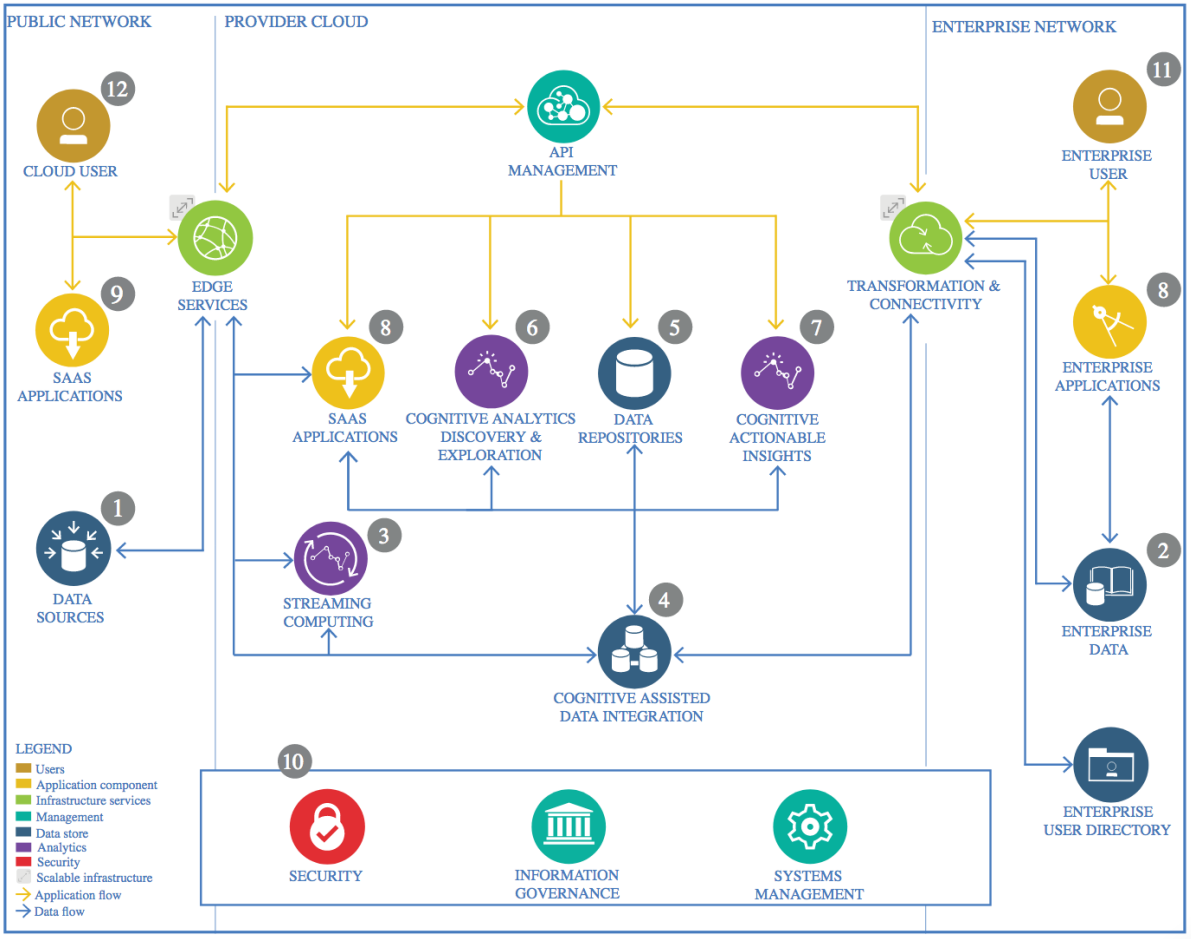
The Lightweight IBM Cloud Garage Method for Data Science

Architectural Decisions Document Template

1 Architectural Components Overview



IBM Data and Analytics Reference Architecture. Source: IBM Corporation

1.1 Data Source

1.1.1 Technology Choice

1. **Python** with Pandas NumPy, and matplotlib for data manipulation, visualization and analysis.

2. Leverage **Kaggle** as the primary source for the dataset.

3. **Jupyter Notebook** for interactive development and visualization.

1.1.2 Justification

Python: Python is a widely-used language for data science and machine learning, providing extensive libraries such as Pandas and NumPy for efficient data manipulation, Keras for deep learning and .

Kaggle Dataset: Kaggle datasets are often well-documented and diverse, making them suitable for experimentation. The dataset is crucial for training and evaluating the fraud detection model.

Jupyter Notebook: Jupyter Notebooks provide an interactive environment for iterative development and visualization, which is crucial for exploring and understanding patterns in the data.

1.2 Enterprise Data

1.2.1 Technology Choice

Local Storage: Utilize local storage or a traditional file system to store static datasets. This can be achieved using the file I/O capabilities in Python or other programming languages.

1.2.2 Justification

Local Storage: Since your data is static and doesn't require real-time processing, using local storage or a traditional file system is a simpler and more straightforward solution. You can read and manipulate the data directly from your local environment without the need for cloud services, streaming processing, or specialized databases.

1.3 Streaming analytics

1.3.1 Technology Choice

Pandas for Batch Processing: Continue to use Pandas for batch processing and analyzing static data.

1.3.2 Justification

Batch Processing with Pandas: Since my data is static and not streaming in real-time, Pandas remains a suitable choice for batch processing. Pandas is a powerful library for data manipulation, analysis, and statistics in a static environment. It allows you to perform in-depth exploratory data analysis, calculate various statistics for each feature, generate the variance matrix, assess skewness, and standardize the data.

1.4 Data Integration

1.4.1 Technology Choice

Pandas and Python Scripts: Utilize Pandas and custom Python scripts for data integration, transformation, and cleaning.

1.4.2 Justification

Flexibility of Pandas: Pandas provides a flexible and powerful framework for data manipulation and integration. It allows you to easily handle various data formats, merge datasets, and perform transformations, making it suitable for integrating different sources of data.

1.5 Data Repository

1.5.1 Technology Choice

I have created a GitHub repository to host my fraud detection data science project. The dataset used in the project was downloaded from Kaggle and is included in the repository. The repository serves as a centralized location for storing and managing all project-related files, including Jupyter Notebooks, data files, and documentation.

1.5.2 Justification

GitHub was chosen as the version control and collaboration platform. It facilitates easy collaboration with others, enables version tracking, and provides a platform for showcasing the project to the wider community. Additionally, the use of GitHub allows for seamless integration with other tools and services commonly used in the data science community.

1.6 Discovery and Exploration

1.6.1 Technology Choice

Jupyter Notebooks with Python (Using Libraries such as Pandas, Matplotlib, Seaborn): Leverage Jupyter Notebooks as an interactive environment for data exploration, using Python along with Pandas for data manipulation and analysis, and Matplotlib/Seaborn for visualization.

1.6.2 Justification

For this project, I chose to use Jupyter Notebooks for the development and documentation of the data science workflow. Jupyter Notebooks provide an interactive and shareable environment, allowing for a step-by-step exploration of the data, feature engineering, model development, and evaluation. The primary programming language used is Python, leveraging popular libraries such as pandas, scikit-learn, and matplotlib for data manipulation, machine learning modeling, and visualization.

1.7 Actionable Insights

1.7.1 Technology Choice

Matplotlib and Seaborn: Leverage Matplotlib and Seaborn within Jupyter Notebooks for creating visualizations directly in your analysis.

1.7.2 Justification

These libraries are well-integrated with Jupyter and offer a wide range of plotting options.

1.8 Applications / Data Products

1.8.1 Technology Choice

Programming Language: Python

Jupyter Notebooks for Prototyping

Machine Learning Frameworks: Scikit-Learn, TensorFlow

1.8.2 Justification

1. Python is widely used in the data science and machine learning community. Its extensive libraries and frameworks, such as Scikit-Learn, TensorFlow, and PyTorch, provide robust tools for building and deploying machine learning models. Python's readability and versatility make it a suitable choice for developing various components of the fraud detection system.

2. Jupyter Notebooks facilitate interactive and exploratory data analysis. They are well-suited for prototyping machine learning models and experimenting with different algorithms. Their integration with Python and support for visualizations make them a valuable tool for understanding and iterating on the fraud detection model.

3. These frameworks provide a comprehensive set of tools for building, training, and evaluating machine learning models. Scikit-Learn is excellent for traditional machine learning algorithms, while TensorFlow and PyTorch are powerful for deep learning models. The choice depends on the complexity and requirements of the fraud detection model.

1.9 Security, Information Governance and Systems Management

1.9.1 Technology Choice

Since the features are anonymized, the sensitive information that might require additional security measures is not present in its original form.