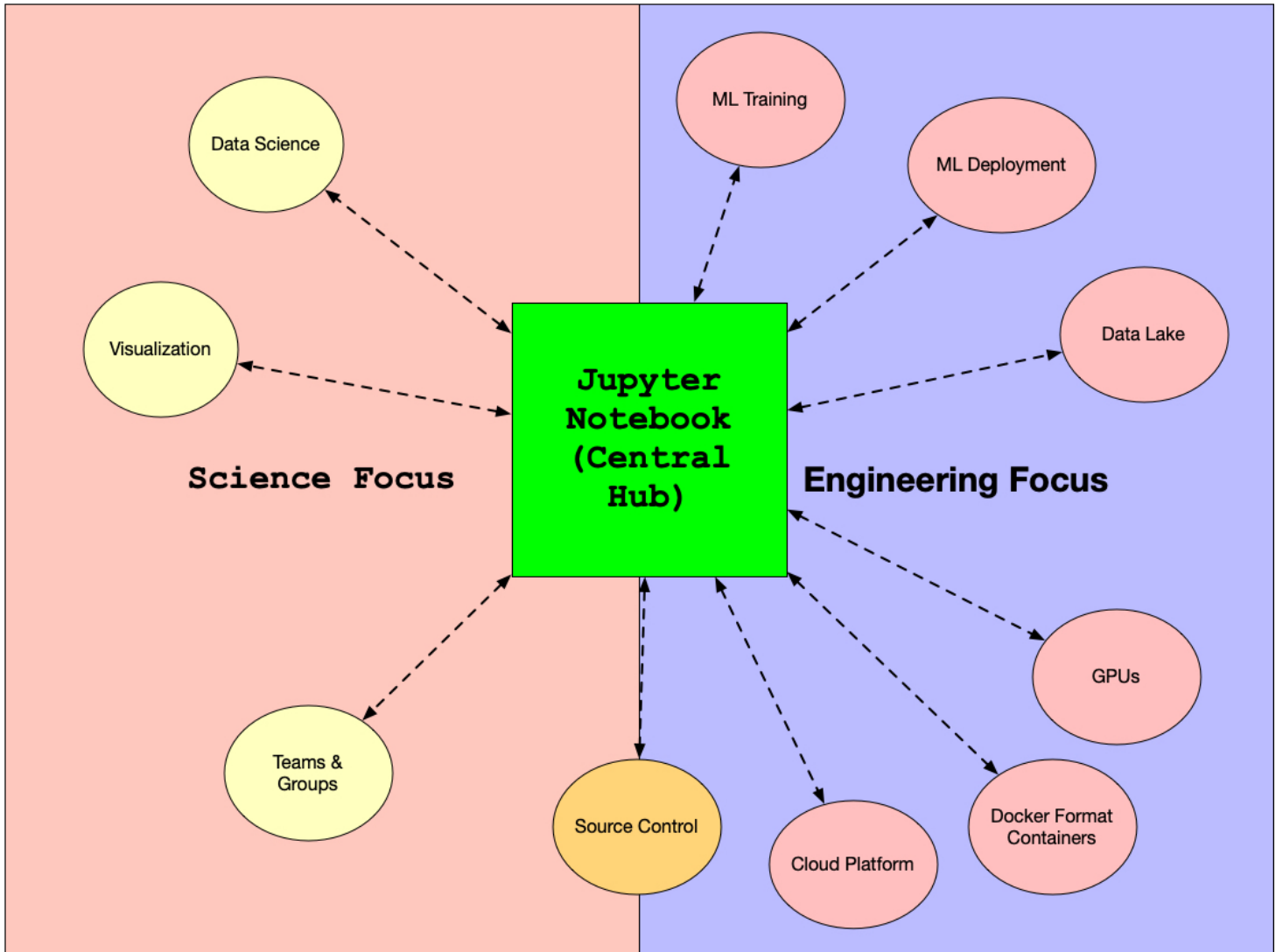


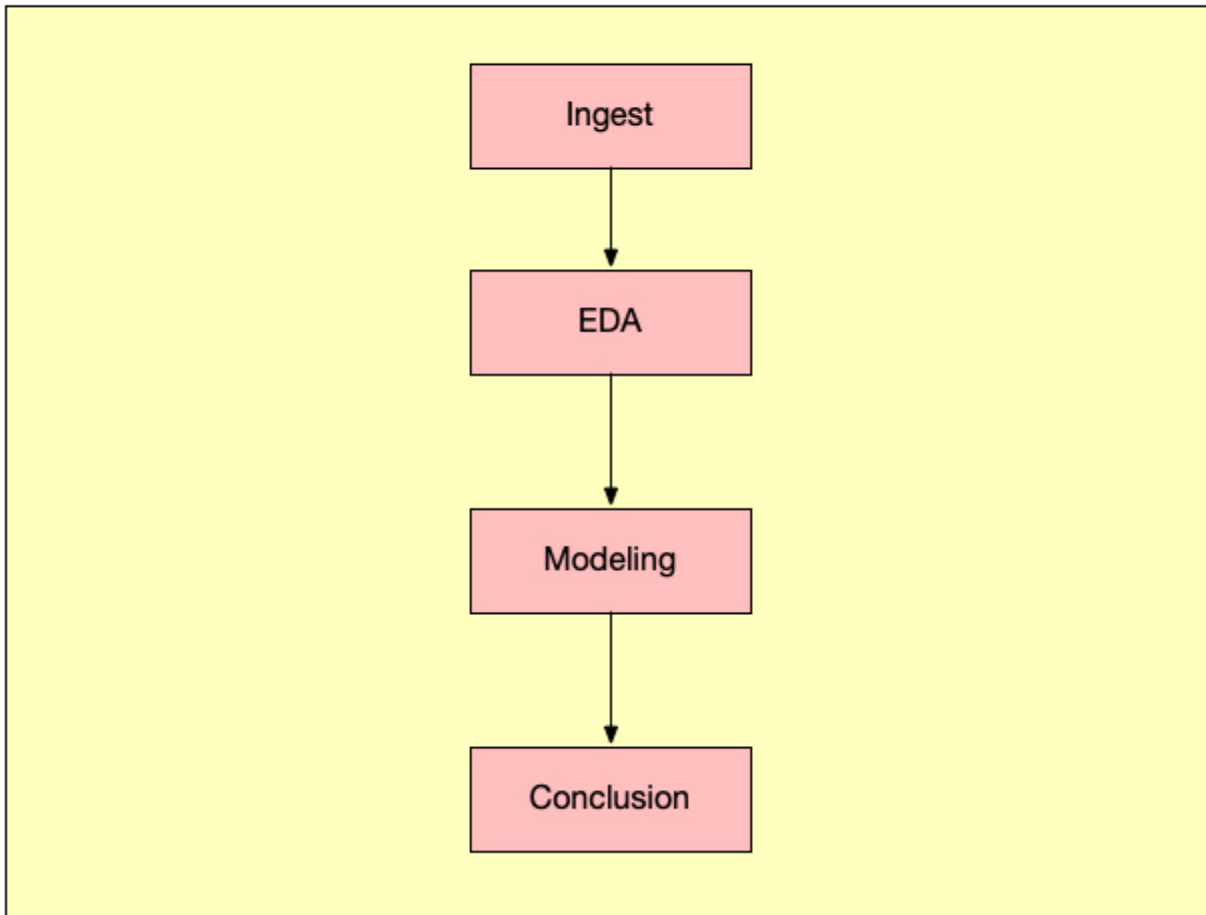
## Jupyter Notebook Workflow

Jupyter notebooks are increasingly the hub in both Data Science and Machine Learning projects. All major vendors have some form of Jupyter integration. Some tasks orient in the direction of engineering, and others in the order of science.



An excellent example of a science-focused workflow is the traditional notebook based Data Science workflow. Data is collected; it could be anything from a SQL query to a CSV file hosted in Github. Next, the EDA (exploratory data analysis) using visualization, statistics, and unsupervised machine learning. Finally, an ML model, and then a conclusion.


## Data Science Notebook Workflow



This methodology often fits very well into a markdown based workflow where each section is a Markdown heading. Often that Jupyter notebook is in source control. Is this notebook source control or a document? This distinction is an important consideration, and it is best to treat it as both.

[Code](#) [Issues 0](#) [Pull requests 0](#) [Actions](#) [Projects 0](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Branch: master [devops-for-datascience / data\\_science\\_workflow.ipynb](#) [Find file](#) [Copy path](#)

 **noahgift** Data Science Workflow 759049d now  
1 contributor

97 lines (97 sloc) 1.66 KB [Code](#) [Raw](#) [Blame](#) [History](#) [Open in IDE](#) [Download](#)

# Ingest

# EDA

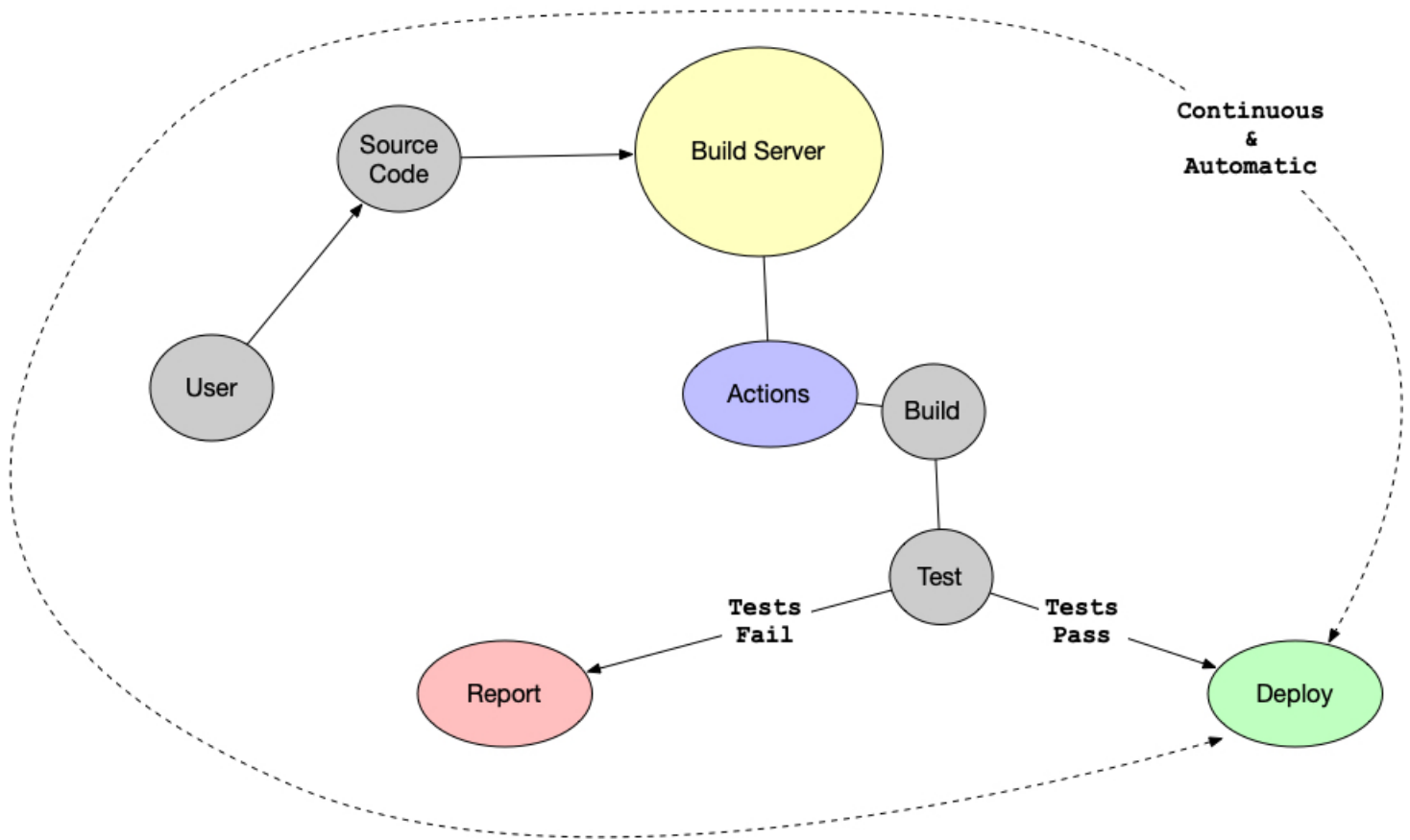
# Modeling

# Conclusion

## DevOps for Jupyter Notebooks

DevOps is a popular technology best practice, and it is often used in combination with Python. The center of the universe for DevOps is the build server. This guardian of the software galaxy facilitates automation. This automation includes linting, testing, reporting, building, and deploying code. This process is called continuous delivery.

## DevOps Workflow



The benefits of continuous delivery are many. First, tested code is always in a deployable state. Automation of best practices then creates a cycle of constant improvement in a software project. A question should crop up if you are a data scientist. Isn't Jupyter notebook source code too? Wouldn't it benefit from these same practices? The answer is yes.

This diagram exposes a proposed best practices directory structure for a Jupyter based project in source control. The `Makefile` holds the recipes to build, run and deploy the project via make commands: `make test`, etc. The `Dockerfile` contains the actual runtime logic, which makes the project genuinely portable.

```

FROM python:3.7.3-stretch

# Working Directory
WORKDIR /app

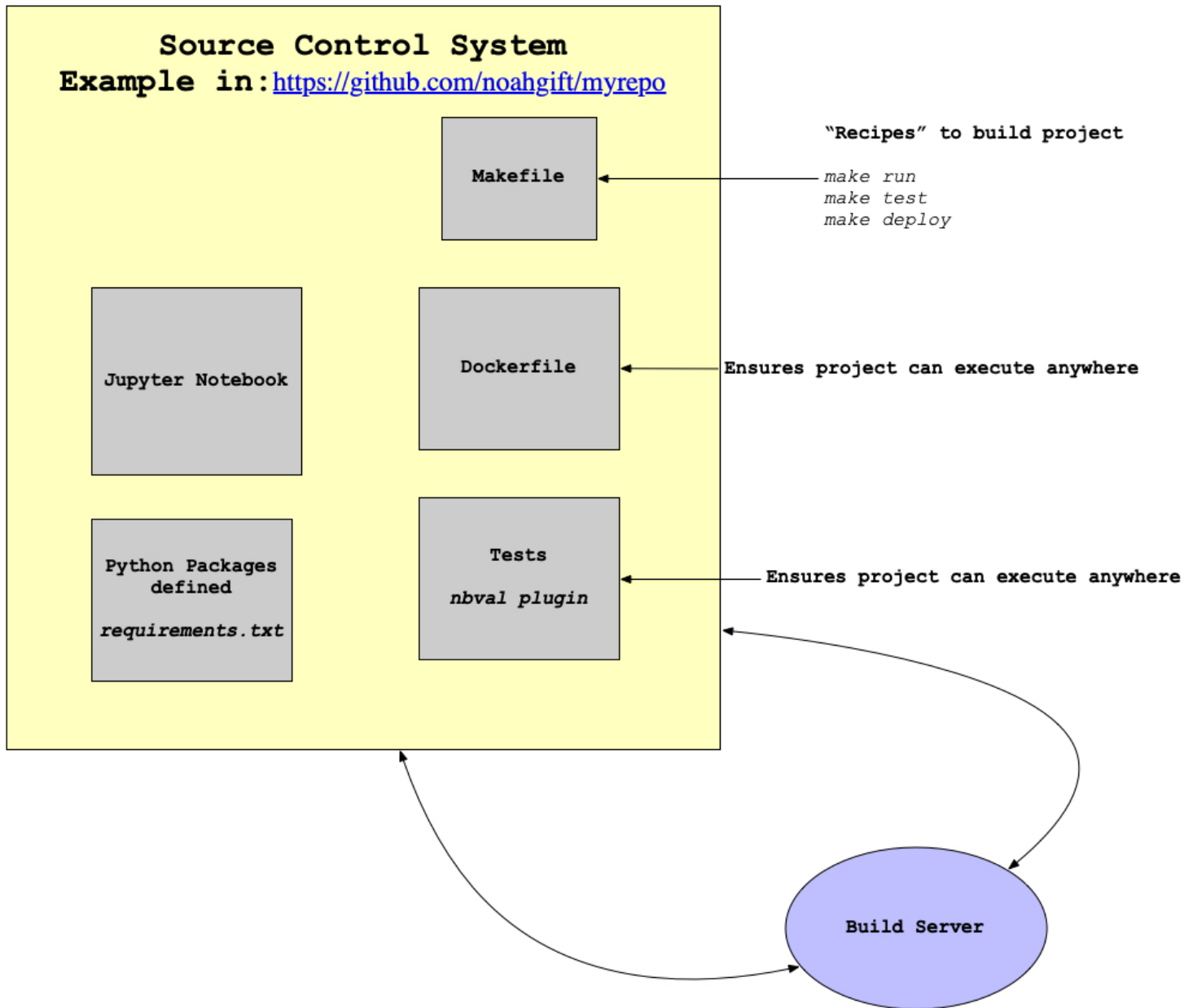
# Copy source code to working directory
COPY . app.py /app/

# Install packages from requirements.txt
# hadolint ignore=DL3013
RUN pip install --upgrade pip &&\
    pip install --trusted-host pypi.python.org -r requirements.txt

# Logic to run Jupyter could go here...
# Expose port 8888
#EXPOSE 8888
  
```

```
# Run app.py at container launch
#CMD ["jupyter", "notebook"]
```

## DevOps for Jupyter Workflow



The Jupyter notebook itself can be tested via the `nbval` plugin as shown.

```
python -m pytest --nbval notebook.ipynb
```

The requirements for the project are in a `requirements.txt` file. Every time the project is changed, the build server picks up the change and runs tests on the Jupyter notebook cells themselves.

DevOps isn't just for software-only projects. DevOps is a best practice that fits well with the ethos of Data Science. Why guess if your notebook works, your data is reproducible or that it can deploy?

# AWS Sagemaker Overview

One of the most prevalent managed ML Systems is AWS Sagemaker. This platform is a complete solution for an organization that wants to build and maintain large-scale Machine Learning projects. Sagemaker makes heavy use of the concept of MLOPs(Machine Learning Operations).