

IBM

# Tools for Data Science

- **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

# Module I

## Overview of Data Science Tools

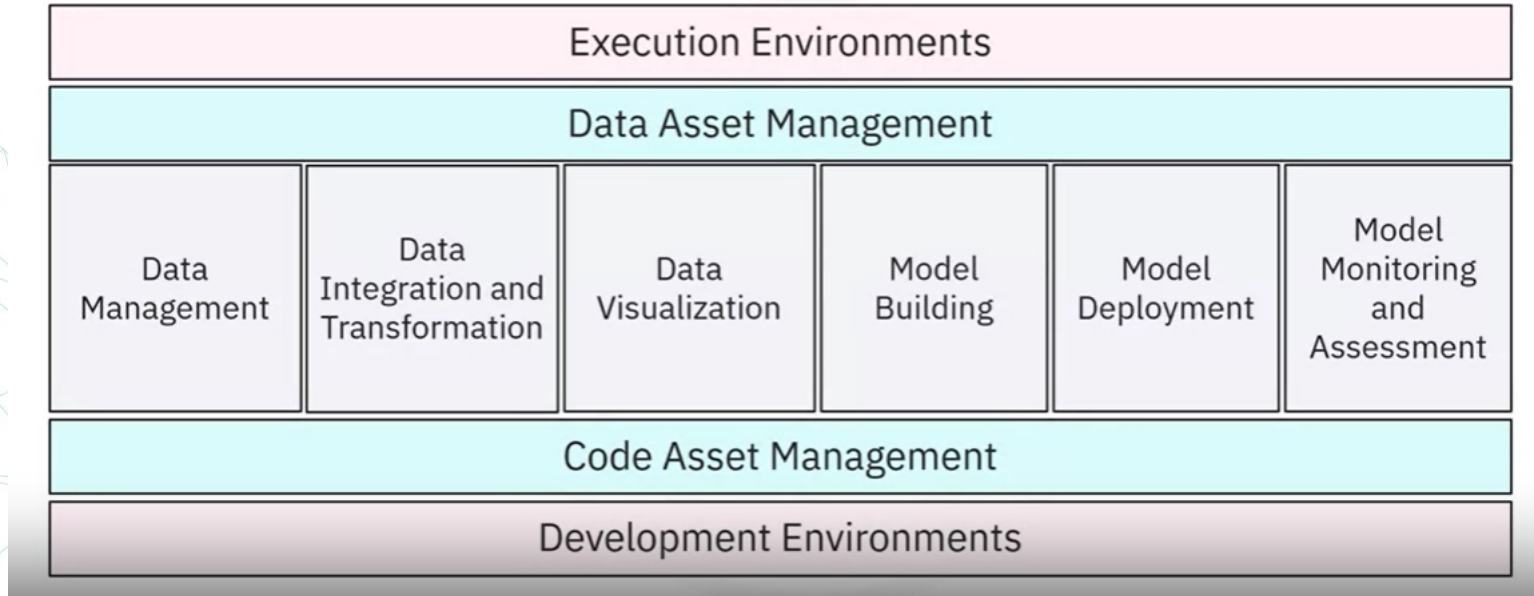


# Module I: Overview of Data Science Tools

## *Categories of Data Science Tools*



# Data Science categories



# 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

## *Open-Source Tools for Data Science*





# Open-Source Data Management, Integration, and Visualization Tools

- **Data Management Tools (Relational Databases):**

- MySQL, PostgreSQL
- MongoDB, Apache CouchDB, Apache Cassandra
- Hadoop File System, Ceph, Elastic Search



- **Data Integration and Transformation Tools:**

- Termed: **Data Refinery and Cleansing**
- Apache AirFlow, KubeFlow
- Apache Kafka, Apache Nifi, Apache SparkSQL
- NodeRED



- **Data Visualization Tools:**

- PixieDust, Hue
- Kibana
- Apache Superset



## Model Tools, Deployment, and Monitoring

- **Model Tools for Building, Deployment, and Monitoring:**
  - Apache PredictionIO, Seldon
  - Kubernetes, Redhat OpenShift, MLeap
  - TensorFlow service, TensorFlow lite, TensorFlow dot JS
- **Model Monitoring Tools:**
  - ModelDB, Prometheus
  - IBM AI Fairness 360, IBM Adversarial Robustness 360 Toolbox
  - IBM AI Explainability 360
- **Code and Data Asset Management Tools:**
  - Git, GitHub, GitLab, Bitbucket
  - Apache Atlas, ODPi Egeria, Kylo



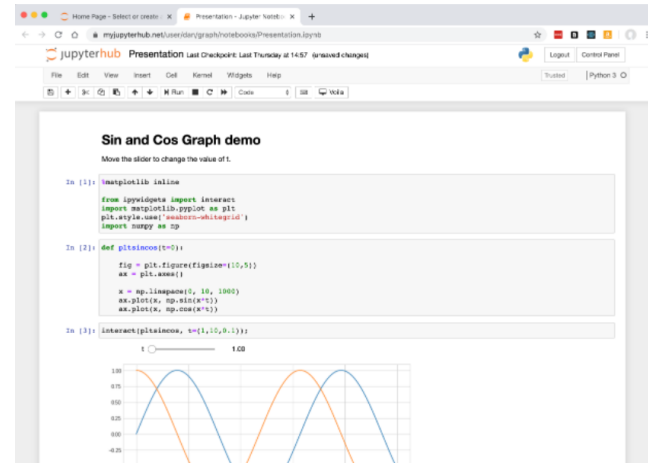
# Development Environments and Execution Environments

- **Development Environments:**

- Jupyter Notebook, Jupyter Lab, Apache Zeppelin
- RStudio, Spyder

- **Execution Environments:**

- 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*



# 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)*



## Understanding APIs

- **Definition of API:** Facilitates communication between software components.
- **API:** Examples include Pandas and TensorFlow with multilingual support.
- **Backend Communication:** APIs abstract backend complexities for diverse language support.
- **REST API Basics:** (Representational State Transfer) API is an architectural style for designing networked applications. It enables communication between clients and servers over the internet using standard HTTP methods (such as GET, POST, PUT, DELETE) to perform operations on resources.
- **Request-Response Model:** Clients send requests and receive responses via HTTP methods.
- **Data Transmission:** JSON files transmit instructions and information between client and server.

## API Functionality and Usage

- **REST API Advantages:** Facilitates internet-based communication for resource utilization.
- **Client-Server Interaction:** Clients access services via endpoints for data transmission.
- **Common API Terms:** Client, web service, endpoint, HTTP methods, and JSON format.
- **Example APIs:** Watson Text to Speech for audio-to-text conversion, Watson Language Translator for translation tasks.
- **API Communication:** POST and GET requests for data transmission and retrieval.
- **API Applications:** Transcription, translation, and other functionalities for software integration.



# Module II: Packages, APIs, Datasets and Models

## *Data Sets - Powering Data Science*



## Understanding Data Sets

- **Definition of Data Set:** Structured collection of information, including text, numbers, and media.
- **Tabular Data Sets:** Comprise rows and columns, often in CSV format.
- **Hierarchical and Network Data:** Represent relationships in tree or graph formats.
- **Raw Data Files:** Include images or audio, such as the MNIST dataset for image processing.
- **Data Ownership:** Traditionally private but now available as open data for public use.
- **Open Data Impact:** Enables insights, applications, research, and supports data science growth.

## 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.

# Module II: Packages, APIs, Datasets and Models

## *Sharing Enterprise Data - Data Asset eXchange*



## 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.

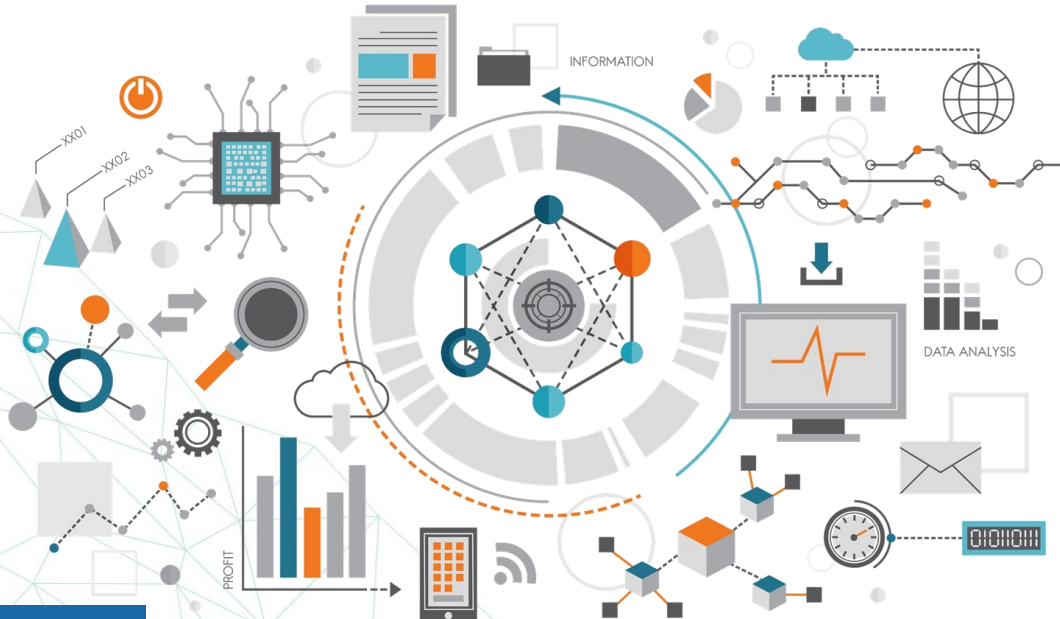


## Navigating Data Asset eXchange (DAX)

- **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.

# Module II: Packages, APIs, Datasets and Models

## *Machine Learning Models Learning from Models to Make Predictions*





# Introduction to Machine Learning Models

**Definition of Machine Learning Models:** Algorithms that identify data patterns.

**Model Training Process:** Learning patterns from data through algorithms.

**Types of Machine Learning:** Supervised, Unsupervised, and Reinforcement Learning.

- **Supervised Learning Models:** Regression (predict numeric values) and Classification (predict categories).
- **Unsupervised Learning Models:** Clustering (grouping similar data) and Anomaly Detection (identifying outliers).
- **Reinforcement Learning:** Learning through trial and error, based on rewards.

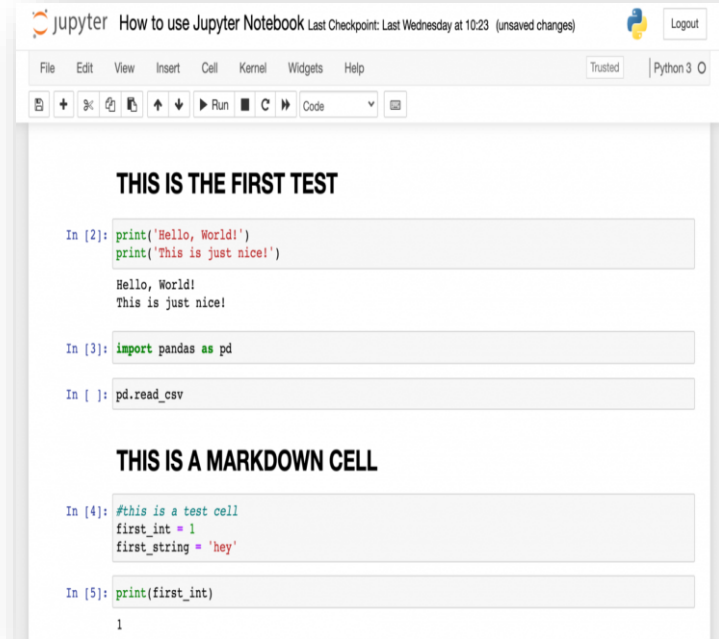
## Deep Learning and Model Implementation

- **Deep Learning Overview:** Emulates human brain for solving complex problems.
- **Deep Learning Applications:** Natural language processing, image analysis, time series forecasting.
- **Training Deep Learning Models:** Requires large labeled datasets and specialized hardware.
- **Implementation Frameworks:** TensorFlow, PyTorch, Keras for building and training models.
- **Model Repositories:** Access pre-trained models from TensorFlow, PyTorch, Keras, ONNX.
- **Building a Custom Model:** Collect, prepare data, label, select or build a model, train, analyze, and deploy.

# Module III

## Jupyter Notebooks and Jupyter lab

7/5/2025



The screenshot shows a Jupyter Notebook window titled "How to use Jupyter Notebook". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and saving. The notebook contains the following cells:

- A markdown cell with the text: **THIS IS THE FIRST TEST**
- A code cell (In [2]:) containing:

```
print('Hello, World!')  
print('This is just nice!')
```

The output of this cell is:

```
Hello, World!  
This is just nice!
```
- A code cell (In [3]:) containing:

```
import pandas as pd
```
- A code cell (In [ ]:) containing:

```
pd.read_csv
```
- A markdown cell with the text: **THIS IS A MARKDOWN CELL**
- A code cell (In [4]:) containing:

```
#this is a test cell  
first_int = 1  
first_string = 'hey'
```
- A code cell (In [5]:) containing:

```
print(first_int)
```

The output of this cell is:

```
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.

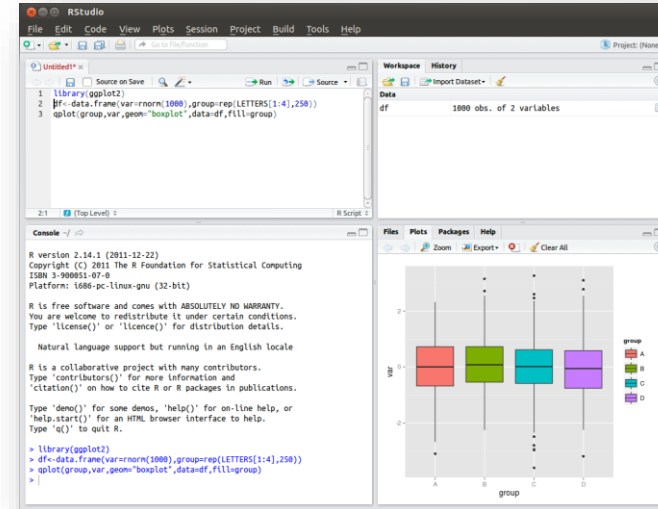


## Additional Cloud Based Jupyter Environments

- **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 III

## RStudio & GitHub



## 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.

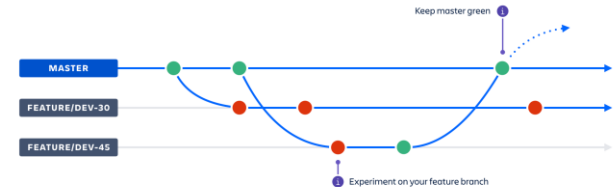
## Plotting in RStudio

- **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.

# Module IIII

## IBM Watson Studio





## 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.

## 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.

# Questions & Answers



# Thank you!