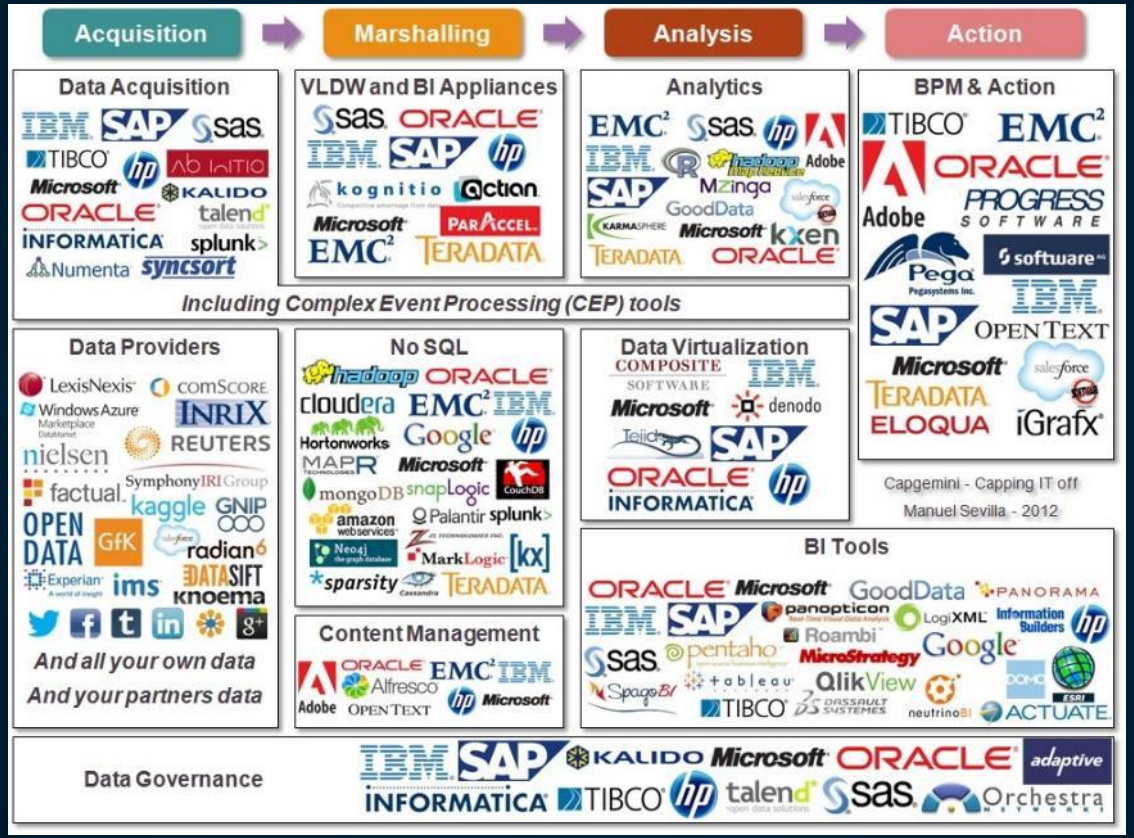
Technology Providers



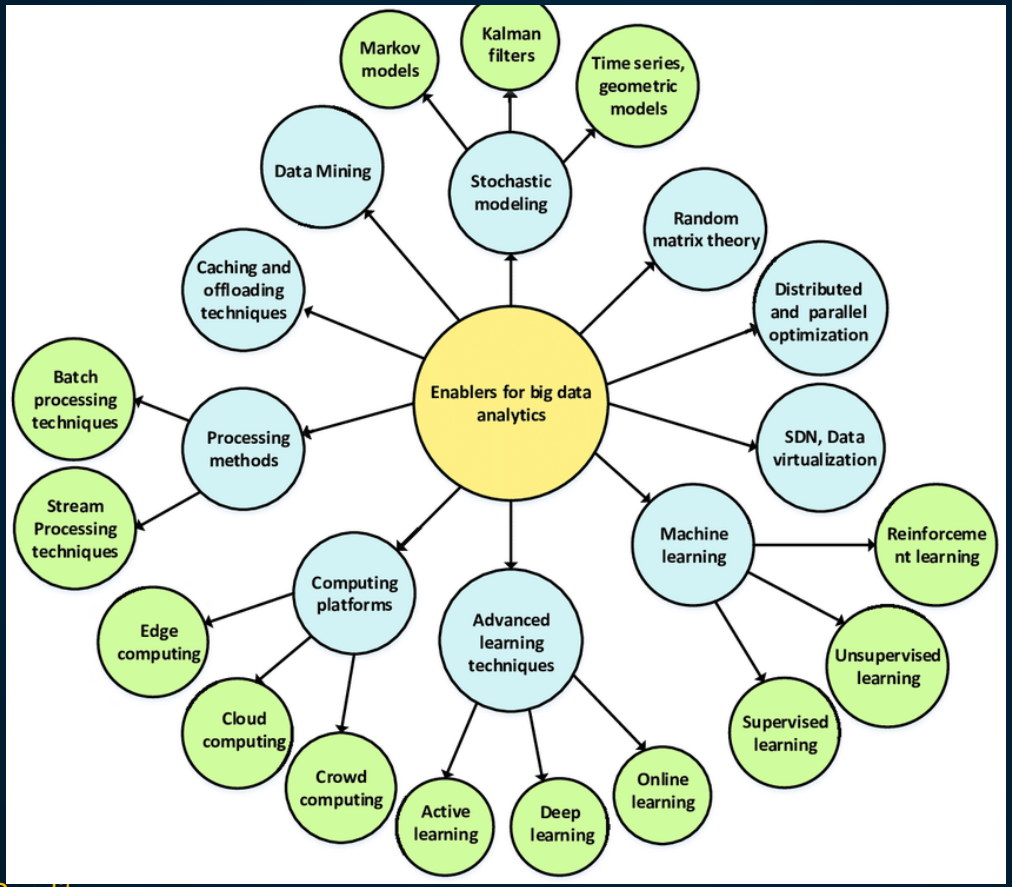
Infrastructure of Big Data



Technology Providers



What enables Data Analytics?



Top 10 Technologies for Data Analytics

- R and Python
- Microsoft Excel
- Tableau
- RapidMiner
- KNIME
- Power BI
- Apache Spark
- QlikView
- Talend
- Splunk

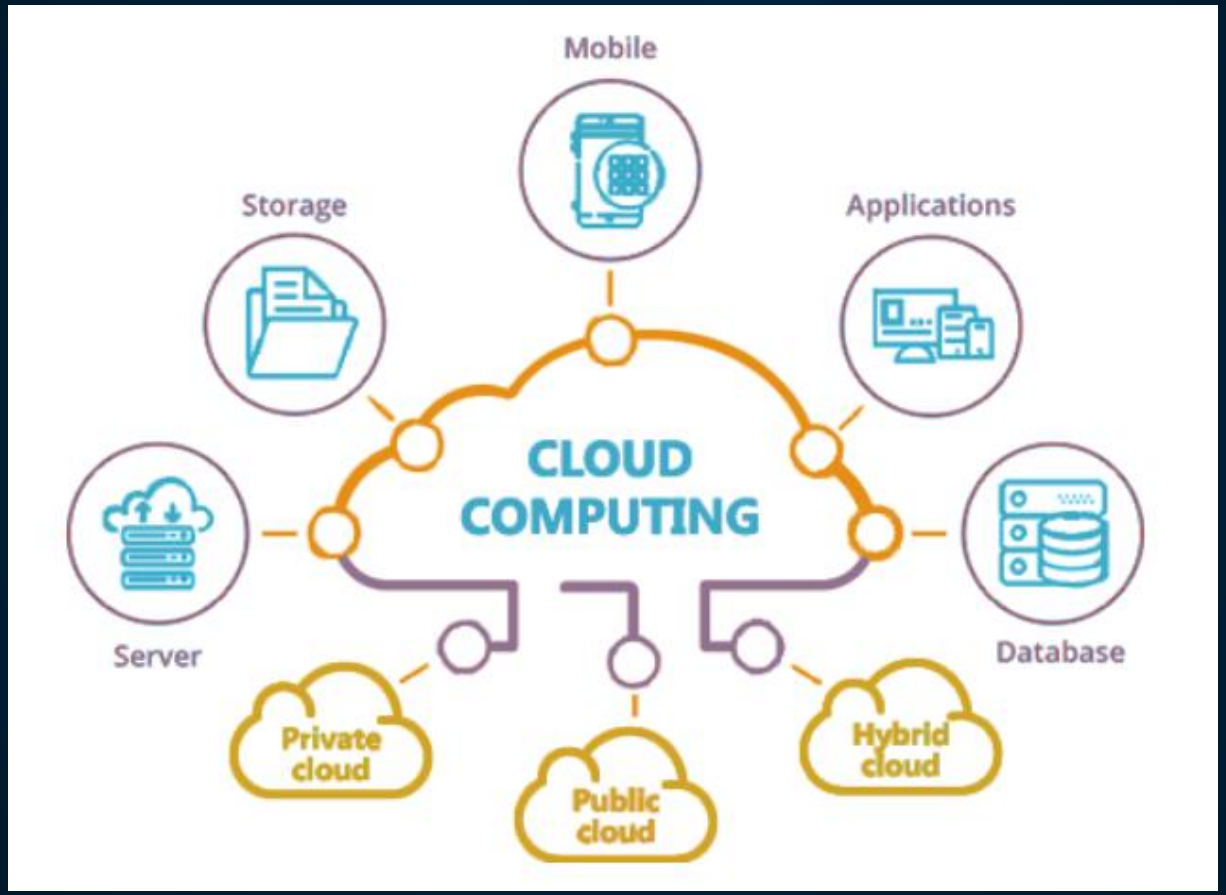
Cloud Computing Technologies

Unit 3.9

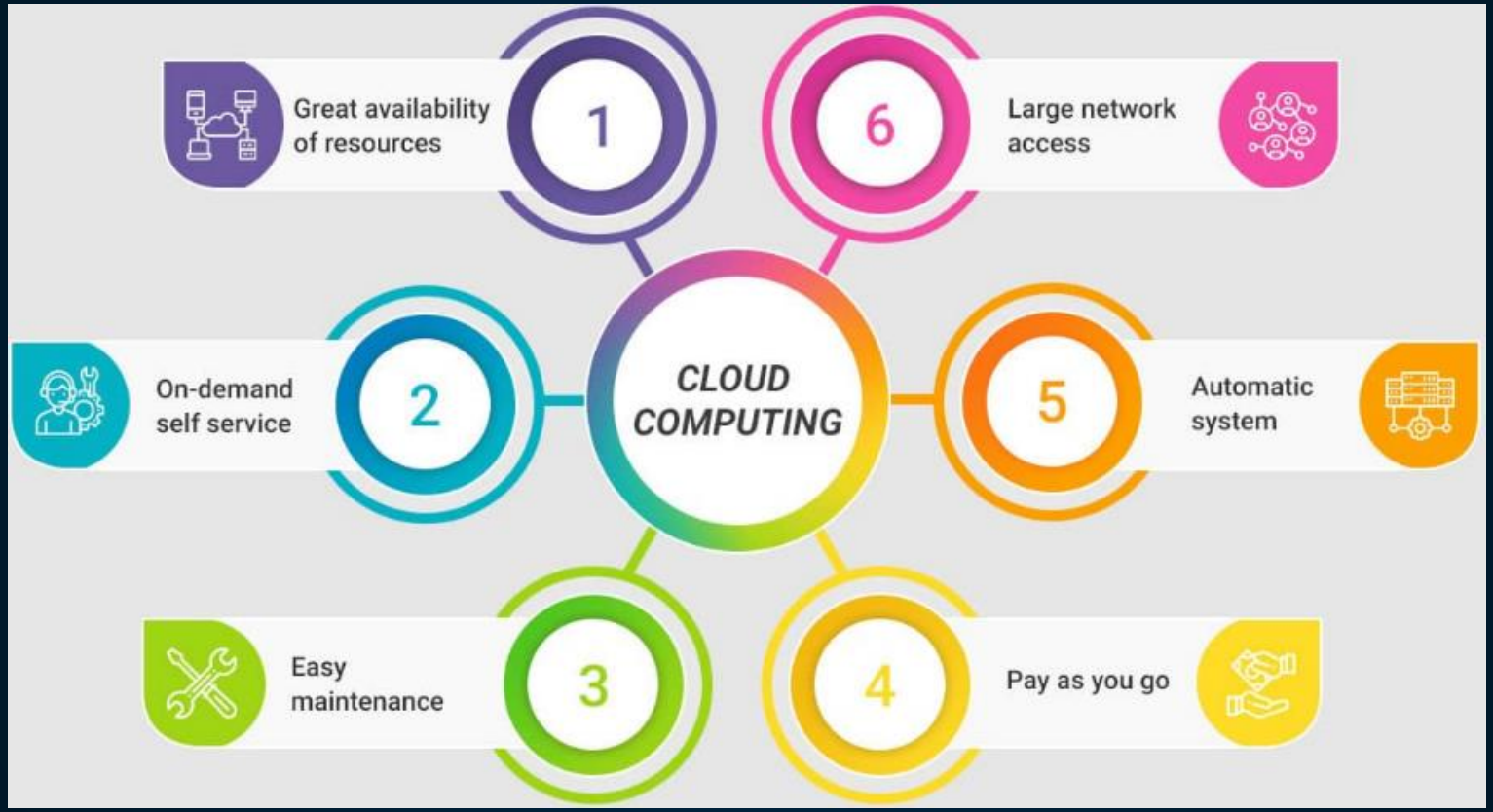
Dr. Mahesha BR Pandit

11-Dec-21 Version 1.0

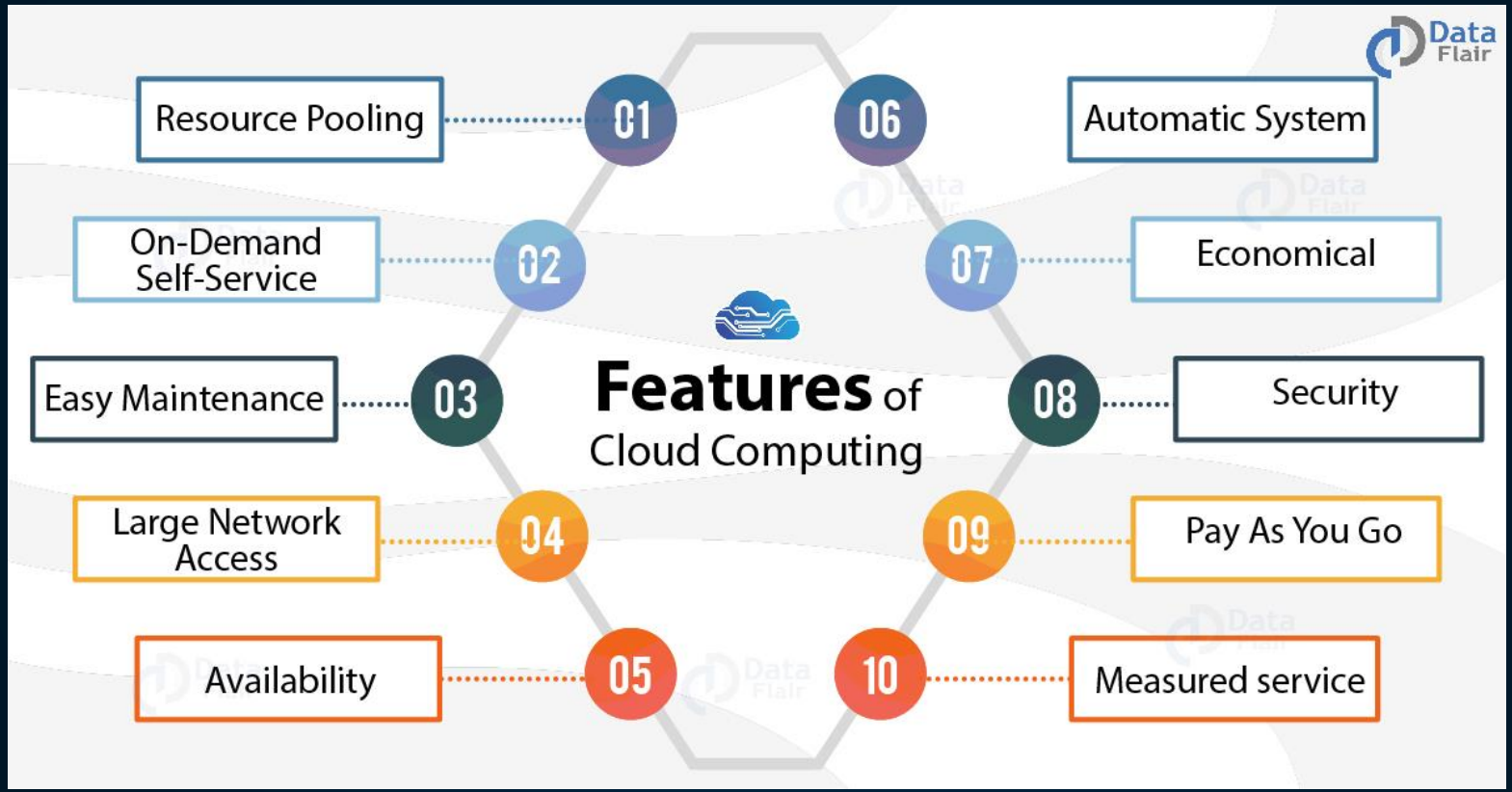
The Cloud


Mobile
Storage
Applications
CLOUD
COMPUTING
Server
Database
Private cloud
Public cloud
Hybrid cloud

Features of Cloud Computing

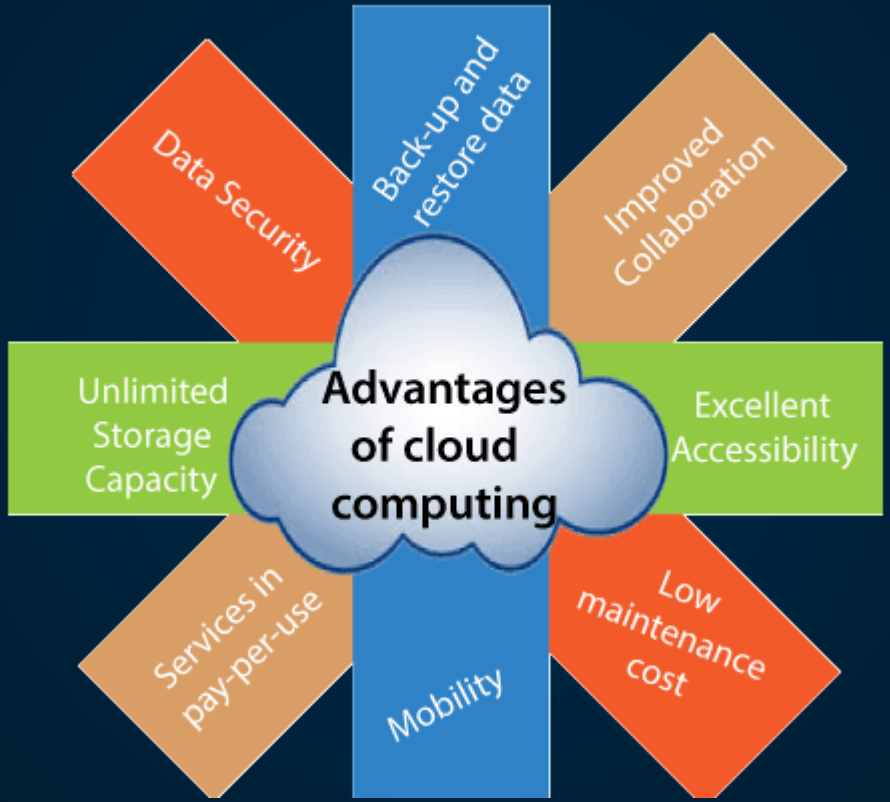


Features of Cloud Computing

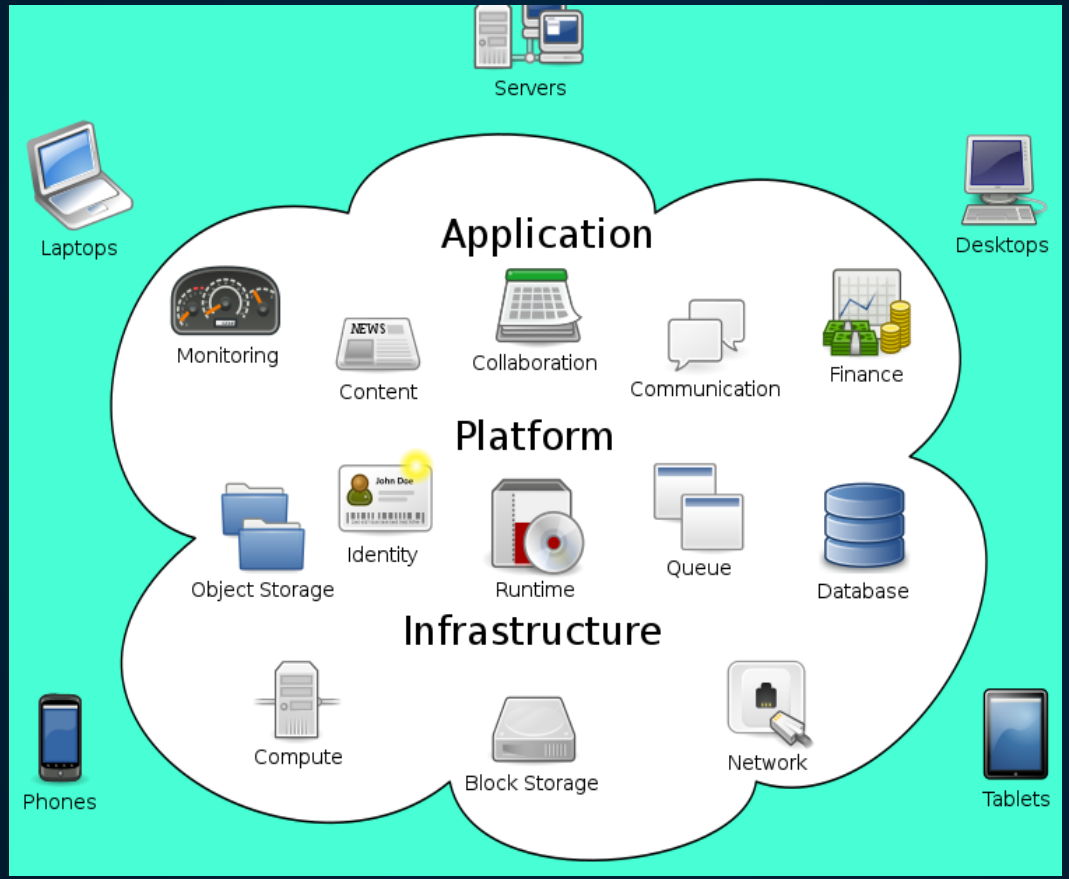


https://data-flair.training/blogs/features-of-cloud-computing/

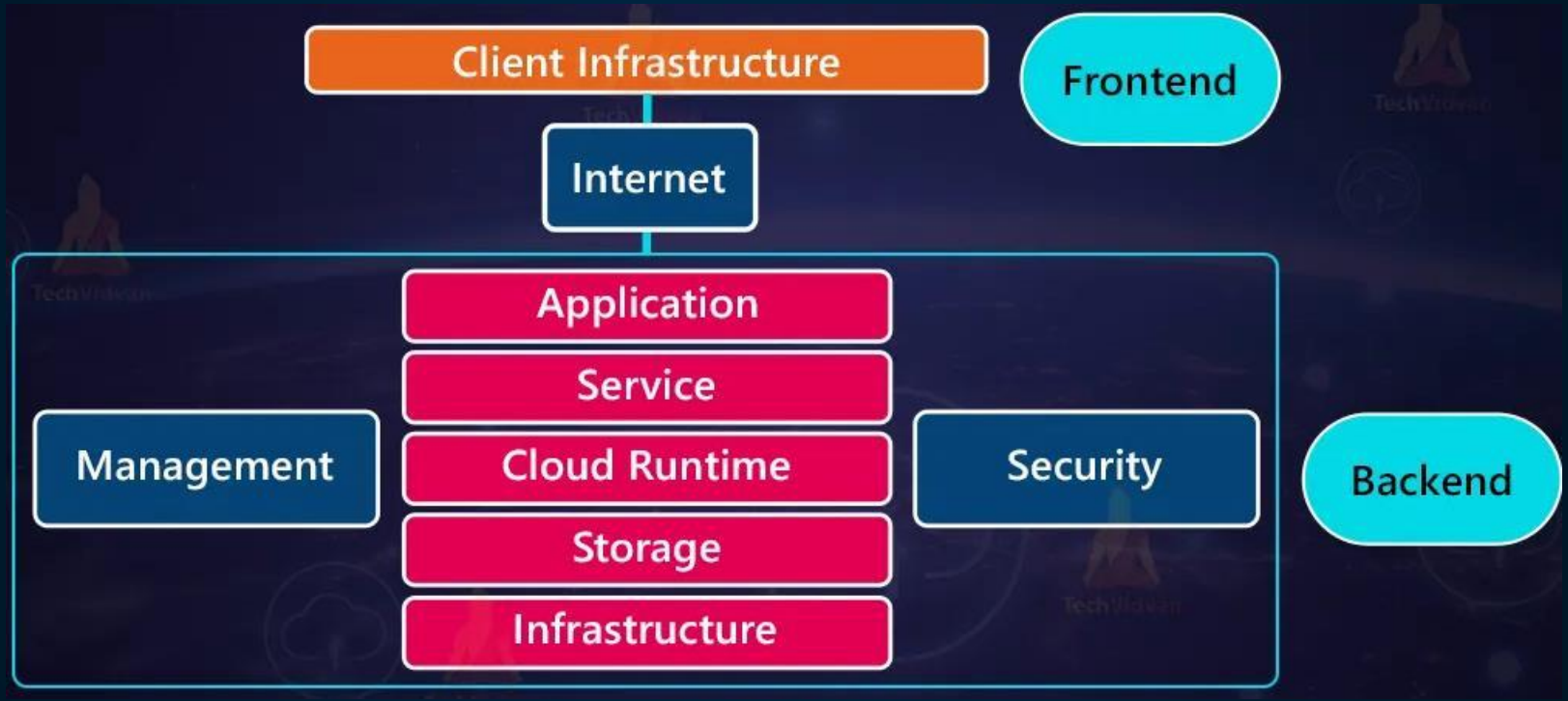
Advantages of Cloud Computing



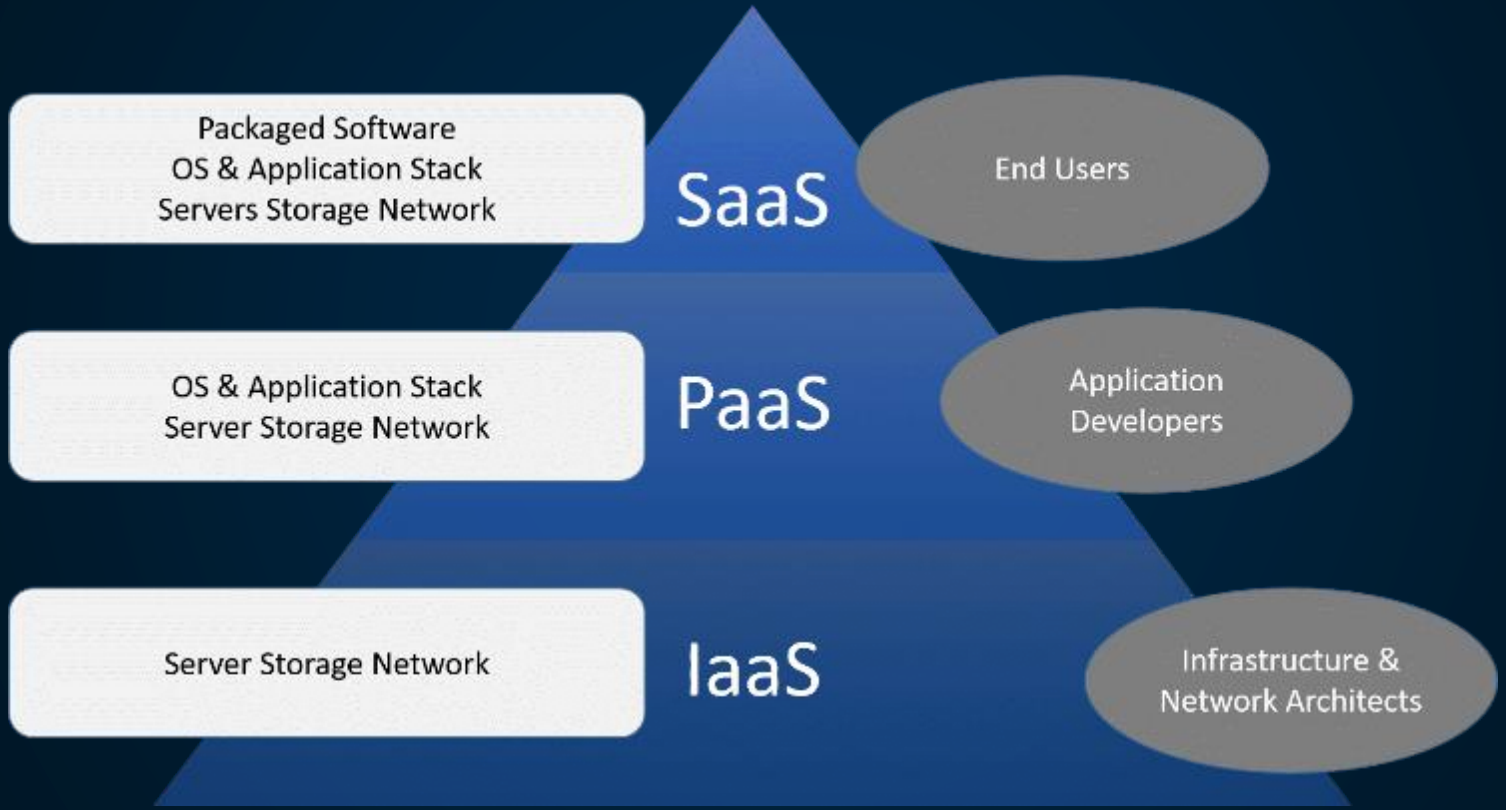
What is What?


Servers
Laptops
Desktops
Application
Monitoring
Content
Collaboration
Communication
Finance
Platform
Object Storage
Identity
Runtime
Queue
Database
Infrastructure
Compute
Block Storage
Network
Phones
Tablets

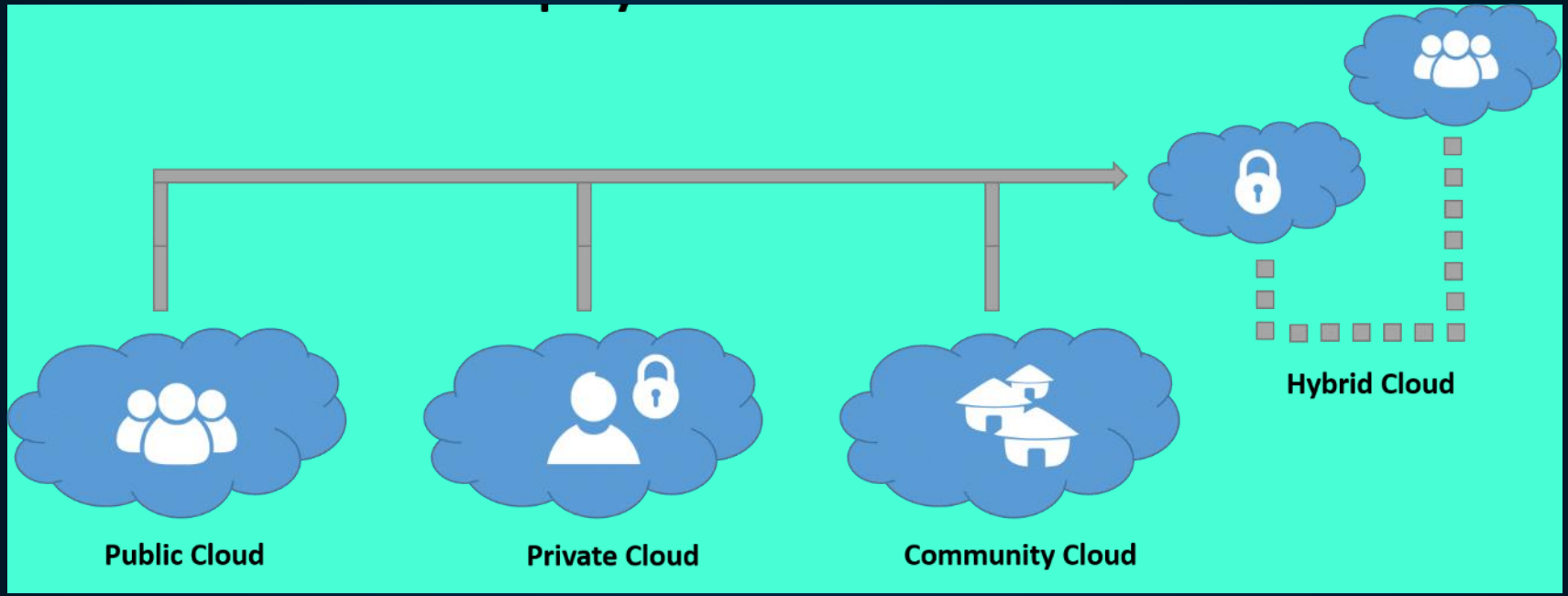
Cloud Architecture



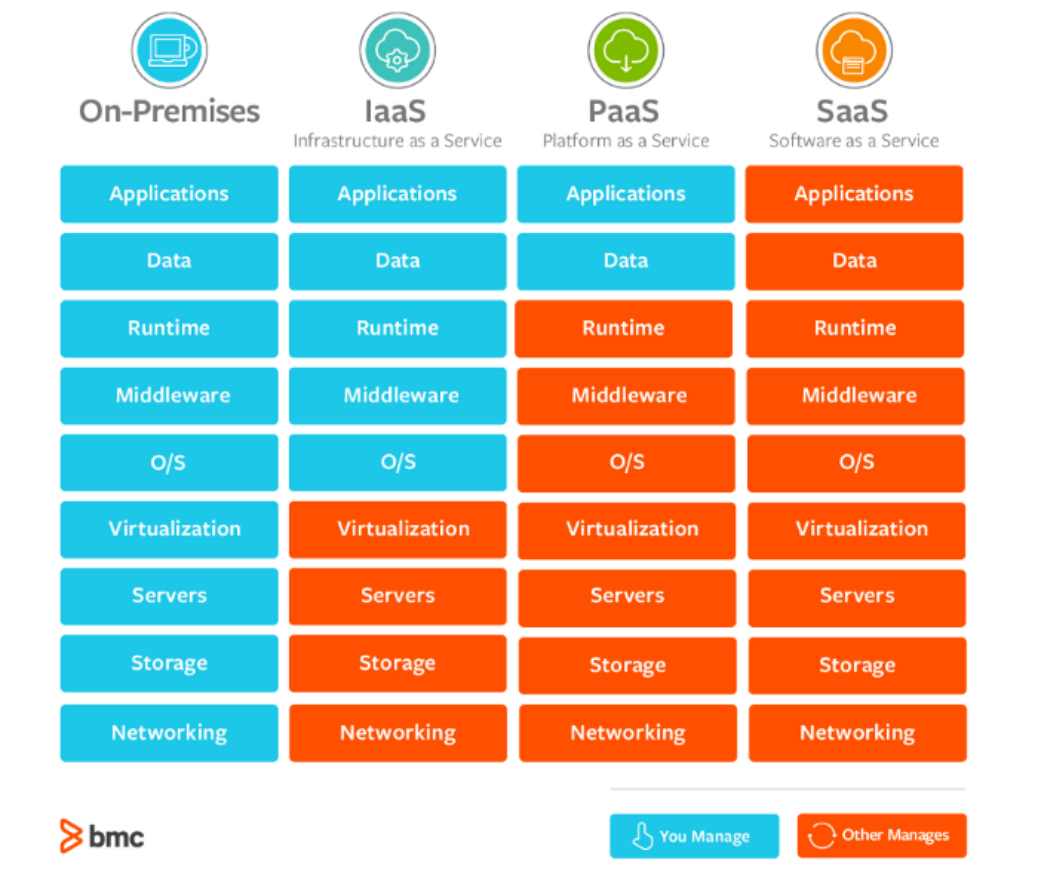
Cloud Service Models



Cloud Deployment Models

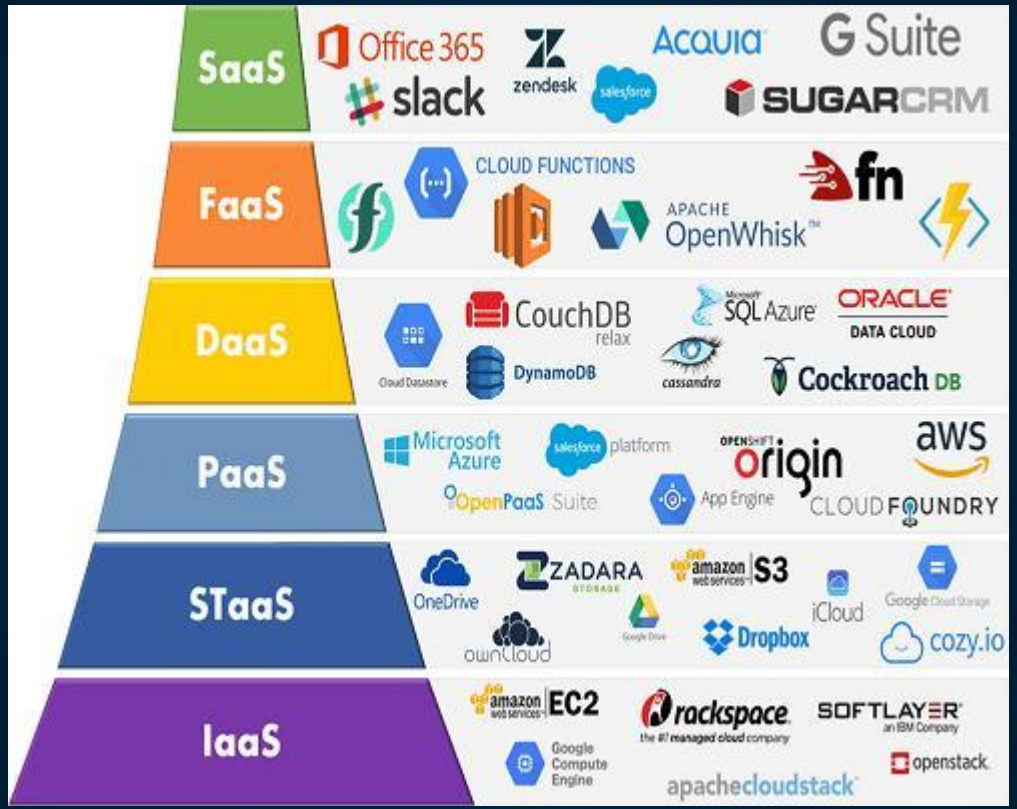


Cloud – Layerwise Provisioning



https://medium.com/@colinbaird_51123/a-primer-on-cloud-computing-9a34e90303c8

Cloud Technologies



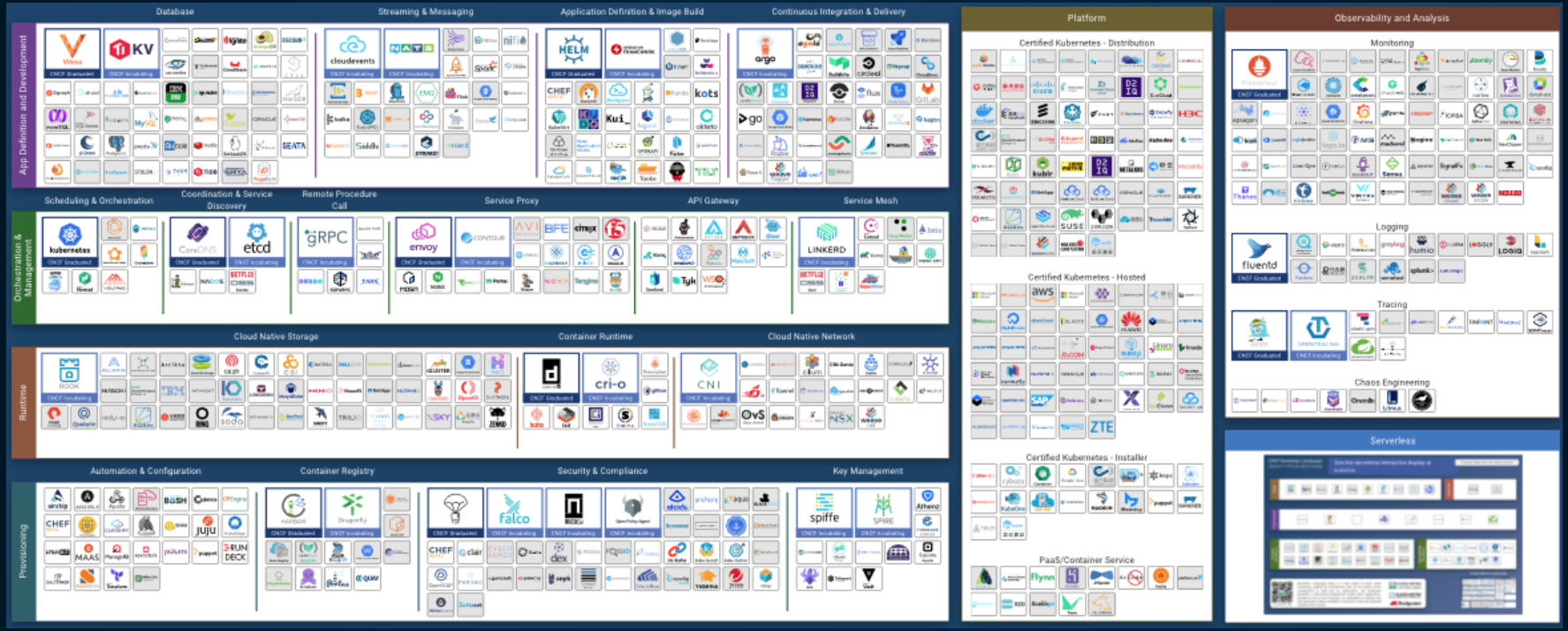
Business Cloud Technologies

Cloud Native

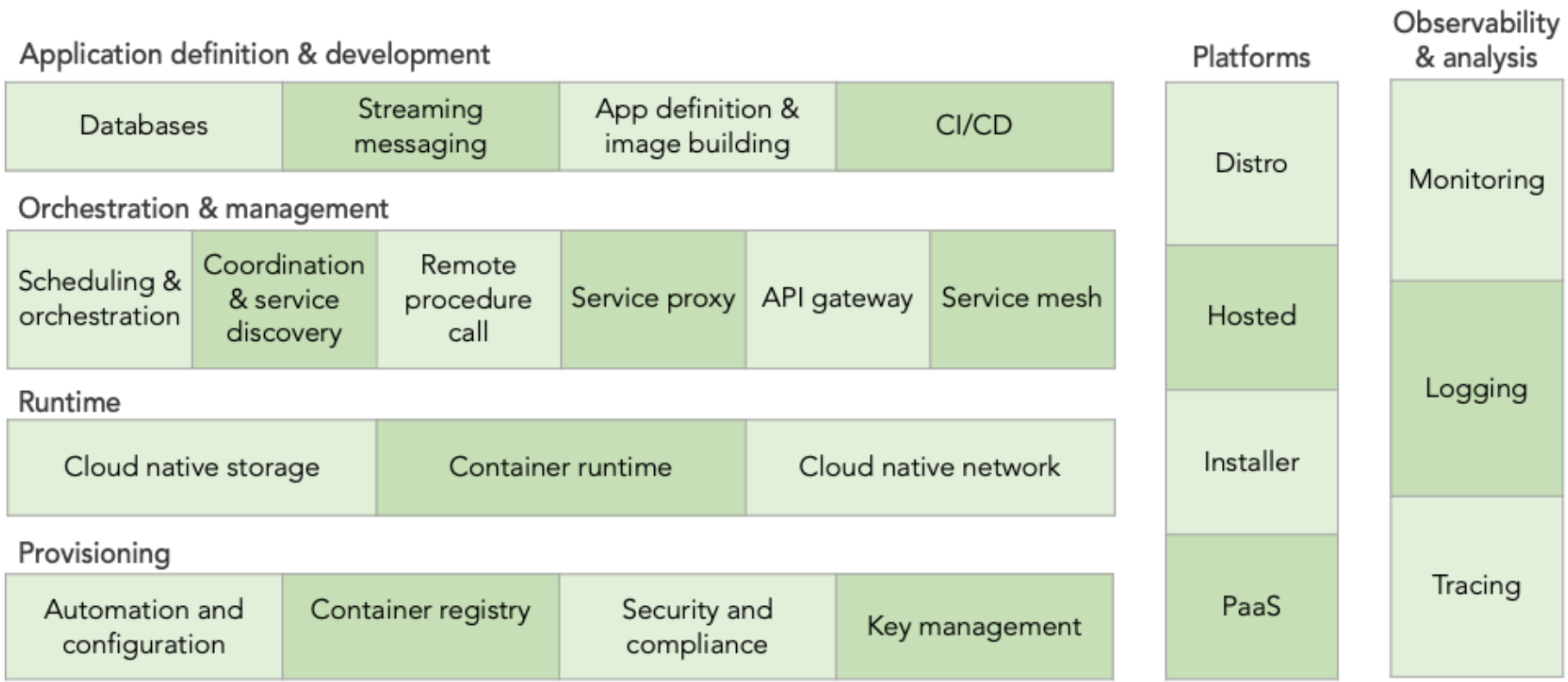
Cloud native computing uses an open-source software stack to deploy applications as microservices, packaging each part into its own container and dynamically orchestrating those containers to optimize resource utilization

> "Cloud native technologies are used to develop applications built with services packaged in containers, deployed as microservices and managed on elastic infrastructure through agile DevOps processes and continuous delivery workflows."
>
> — Writes **Janakiram MSV, principal analyst at Janakiram & Associates and an adjunct faculty member at the International Institute of Information Technology.**

Cloud Native Landscape



Cloud Native Landscape

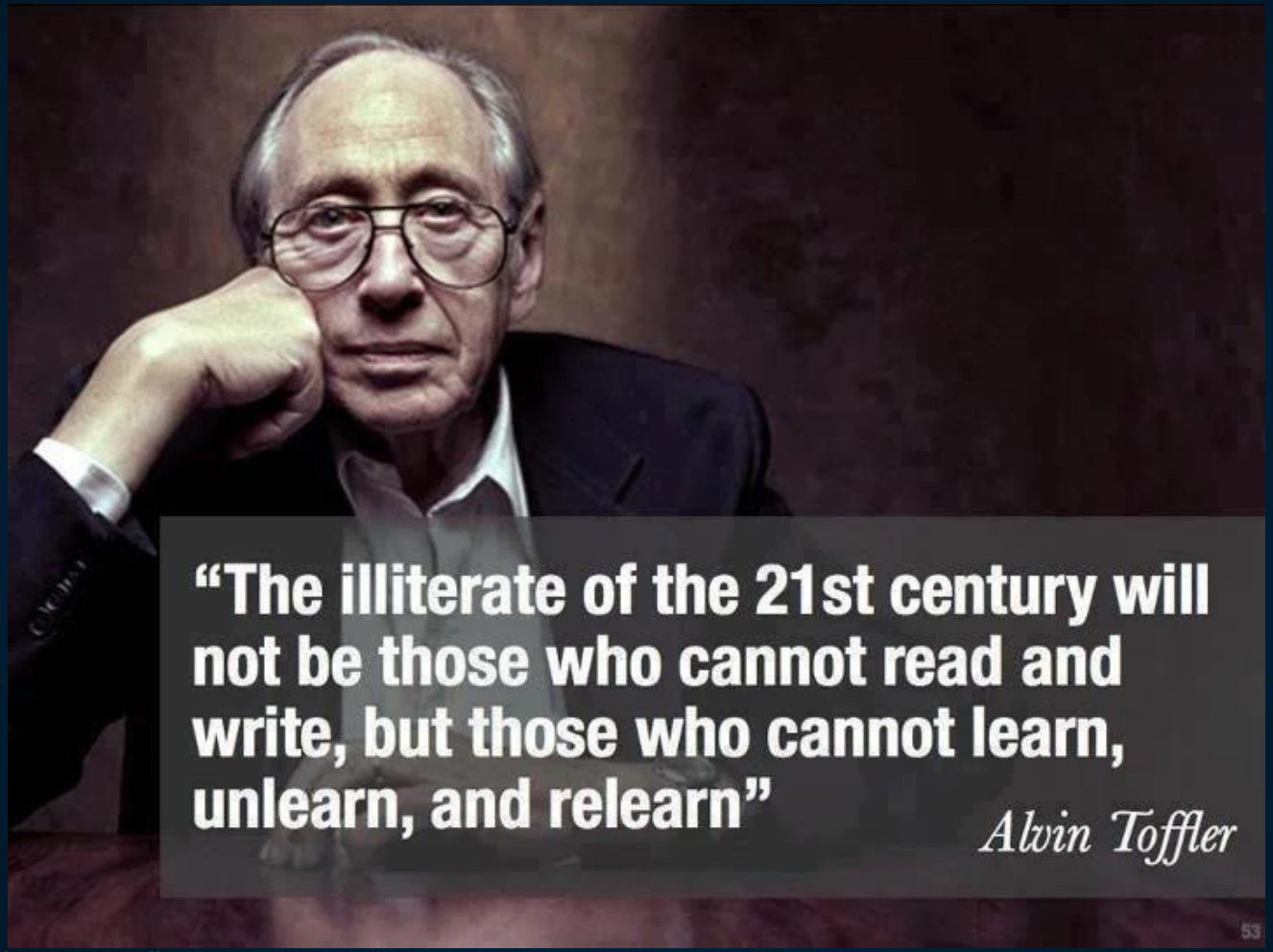


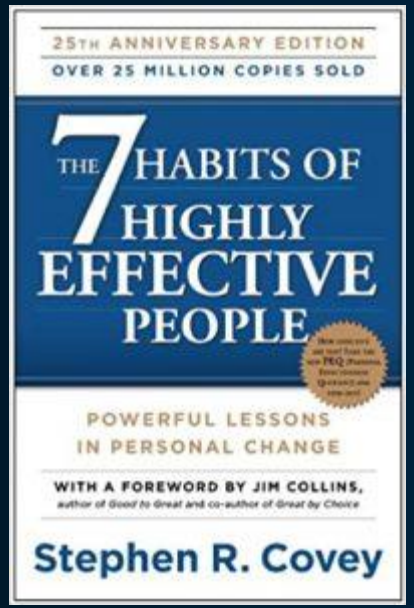
The Art of Learning

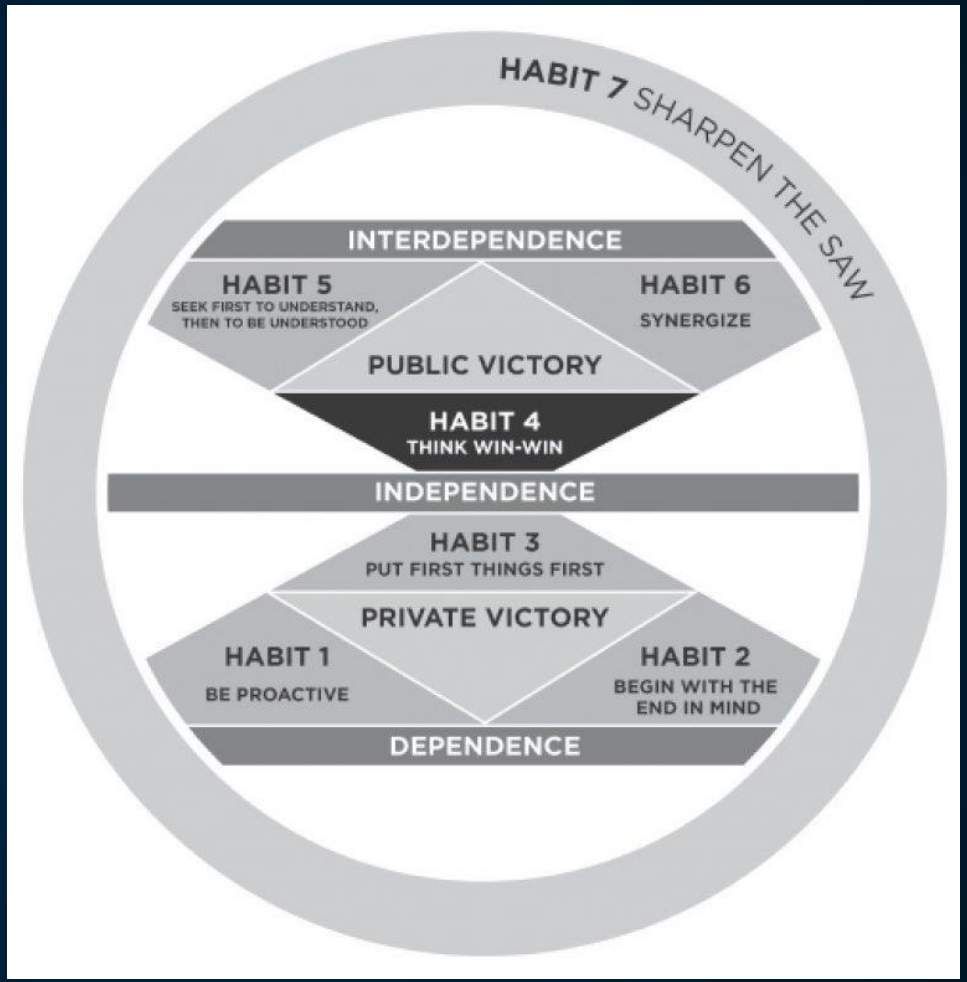
Unit 4.1

Dr. Mahesha BR Pandit

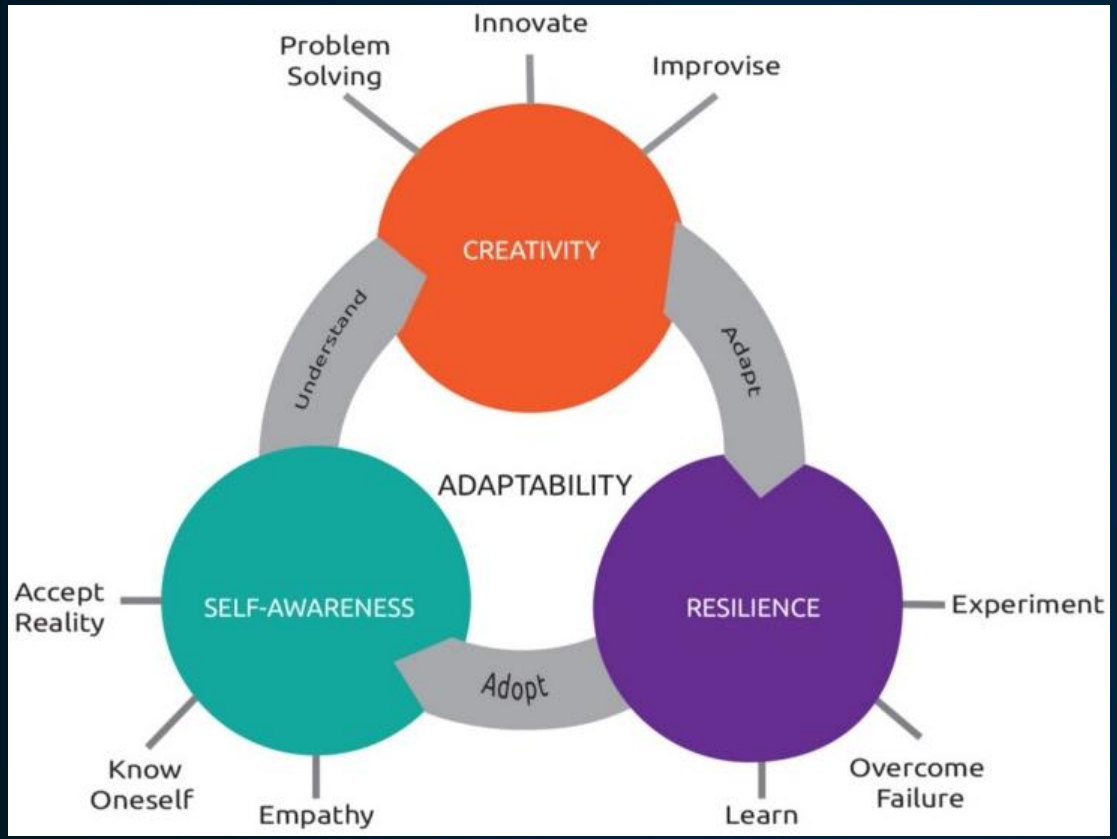
14-Dec-21 Version 1.0


“The illiterate of the 21st century will not be those who cannot read and write, but those who cannot learn, unlearn, and relearn”
Alvin Toffler
53


25TH ANNIVERSARY EDITION
OVER 25 MILLION COPIES SOLD
THE 7 HABITS OF HIGHLY EFFECTIVE PEOPLE
POWERFUL LESSONS IN PERSONAL CHANGE
WITH A FOREWORD BY JIM COLLINS,
author of Good to Great and co-author of Great by Choice
Stephen R. Covey

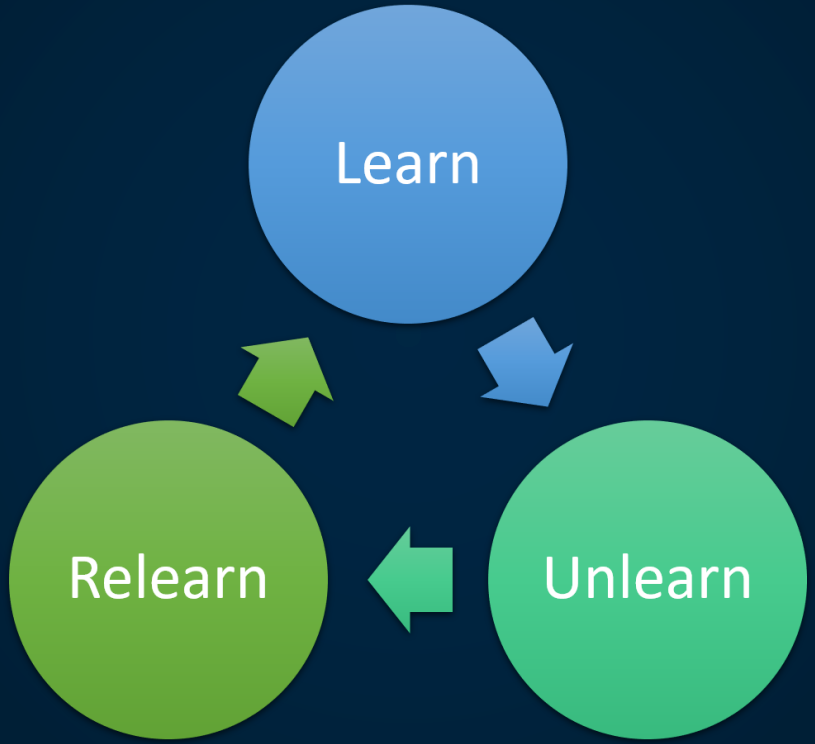

HABIT 7 SHARPEN THE SAW
INTERDEPENDENCE
HABIT 5
SEEK FIRST TO UNDERSTAND, THEN TO BE UNDERSTOOD
HABIT 6
SYNERGIZE
PUBLIC VICTORY
HABIT 4
THINK WIN-WIN
INDEPENDENCE
HABIT 3
PUT FIRST THINGS FIRST
PRIVATE VICTORY
HABIT 1
BE PROACTIVE
HABIT 2
BEGIN WITH THE END IN MIND
DEPENDENCE

Three Meta Skills

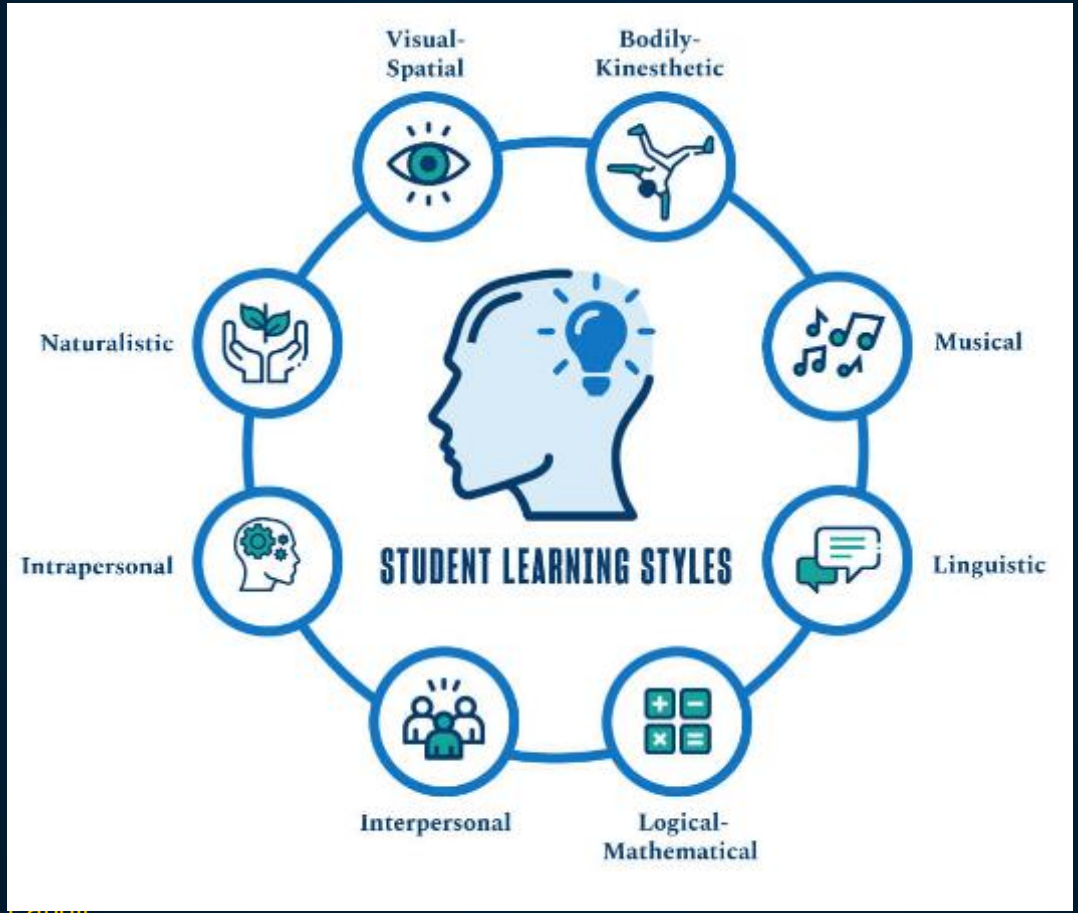


1. Self-awareness is much more than simply know oneself; it's crucial to understand reality and others.
2. Creativity is not just about developing new ideas—the changing reality requires us to improvise and solve complex problems.
3. Resilience is critical to face adversity and recover from failure—discovering new solutions requires experimenting and failing more often than not.

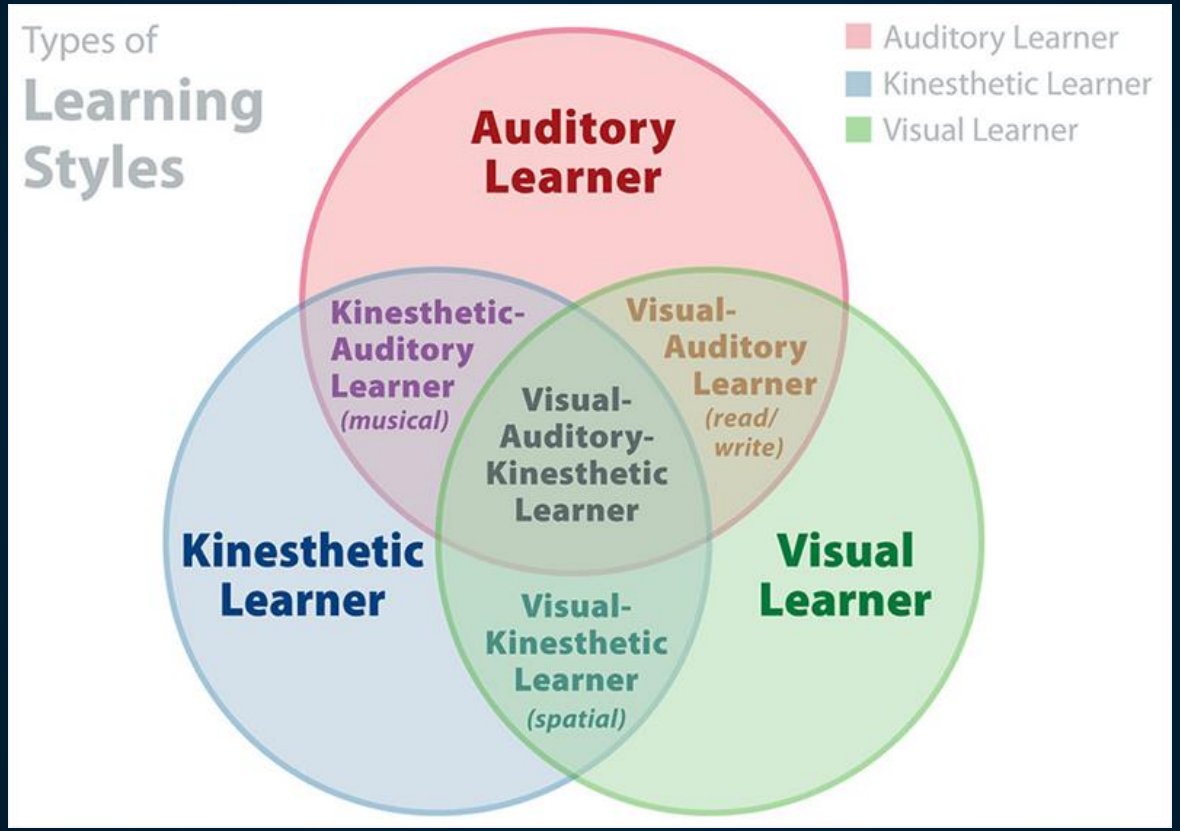
Learn – Unlearn – Relearn



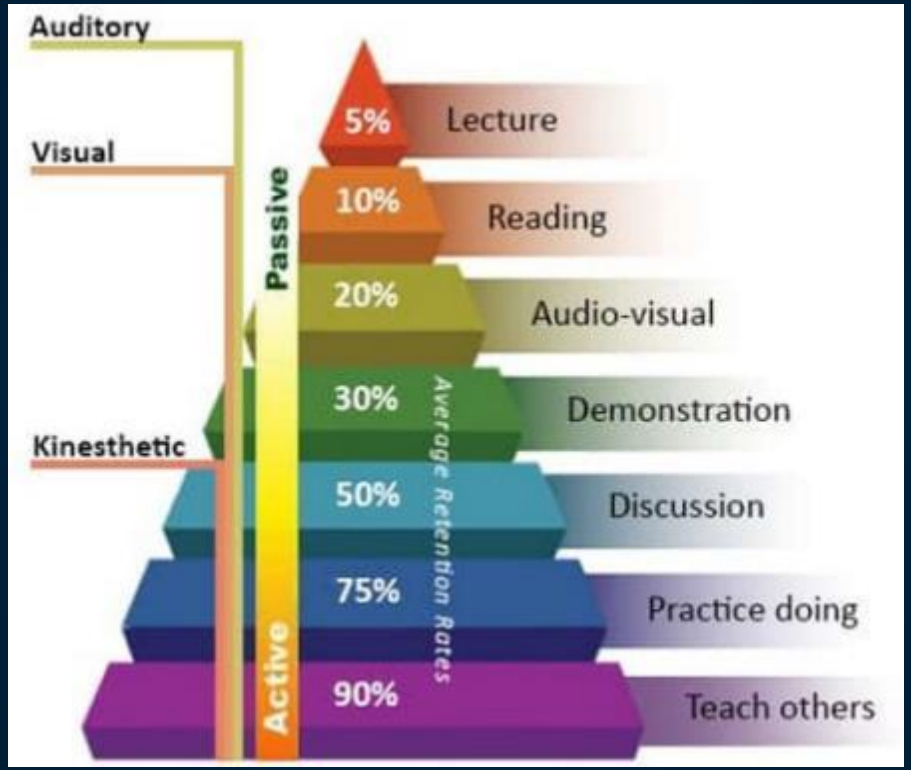
Learning Styles



VARK Model


Types of
Learning
Styles
Auditory Learner
Kinesthetic Learner
Visual Learner
Auditory Learner
Kinesthetic-Auditory Learner (musical)
Visual-Auditory Learner (read/write)
Visual-Auditory-Kinesthetic Learner
Kinesthetic Learner
Visual Learner
Visual-Kinesthetic Learner (spatial)

Learning Pyramid



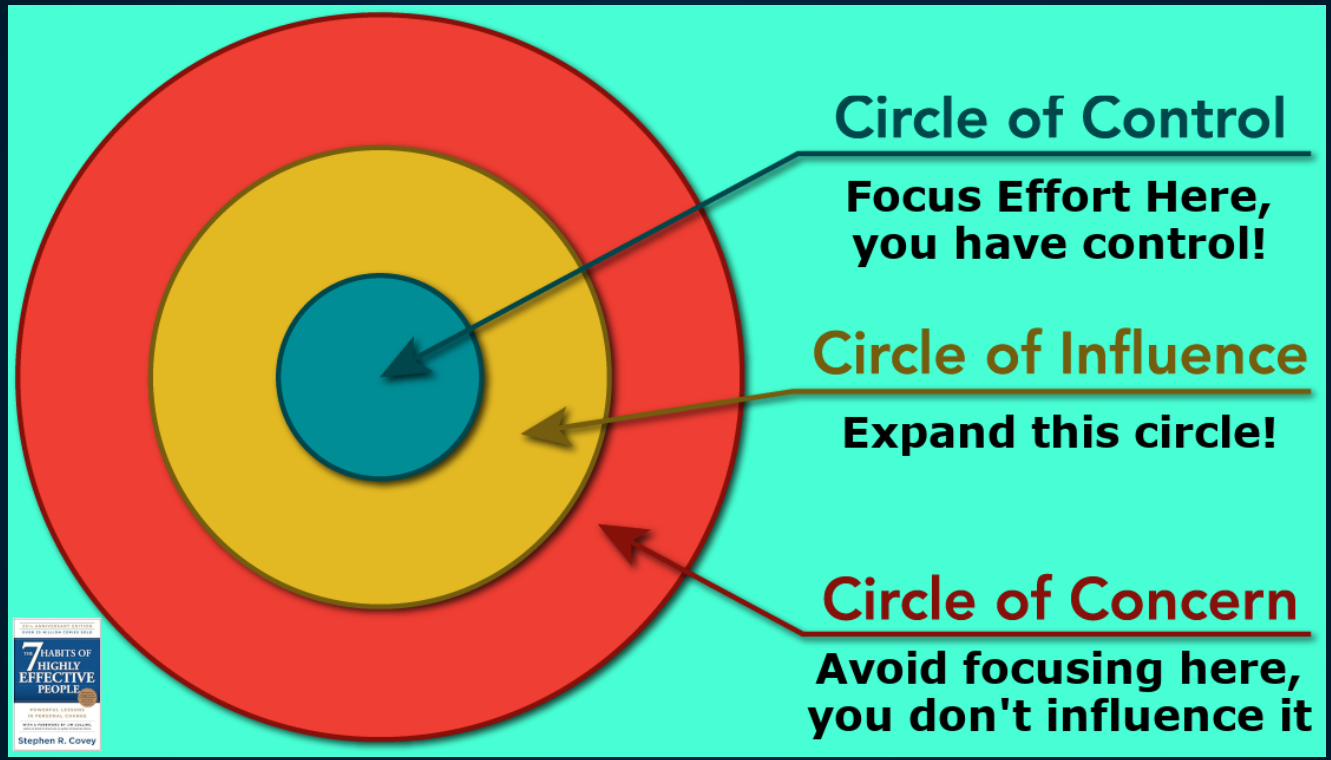
Ten Learning Techniques – 1

Ten Learning Techniques – 2

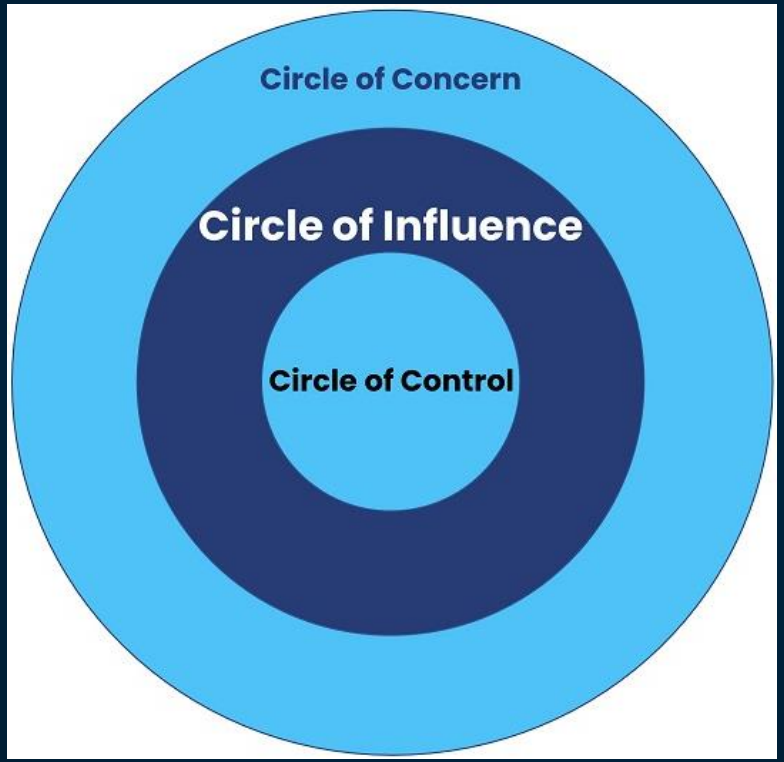
Comparison of Learning Styles

Slightly Effective	Moderately Effective	Highly Effective
• Summarization • Highlighting • Mnemonic Devices • Creating an Image • Rereading	• Elaborative Interrogation • Self-explanation • Interleaved Practice	• Distributed Practice • Practice Tests

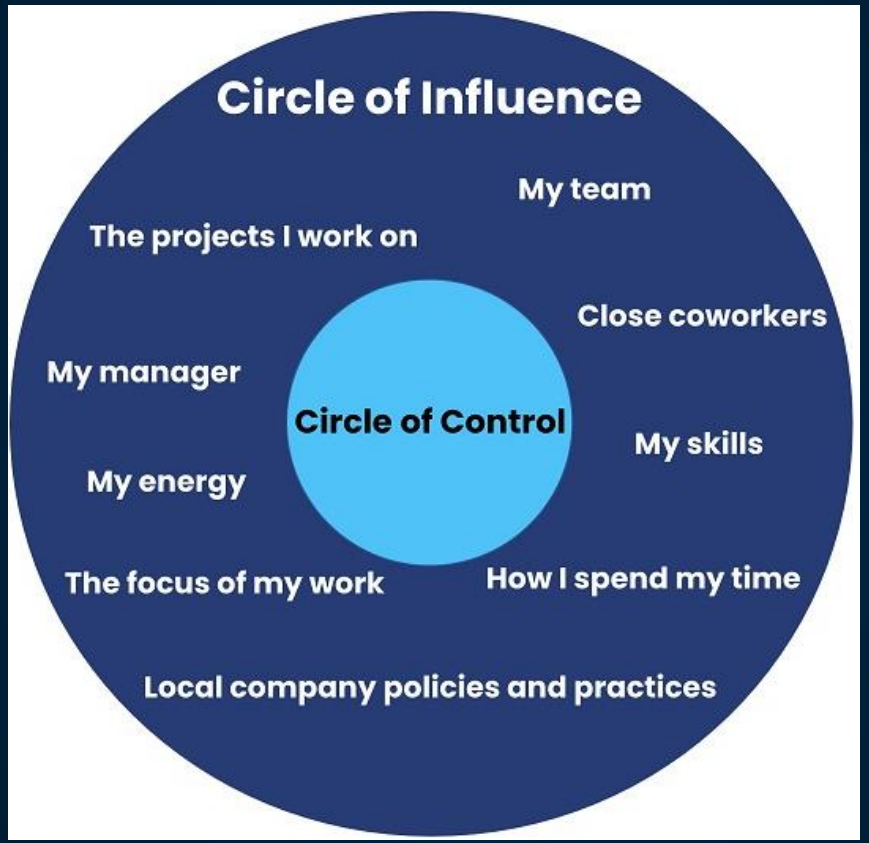
Covey's Circles



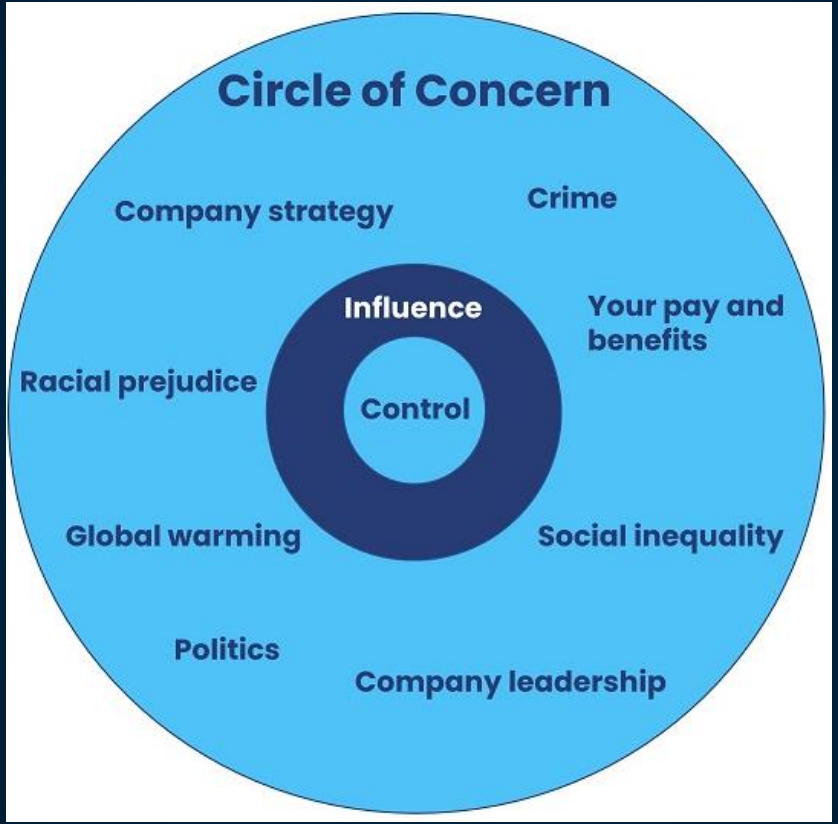
What is What?



Circle of Influence



Circle of Concern



Cultural Aspects

Unlearning (Your Cultural Conditioning)	Learning (The American Culture)
• If I ask a question, I will look stupid.	• It is the speaker's responsibility to help me understand.
• I need permission or invitation to speak up.	• It's my right to voice my views and thoughts.
• I should show respect to others because I am younger, I am a woman, or I am less experienced.	• I am an equal member of an interaction regardless of age, gender, and experience.
• I should not show much confidence because people may perceive me as arrogant.	• I can present myself confidently without looking arrogant.
• I should not disagree because it is disrespectful.	• Disagreeing doesn't mean disrespect; it is to share another perspective.