IBM

Tools for Data Science

What is a Data Scientist

a data scientist is simply a person who can

write code

understand statistics

derive insights from data

Oh really, is this a Data Scientist ?

a data scientist is simply a person who can

write code = in R,Python,Java, SQL, Hadoop (Pig,HQL,MR) etc

= for data storage, querying, summarization, visualization

= how efficiently, and in time (fast results?)

= where on databases, on cloud, servers

and understand **enough** statistics

to derive **insights** from data

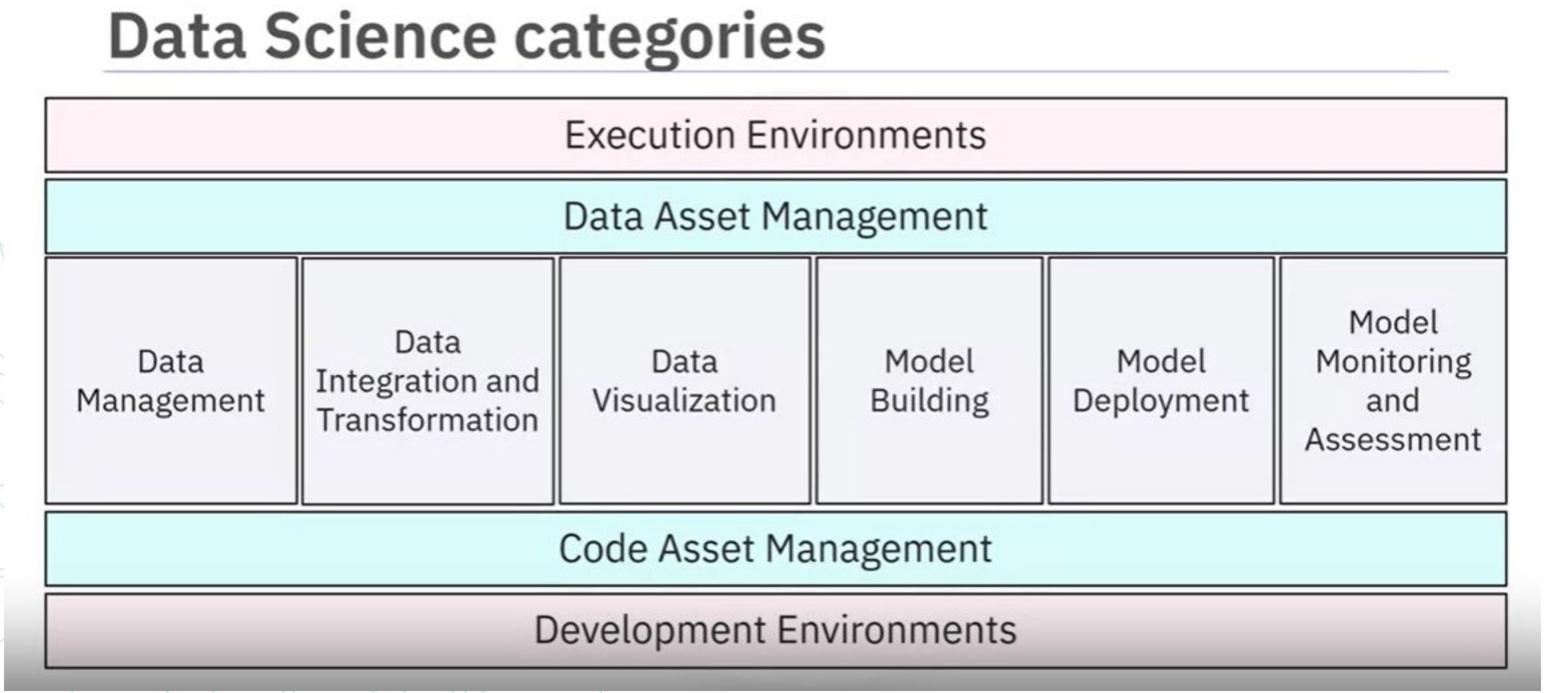
so business can make **decisions**

- **In this course you will learn how to:**

- Describe the Data Scientist’s tool kit which includes Libraries & Packages, Data sets, Machine learning models, and Big Data tools.
- Utilize languages commonly used by data scientists like Python, R, and SQL
- Demonstrate working knowledge of tools such as Jupiter notebooks and RStudio and utilize their various features.
- Create and manage source code for data science using Git repositories and GitHub.

- **Course Modules:**

1. Overview of Data Science Tools
2. Languages of Data Science
3. Packages, APIs, Database & Models
4. Jupyter Notebooks and Jupyter lab
5. RStudio & GitHub
6. IBM Watson Studio



Data Science categories

Execution Environments

Data Asset Management

Data Management

Data Integration and Transformation

Data Visualization

Model Building

Model Deployment

Model Monitoring and Assessment

Code Asset Management

Development Environments

Data Science Task Categories

- **Data Management:** Collect, persist, and retrieve data securely and efficiently from various sources like social media and sensors.
- **Data Integration and Transformation (ETL):** Extract data from multiple repositories, transform its values and structure, and load it into a central Data Warehouse.
- **Data Visualization:** Graphical representation of data using charts, plots, maps, etc., for effective decision-making.
- **Model Building:** Train data with machine learning algorithms to analyze patterns and make predictions.
- **Model Deployment:** Integrate developed models into production environments via APIs for data-based decision-making.
- **Model Monitoring and Assessment:** Continuous quality checks to ensure model accuracy, fairness, and robustness.

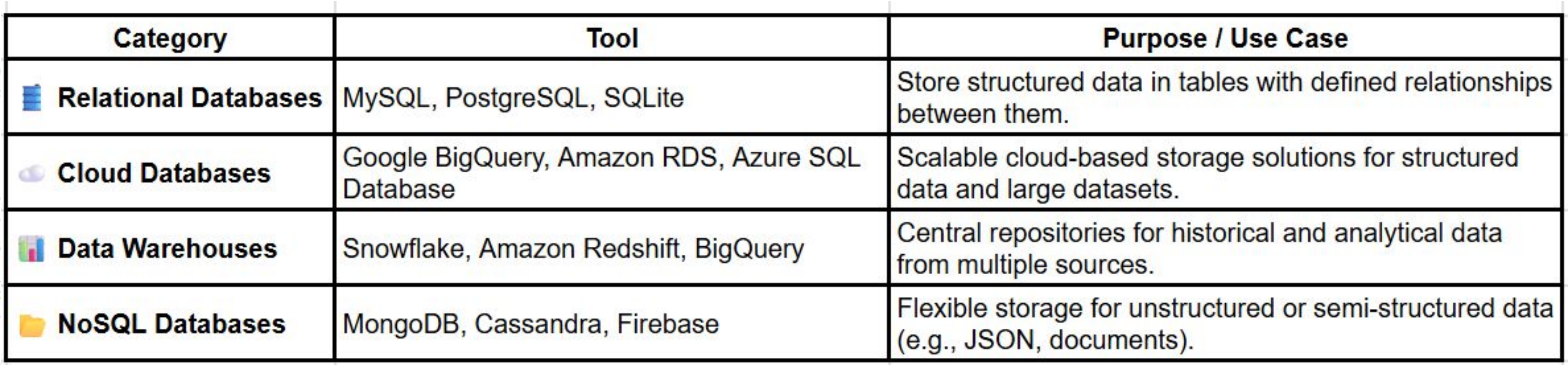
Supporting Tools and Environments

- **Code Asset Management (GitHub):** Version control and collaboration for managing code files and project updates.
- **Data Asset Management (DAM Platforms):** Organize and manage data collected from different sources with versioning and collaboration support.
- **Development Environments (IDEs):** Provide workspace and tools for developing, testing, and deploying source code.
- **Execution Environments (Cloud-based):** Libraries for compiling source code and resources for executing and verifying code.
- **Fully Integrated Visual Tools (IBM Watson Studio, Cognos):** Cover all aspects of data science tasks and enable deep learning and machine learning model development.

Module I: Overview of Data Science Tools

Tools for Data Science

Data Management Tools



Category	Tool	Purpose / Use Case
🗄️ **Relational Databases**	MySQL, PostgreSQL, SQLite	Store structured data in tables with defined relationships between them.
☁️ **Cloud Databases**	Google BigQuery, Amazon RDS, Azure SQL Database	Scalable cloud-based storage solutions for structured data and large datasets.
📊 **Data Warehouses**	Snowflake, Amazon Redshift, BigQuery	Central repositories for historical and analytical data from multiple sources.
📁 **NoSQL Databases**	MongoDB, Cassandra, Firebase	Flexible storage for unstructured or semi-structured data (e.g., JSON, documents).

- **MySQL** **(in course)**

Open-Source Data Management, Integration, and Visualization Tools

- **Data Integration and Transformation Tools:**

- Termed: **Data Refinery and Cleansing**
- **Pandas** (in course)
- Apache AirFlow, KubeFlow
- Apache Kafka, Apache Nifi, Apache SparkSQL
- NodeRED

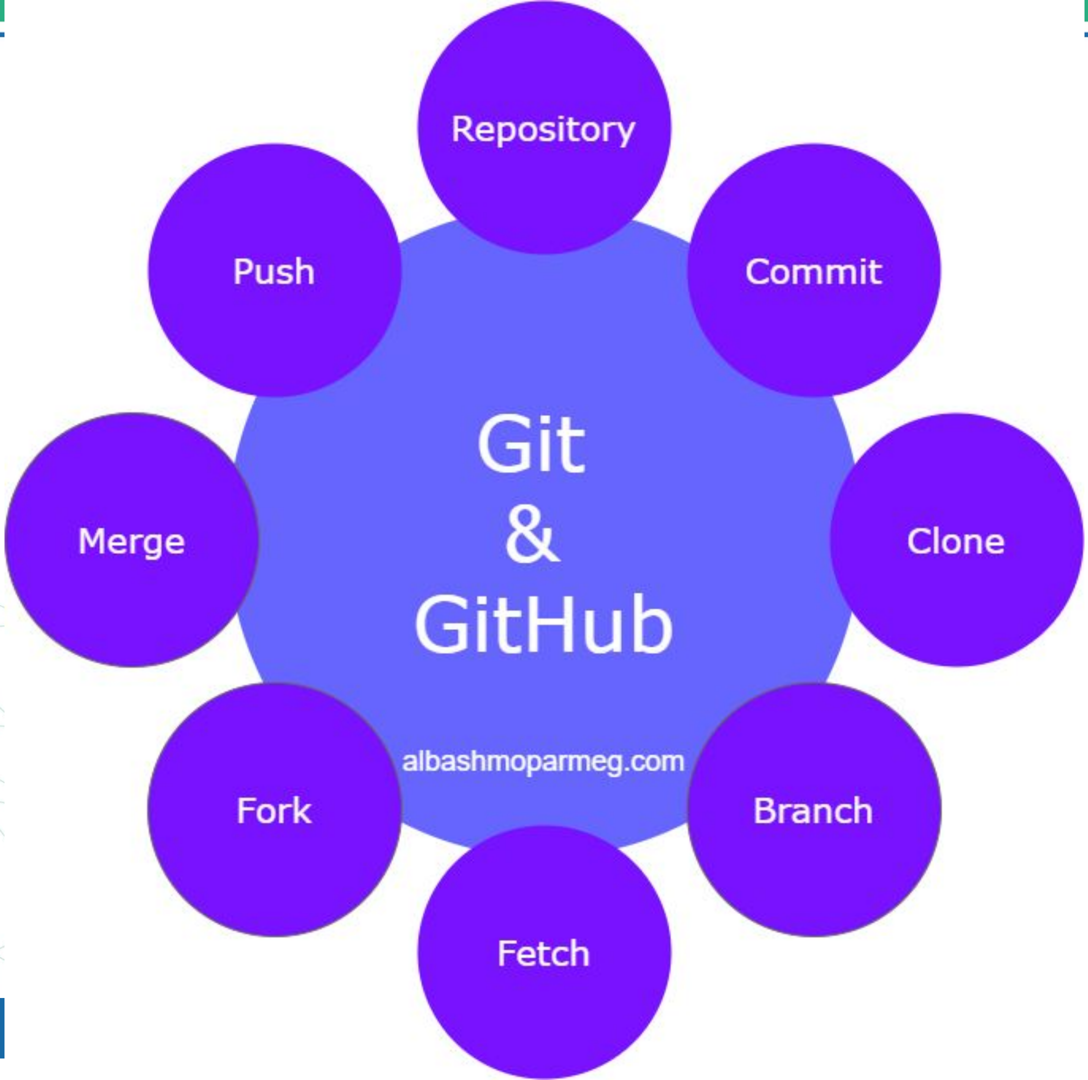
- **Data Visualization Tools:**
  - **Matplotlib & Seaborn (Python)** (in course)
  - **Tableau** (Good to try it and test)
  - **Power BI** (Good to try it and test)
  - PixieDust, Hue
  - Kibana
  - Apache Superset

Model Tools, Deployment, and Monitoring

- **Model Tools for Building,**
  - Scikit-learn (as we need in a course)
  - TensorFlow & Keras (as we need in a course)
  - PyTorch (as we need in a course)
- **Model Tools Deployment**
  - Amazon SageMaker, Google AI Platform, Azure Machine Learning
  - Flask & FastAPI (in course)
  - Streamlit (in course)
  - Apache PredictionIO, Seldon
  - Kubernetes, Redhat OpenShift, Mleap
  - **TensorFlow Tools**
    - TensorFlow service, TensorFlow lite, TensorFlow dot Js

- **Model Monitoring Tools:**
  - MLflow (in course)
  - ModelDB, Prometheus
  - IBM AI Fairness 360, IBM Adversarial Robustness 360 Toolbox
  - IBM AI Explainability 360

- **Code Asset Management Tools:**
  - Git **Version Control**
    - GitHub (self working in course)
    - GitLab, Bitbucket
- **Data Asset Management Tools**
  - Apache Atlas, ODPi Egeria, Kylo


Repository
Push
Commit
Merge
Git
&
GitHub
Clone
albashmoparmeg.com
Fork
Branch
Fetch

Development Environments and Execution Environments

- **Development Environments:**

- Anaconda,Jupyter Notebook & lab (using in course)
- Spyder
- RStudio

- **Execution Environments:**

- Docker
- AWS SageMaker
- Google Cloud AI Platform
- Watson Studio Desktop for data science development
- Watson Studio, Watson Open Scale for fully integrated data science lifecycle
- Apache Spark, Apache Flink, Ray
- Fully Integrated Visual Tools: KNIME, Orange

Module I: Overview of Data Science Tools

Commercial Tools for Data Science

Commercial Data Management, Integration, and Visualization Tools

- **Data Management Tools:**

- Oracle Database
- Microsoft SQL Server
- IBM Db2

- **Data Integration Tools:**

- Informatica PowerCenter
- IBM InfoSphere DataStage
- SAP, SAS, Talend, Microsoft

- **Data Visualization Tools:**

- Tableau
- Microsoft Power BI
- IBM Cognos Analytics

Model Tools, Deployment, Monitoring, and Fully Integrated Solutions

- **Model Tools for Building and Deployment:**

- SPSS Modeler
- SAS Enterprise Miner

- **Model Deployment and Monitoring:**

- SPSS Collaboration and Deployment Services
- PMML format for model export

- **Code and Data Asset Management Tools:**

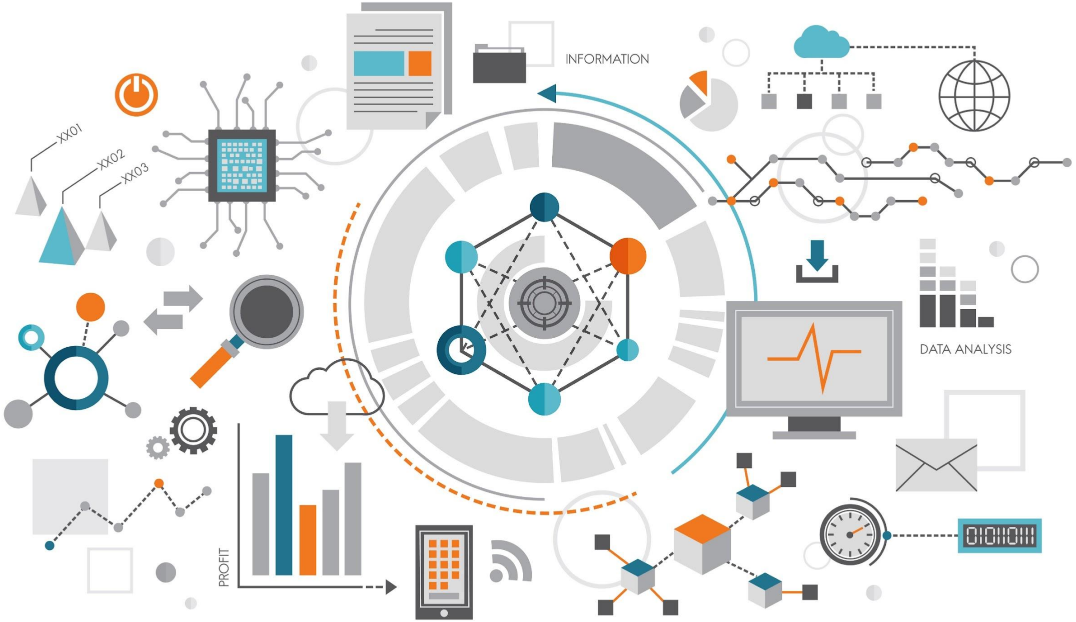
- Informatica Enterprise Data Governance, IBM tools for data asset management

- **Development Environment and Fully Integrated Tools:**

- Watson Studio Desktop for data science development
- Watson Studio, Watson Open Scale for fully integrated data science lifecycle

Module I: Overview of Data Science Tools

Cloud Based Tools for Data Science



Cloud-Based Tools for Data Science Overview

- **Fully Integrated Visual Tools:**

- Watson Studio & Watson OpenScale
- Microsoft Azure Machine Learning
- H2O Driverless AI

- **Data Management in the Cloud:**

- SaaS versions of open-source and commercial tools
- Examples: AWS DynamoDB, Cloudant, IBM Db2

- **Cloud Data Integration Tools:**

- Informatica Cloud Data Integration
- IBM Data Refinery (in Watson Studio)

Cloud-Based Data Visualization and Model Tools

- **Cloud Data Visualization Tools:**

- Datameer

- IBM Cognos Business Intelligence Suite

- Data exploration and visualization in Watson Studio

- **Model Building and Deployment:**

- Watson Machine Learning

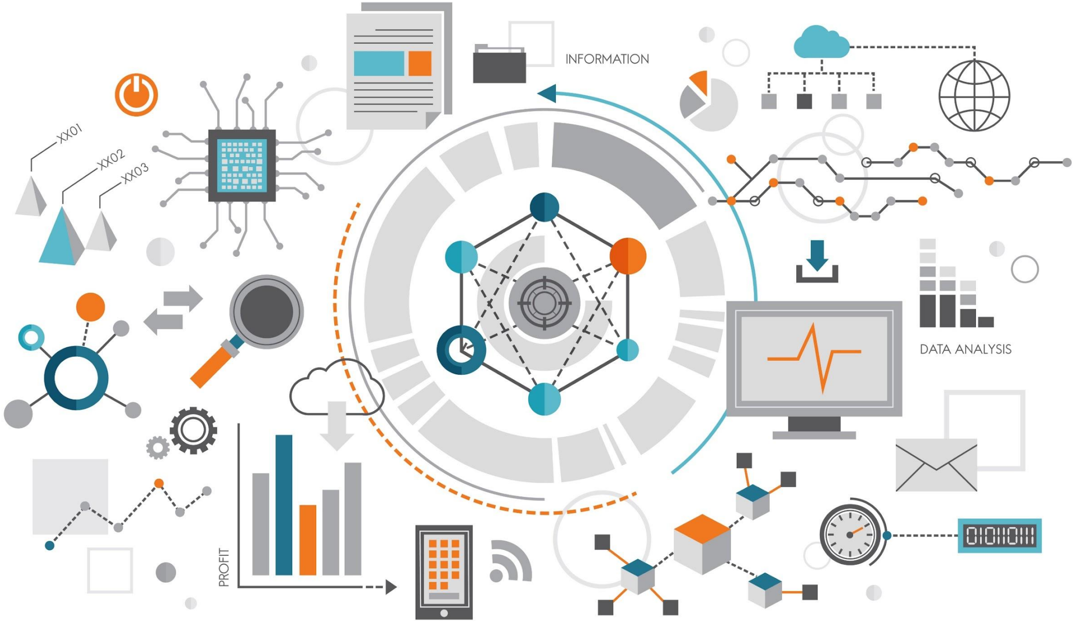
- Amazon SageMaker Model Monitor

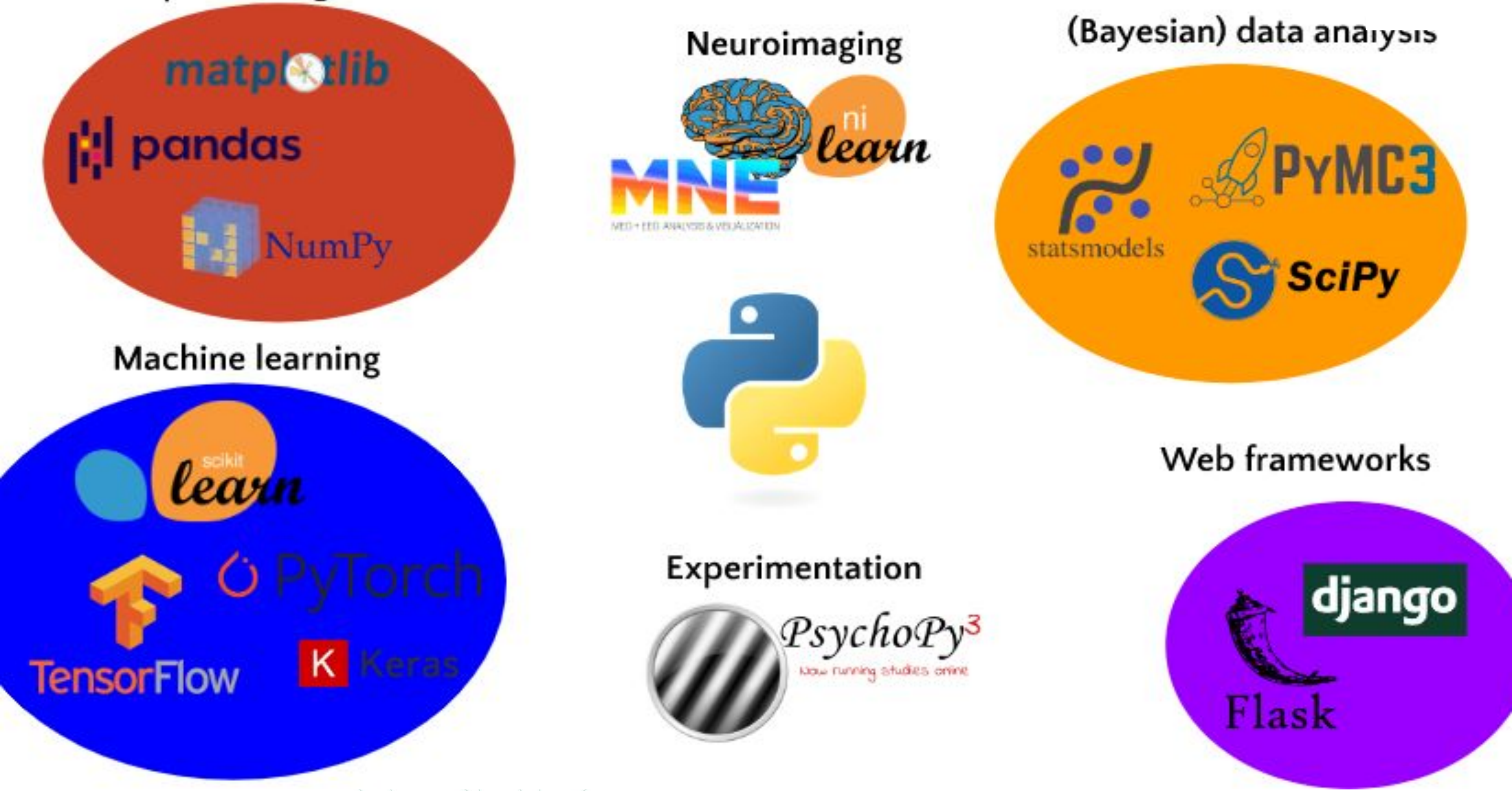
- SPSS Collaboration and Deployment Services

Module II
Packages, APIs, Datasets and Models

Module II: Packages, APIs, Datasets and Models

Libraries for Data Science





Data processing/visualization

Neuroimaging

(Bayesian) data analysis

Machine learning

Experimentation

Web frameworks

Python Libraries for Data Science

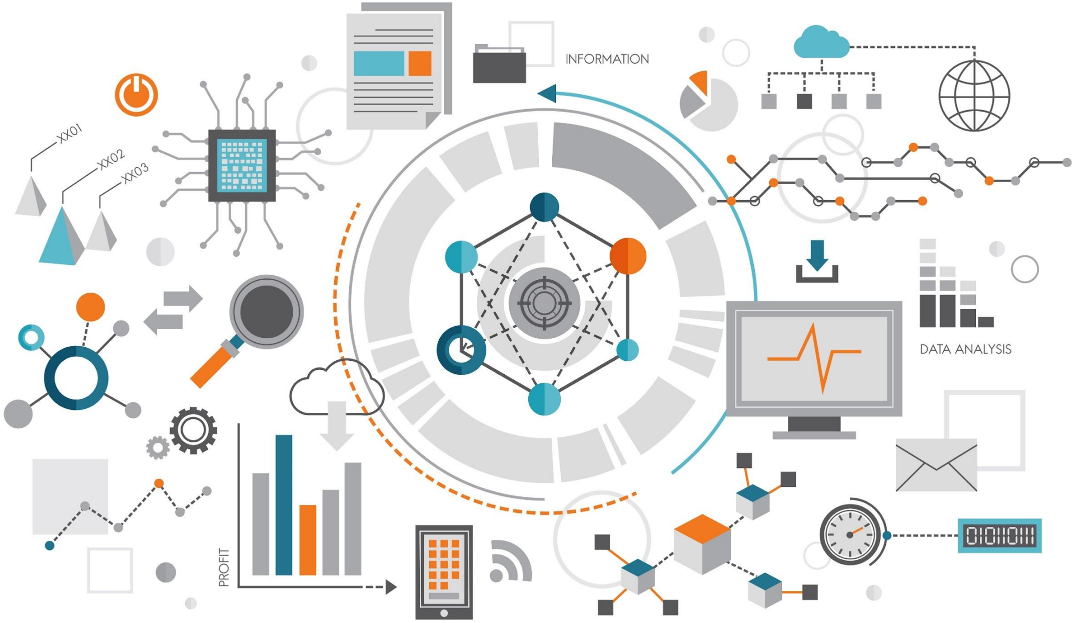
- **Scientific Computing Libraries:** Includes Pandas for data cleaning and manipulation, offering data structures like Data Frames.
- **Visualization Libraries:** Matplotlib for customizable graphs and charts, Seaborn for heat maps and violin plots.
- **High-Level Machine Learning Libraries:** Scikit-learn for regression, classification, and clustering tasks.
- **Deep Learning Libraries:** Keras for quick model building, TensorFlow for large-scale deep learning production.
- **Other Languages Libraries:** LangChain, Transformers, R and Scala.

Complementary Libraries and Tools

- **Scala Libraries:** Vegas for statistical data visualizations, complementary to Apache Spark.
- **R Libraries:** ggplot2 for data visualization, interfaces with Keras and TensorFlow for deep learning.
- **Advantages of Python:** Python libraries and frameworks are replacing R in open-source data science.
- **Interfacing with TensorFlow:** Libraries in R that allow seamless integration with TensorFlow.
- **Apache Spark Capabilities:** Supports data processing in parallel using compute clusters in various languages.

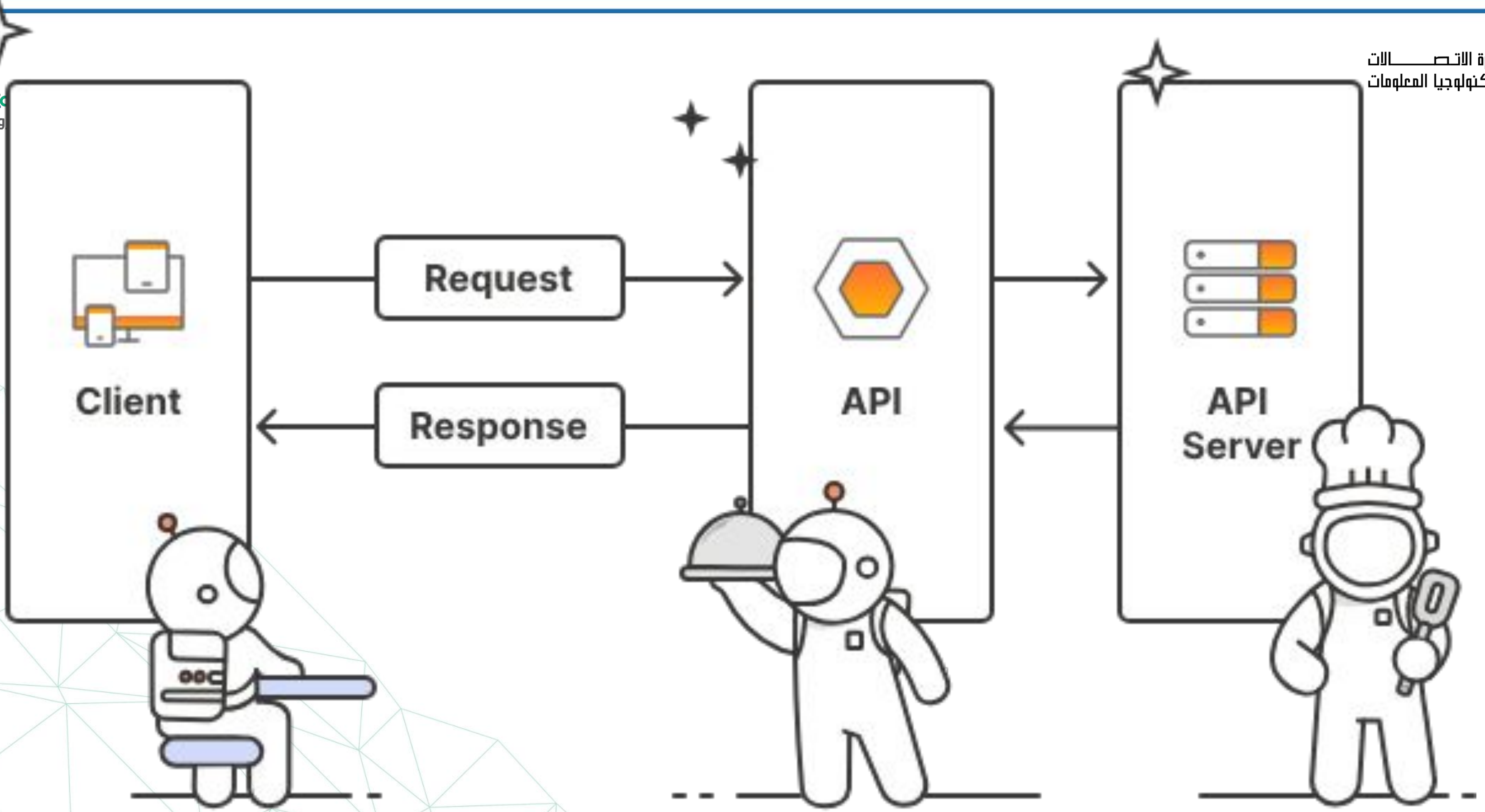
Module II: Packages, APIs, Datasets and Models

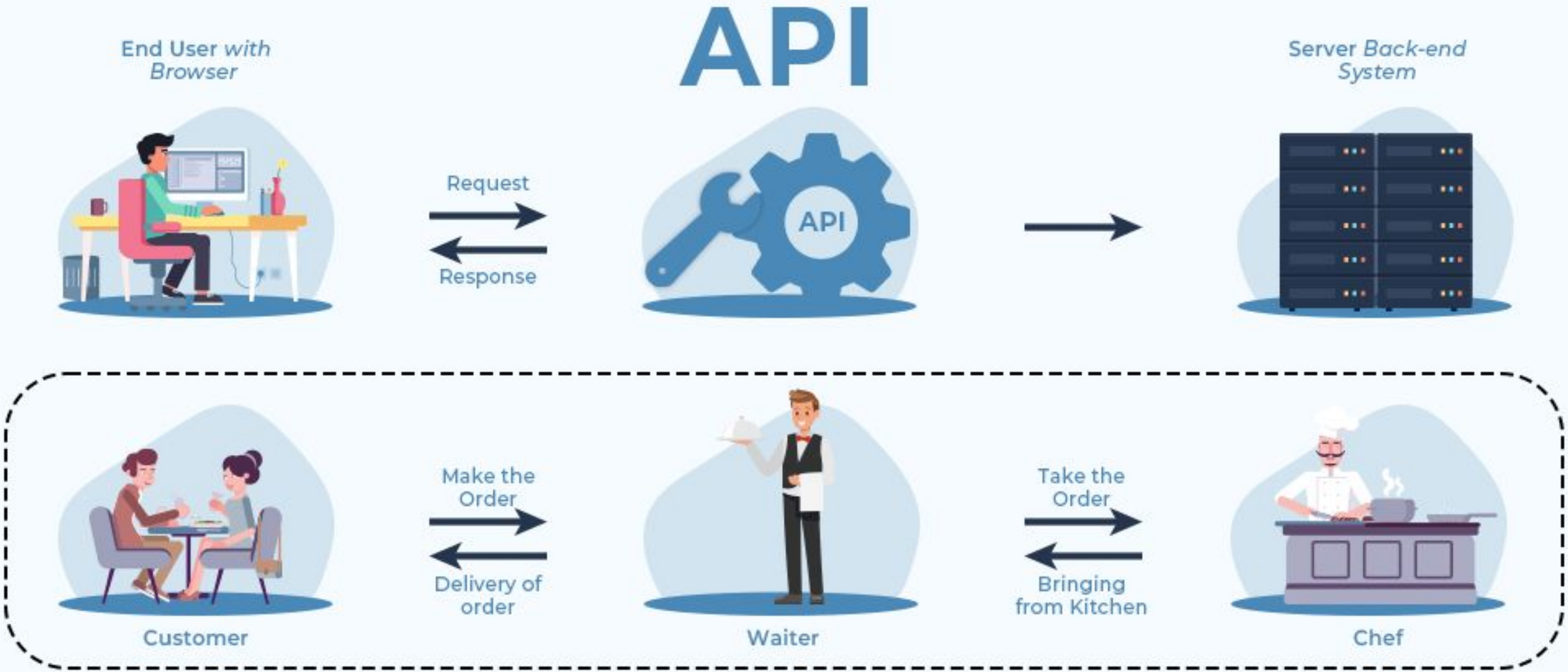
Application Programming Interfaces (APIs)



What is an API?

- API stands for Application Programming Interface.
- It allows different software applications to communicate and share data.
- Works like a translator between two systems, enabling them to work together.


Client
Request
Response
API
API
Server


API
End User *with* Browser
Request
Response
API
Server *Back-end* System
Make the Order
Delivery of order
Take the Order
Bringing from Kitchen
Customer
Waiter
Chef

How APIs Work: Request-Response Cycle

1. **API Client**:
   - The application sending a request.
   - Example: A mobile app asking a server for data.
2. **API Server**:
   - The system that receives the request and processes it.
   - Example: A database or web server that holds the requested information.

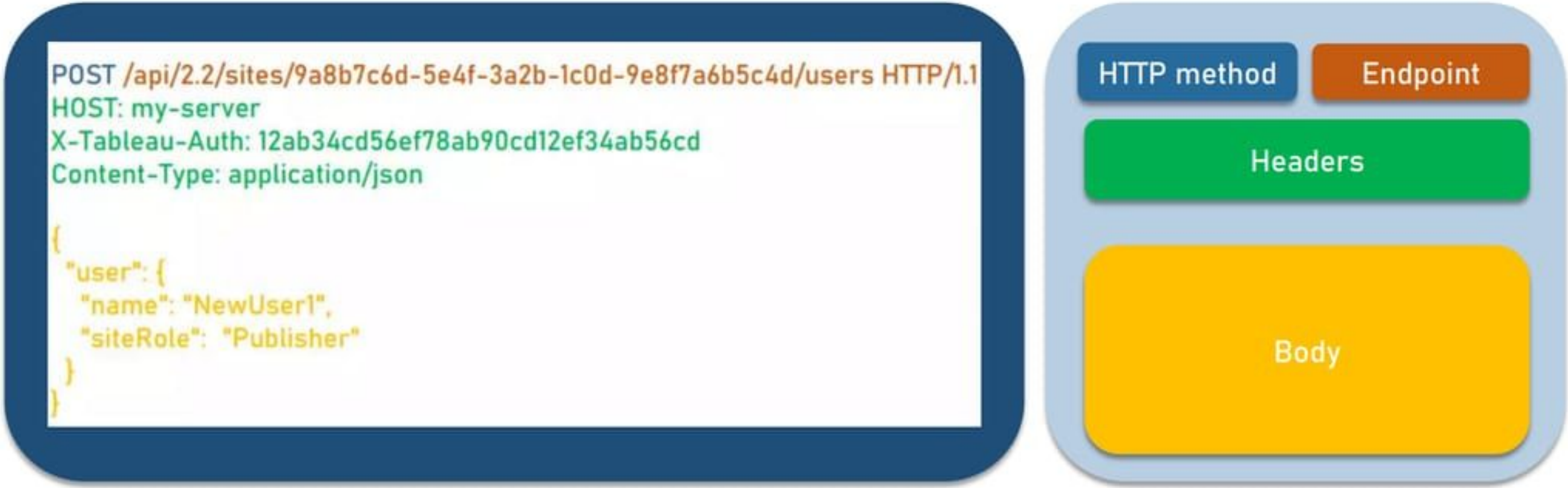
API Request Components

- **Endpoint**: The URL where the API can be accessed (e.g., `/users`).
- **Method**: The type of request (e.g., GET, POST, PUT, DELETE).
- **Parameters**: Information passed to the API, either in the URL or the request body.
- **Headers**: Extra information like authentication or content type.
- **Body**: Data sent with the request (only for POST and PUT methods).

API Response Components

- **Status Code:** https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#redirection_messages
  - Indicates the result of the request (e.g., `200 OK`, `404 Not Found`).
- **Headers**:
  - Additional information like content type.
- **Response Body**:
  - The actual data returned by the API or an error message.

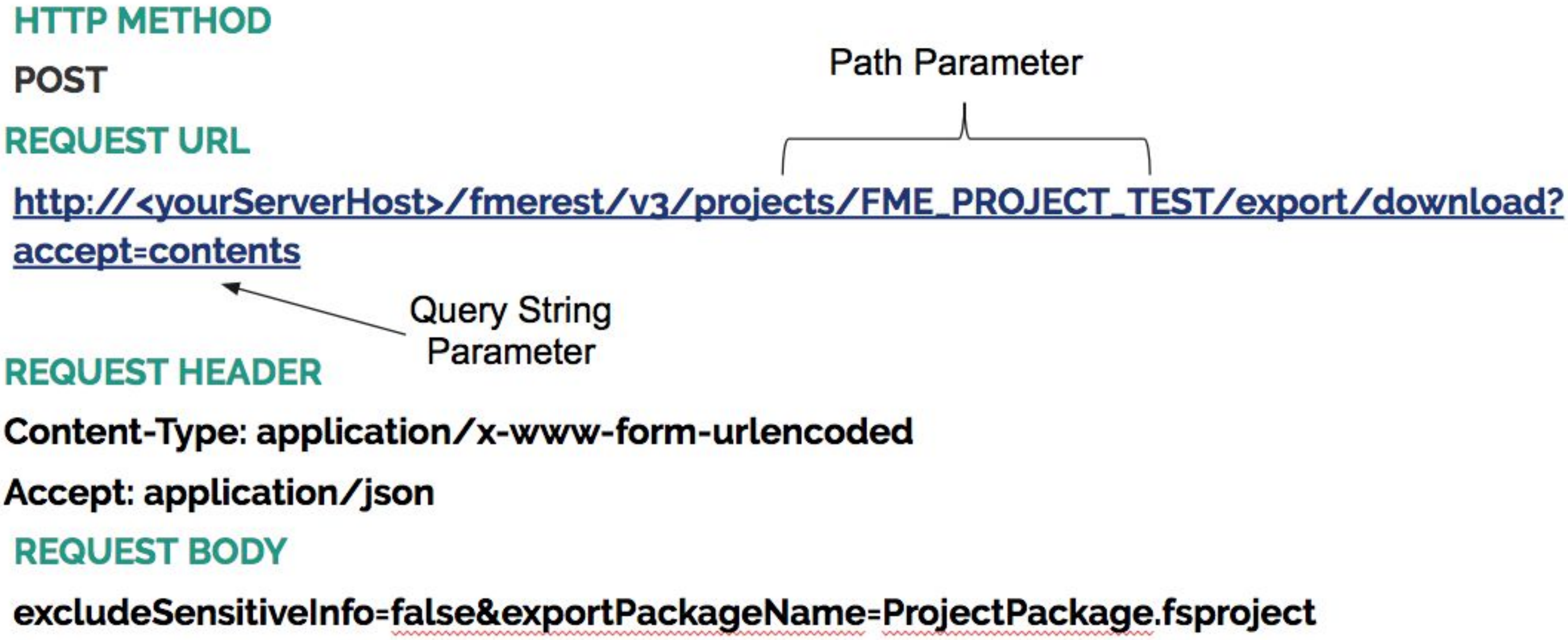
REST API REQUEST



```
POST /api/2.2/sites/9a8b7c6d-5e4f-3a2b-1c0d-9e8f7a6b5c4d/users HTTP/1.1
HOST: my-server
X-Tableau-Auth: 12ab34cd56ef78ab90cd12ef34ab56cd
Content-Type: application/json

{
 "user": {
  "name": "NewUser1",
  "siteRole":  "Publisher"
 }
}
```

altexsoft



HTTP METHOD

POST

Path Parameter

REQUEST URL

http://<yourServerHost>/fmerest/v3/projects/FME_PROJECT_TEST/export/download?accept=contents

Query String Parameter

REQUEST HEADER

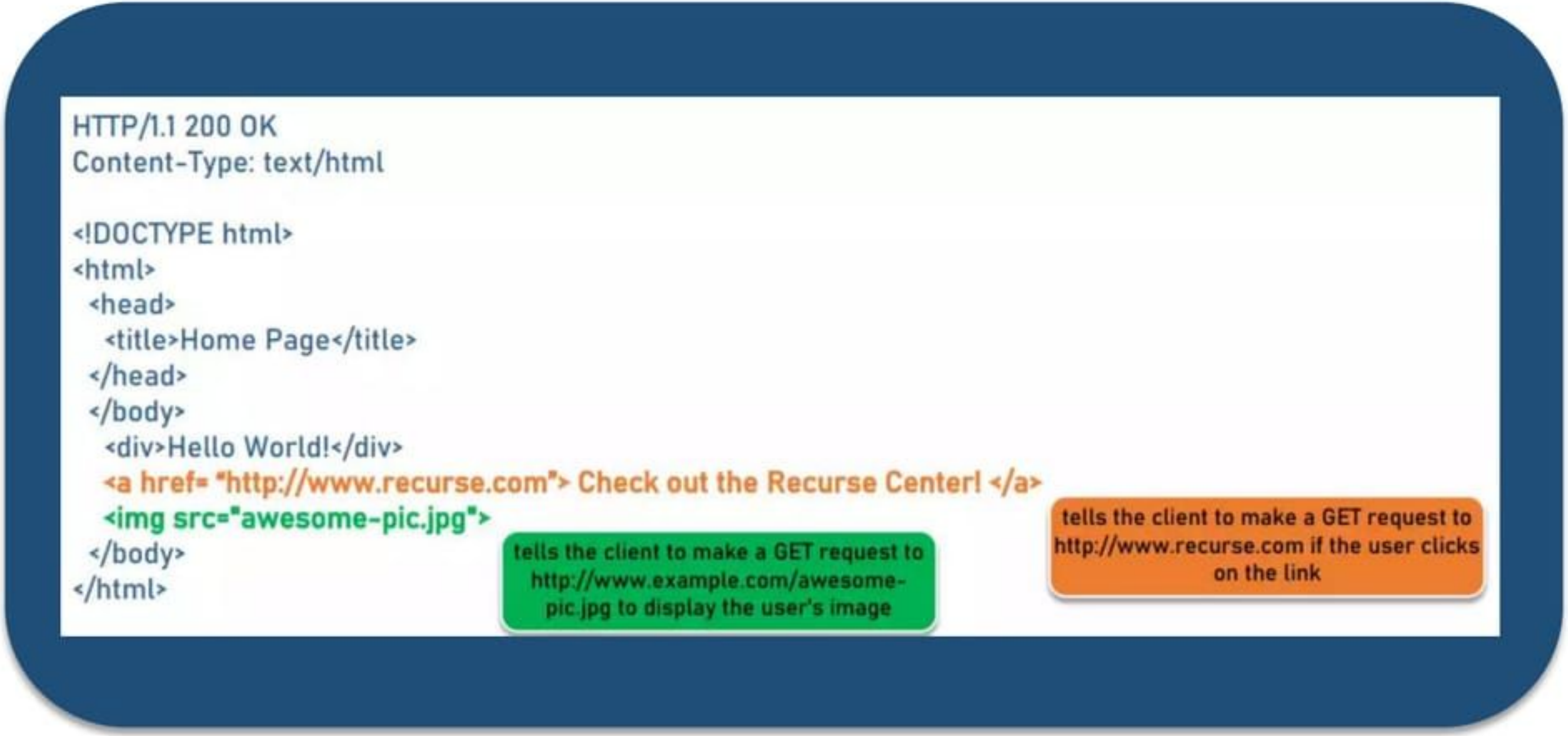
Content-Type: application/x-www-form-urlencoded

Accept: application/json

REQUEST BODY

excludeSensitiveInfo=false&exportPackageName=ProjectPackage.fsproject

REST API RESPONSE WITH HYPERMEDIA



```
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html>
  <head>
    <title>Home Page</title>
  </head>
  </body>
    <div>Hello World!</div>
    <a href= “http://www.recurse.com”> Check out the Recurse Center! </a>
    <img src="awesome-pic.jpg">
  </body>
</html>
```

tells the client to make a GET request to http://www.example.com/awesome-pic.jpg to display the user's image

tells the client to make a GET request to http://www.recurse.com if the user clicks on the link

altexsoft

Example of API in Action

- **User** searches for a product in an e-commerce app.
- The **API Client** (app) sends a GET request to the server.
- The **API Server** returns product details as a **Response**.
- The **Client** displays the results to the user.

API architectural styles

- **REST API**: A simple and widely-used design pattern for communicating over HTTP between applications using standard methods like GET and POST.

- **FastAPI**: A modern, very fast Python framework for building APIs, offering automatic documentation and high performance.

- **GraphQL**: A flexible query language that allows you to request exactly the data you need from a server.

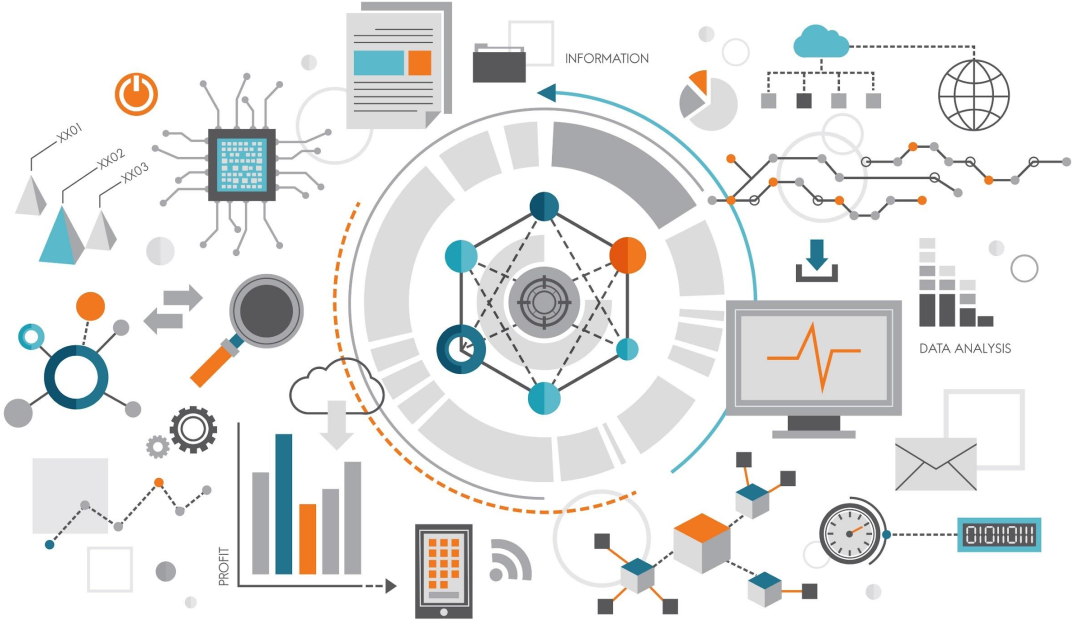
- **gRPC**: A high-performance framework by Google that uses HTTP/2 to efficiently exchange data between services.

API architectural styles

- **SOAP**: An older and reliable protocol that uses XML to transfer data, ideal for sensitive systems like banking.
- **JSON-RPC**: A lightweight protocol based on JSON for performing remote procedure calls between applications.
- **OData**: A protocol that uses HTTP to query and manipulate data in a way similar to SQL.
- **OpenAPI/Swagger**: A standard for documenting and designing REST APIs with an interactive interface for easier understanding and use.

Module II: Packages, APIs, Datasets and Models

Data Sets - Powering Data Science



Data Sources and Licensing

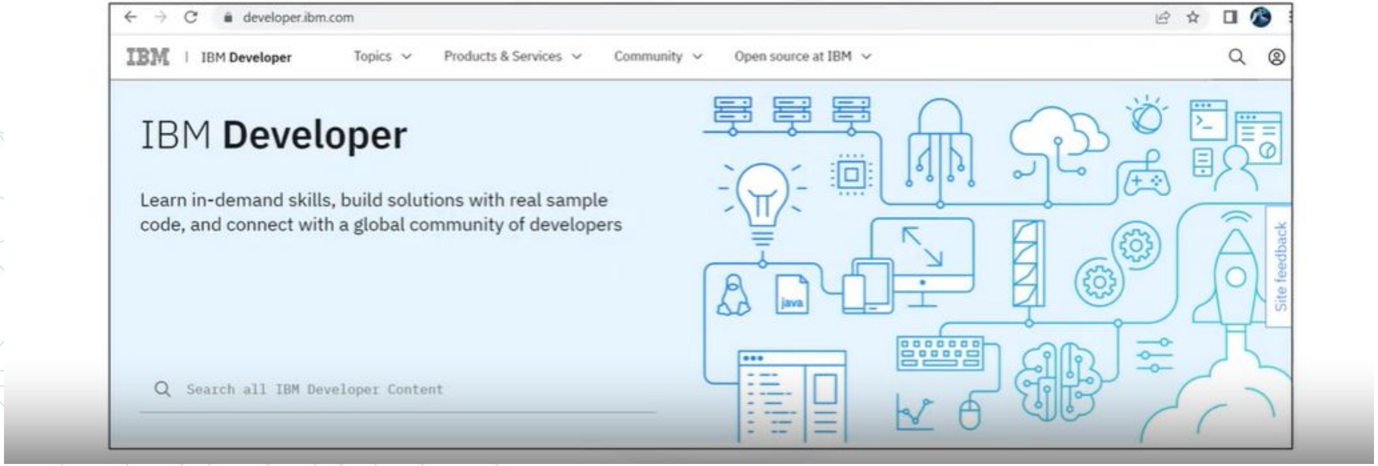
- **Data Sources:** From public entities, organizations, companies, and online communities like Kaggle.
- **Open Data Access:** Facilitated by websites and portals offering a wide range of information.
- **Community Data License Agreement (CDLA):** Addresses licensing terms for open data distribution.
  - **CDLA-Sharing License:** Grants permission to use and modify data with sharing conditions.
  - **CDLA-Permissive License:** Allows data use and modification without sharing changes.
- **Impact on Data Science:** CDLA supports open data sharing without imposing restrictions on derived results.

Introduction to Data Asset eXchange (DAX)

- **Overview of DAX:** IBM's open data repository for high-quality data sets.
- **Purpose of DAX:** Curates open data sets with clear license and usage terms.
- **Data Variety:** Includes images, video, text, and audio data for enterprise applications.
- **Community Data License Agreement (CDLA):** Ensures data sets are available for collaboration.
- **Tutorial Notebooks:** Provide guidance on data cleaning, preprocessing, and analysis.
- **Advanced Notebooks:** Cover complex tasks like machine learning and statistical analysis.

The Data Asset eXchange

https://developer.ibm.com/



IBM **Developer**

Learn in-demand skills, build solutions with real sample code, and connect with a global community of developers

https://developer.ibm.com/exchanges/data/

Navigating Data Asset eXchange (DAX)

- **Accessing DAX:** Visit IBM Developer website and select Data Asset eXchange.
- **Data Exploration:** Explore multiple open data sets on DAX for various purposes.
- **Downloading Data:** Easily download data sets like "NOAA Weather Data - JFK Airport."
- **Notebook Integration:** Use notebooks in Watson Studio for data analysis and processing.
- **Data Files:** Access data files associated with data sets for further analysis.
- **IBM Developer Resources:** DAX and Model Asset eXchange (MAX) available for developers.

Introduction to Watson Studio

- **Collaborative data science platform:** Unites data scientists, developers, and analysts.
- **Build, run, and manage AI models:** Develop and deploy machine learning models.
- **Open-source tools and languages:** Integrates with popular open-source tools and languages.
- **Scalable on cloud or on-premises:** Available on cloud or deployed on your own infrastructure.
- **Automate AI lifecycles:** Streamline the development and deployment of AI models.

https://drive.google.com/file/d/1Xo8U4yud7j_Yb5793LkV9QEtN2abqe6D/view?usp=sharing

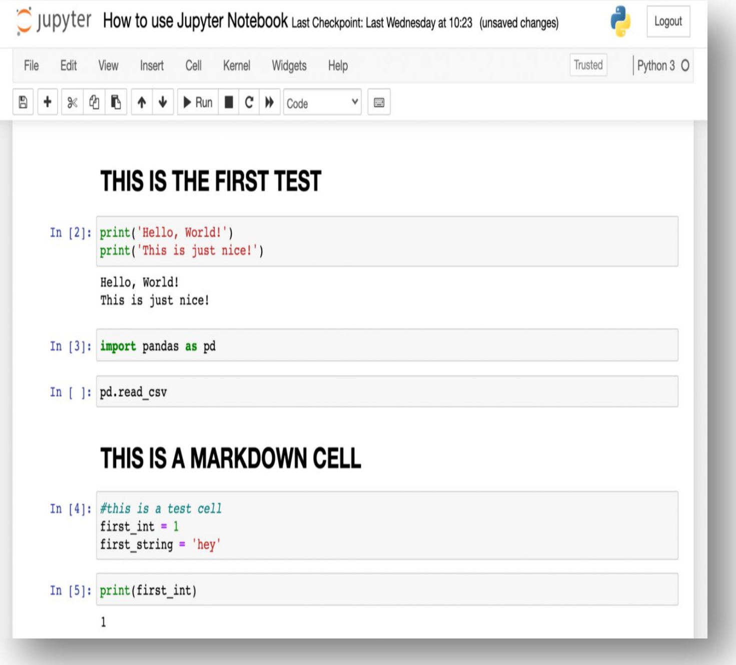
Jupyter Notebooks in Watson Studio

- **Use familiar notebooks for data science:** Leverage Jupyter Notebooks within Watson Studio.
- **Run code, visualize data, and document:** Combine code, results, and explanations in notebooks.
- **Integrate with Watson Studio features:** Access Watson Studio functionalities from notebooks.
- **Collaboration on notebooks:** Share and collaborate on Jupyter Notebooks with your team.
- **Version control for notebooks:** Track changes and revert to previous versions.

Linking GitHub to Watson Studio

- **Integrate Git with Watson Studio:** Manage code using Git within the platform.
- **Version control for your projects:** Track changes to code in your data science projects.
- **Collaboration on GitHub:** Leverage GitHub's features for collaboration and code sharing.
- **Streamlined development workflow:** Integrate development and deployment processes.
- **Open-source project management:** Facilitate collaboration on open-source data science projects.

Module III
Anaconda
Jupyter Notebooks and
Jupyter lab


jupyter
How to use Jupyter Notebook Last Checkpoint: Last Wednesday at 10:23 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

THIS IS THE FIRST TEST

```
In [2]: print('Hello, World!')
        print('This is just nice!')

        Hello, World!
        This is just nice!

In [3]: import pandas as pd

In [ ]: pd.read_csv
```

THIS IS A MARKDOWN CELL

```
In [4]: #this is a test cell
        first_int = 1
        first_string = 'hey'

In [5]: print(first_int)

        1
```

Introduction to Jupyter Notebooks

- **Interactive notebooks:** Combine code, explanations, visualizations in one document.
- **Rapid prototyping & exploration:** Prototype ideas, explore data interactively.
- **Share & collaborate:** Share notebooks for easy explanation and collaboration.
- **Teaching and learning:** Effective tool for teaching and learning data science concepts.
- **Open-source and language-agnostic:** Free to use, supports various programming languages.

Getting Started with Jupyter

- **Easy installation:** Use conda package manager for straightforward installation.
- **Create notebooks:** Create new Jupyter notebooks for your specific projects.
- **Run code interactively:** Execute code cells and see results displayed directly below.
- **Rich outputs:** Visualize data and get textual output within the notebook.
- **Web-based interface:** Access notebooks from any device with a web browser.

Jupyter Kernels

- **Execution engines for code:** Kernels interpret and run code for specific languages.
- **Selecting the right kernel:** Choose the kernel that matches your programming language.
- **Support for various languages:** Run Python, R, Julia, and other languages in Jupyter.
- **Language specific functionalities:** Leverage functionalities specific to each language.
- **Kernel connection:** Jupyter communicates with the kernel to execute code.

Jupyter Architecture

- **Three core components:** Notebook interface, execution kernel, and user interface.
- **Collaborative workflow:** Code goes from notebook to kernel for execution, then results back to notebook.
- **Interactive environment:** Enables interactive development and analysis.
- **Open architecture:** Allows for customization and extension of functionalities.
- **Web technology foundation:** Built on web technologies for broad browser compatibility.

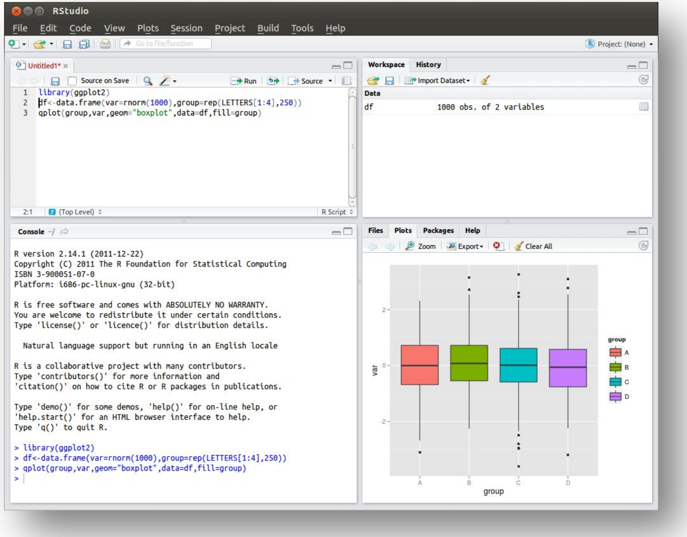
Additional Anaconda Jupyter Environments

- **Isolate project dependencies:** Use conda environments to manage dependencies for each project.
- **Version control:** Manage different versions of libraries for specific projects.
- **Reproducible research:** Ensure consistent environments for replicating research results.
- **Switching between environments:** Easily switch between conda environments for different projects.
- **Environment sharing:** Share conda environments for collaboration with specific setups.

Additional Cloud Based Jupyter Environments

- **Run notebooks on the cloud:** Launch Jupyter notebooks on cloud platforms for easy access.
- **Scalable resources:** Access powerful computing resources on the cloud for large datasets.
- **Collaboration from anywhere:** Collaborate on notebooks from any device with an internet connection.
- **Cost-effective option:** Pay only for the resources you use on cloud platforms.
- **Variety of cloud providers:** Choose a cloud provider that best suits your needs.

Module IIII
GitHub



```
library(ggplot2)
df<-data.frame(var=rnorm(1000),group=rep(LETTERS[1:4],250))
qplot(group,var,geom="boxplot",data=df,fill=group)
```

Data

df 1000 obs. of 2 variables

```
R version 2.14.1 (2011-12-22)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i686-pc-linux-gnu (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(ggplot2)
> df<-data.frame(var=rnorm(1000),group=rep(LETTERS[1:4],250))
> qplot(group,var,geom="boxplot",data=df,fill=group)
> 
```

Introduction to R and RStudio

- **R Programming Language:** Open-source language for statistical computing and graphics.
- **RStudio IDE:** Integrated development environment for working with R.
- **RStudio functionalities:** Code editing, data visualization, debugging, and more.
- **Benefits of using RStudio:** User-friendly interface for efficient R development.
- **Focus on data science tasks:** Streamline data analysis and visualization workflows.

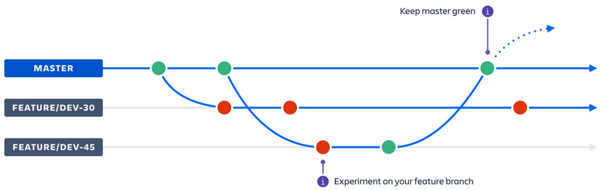
Plotting in RStudio

- **R's rich graphics capabilities:** Create various plots for data exploration and presentation.
- **Customization options:** Extensive control over plot appearance and elements.
- **Interactive graphics:** Create dynamic plots that respond to user interaction.
- **Integration with RStudio:** Utilize RStudio's features for creating and managing plots.
- **Exporting plots:** Save plots as images or other formats for sharing or reports.

Overview of Git/GitHub

- **Version control system (Git):** Track changes in code and data over time.
- **Collaboration platform (GitHub):** Host code repositories and collaborate with others.
- **Benefits of using Git/GitHub:** Version control, collaboration, and code sharing.
- **Open-source development:** Facilitate open-source software development workflows.
- **Improved project management:** Track project progress and manage different versions.

Basic Workflow



Introduction to GitHub

- **Online platform for hosting code:** Create repositories to store and manage your code.
- **Version control with Git:** Track changes and revert to previous versions if needed.
- **Collaboration features:** Share code with others, discuss changes, and work together.
- **Public or private repositories:** Choose to make your code publicly accessible or private.
- **Integrates with various tools:** Works with Git and other developer tools.

GitHub Repositories

- **Fundamental unit of code storage:** Organize your code projects in repositories.
- **Storing code and data:** Store code files, data files, and other project assets.
- **Version history:** Track changes made to files over time.
- **Branching and merging:** Create separate development branches and merge them back.
- **Collaboration features:** Manage access control, track issues, and collaborate on code.

Questions & Answers

Thank you!