

# REPRODUCIBLE WORKFLOWS

Version Control and Computational Notebooks

John Little 

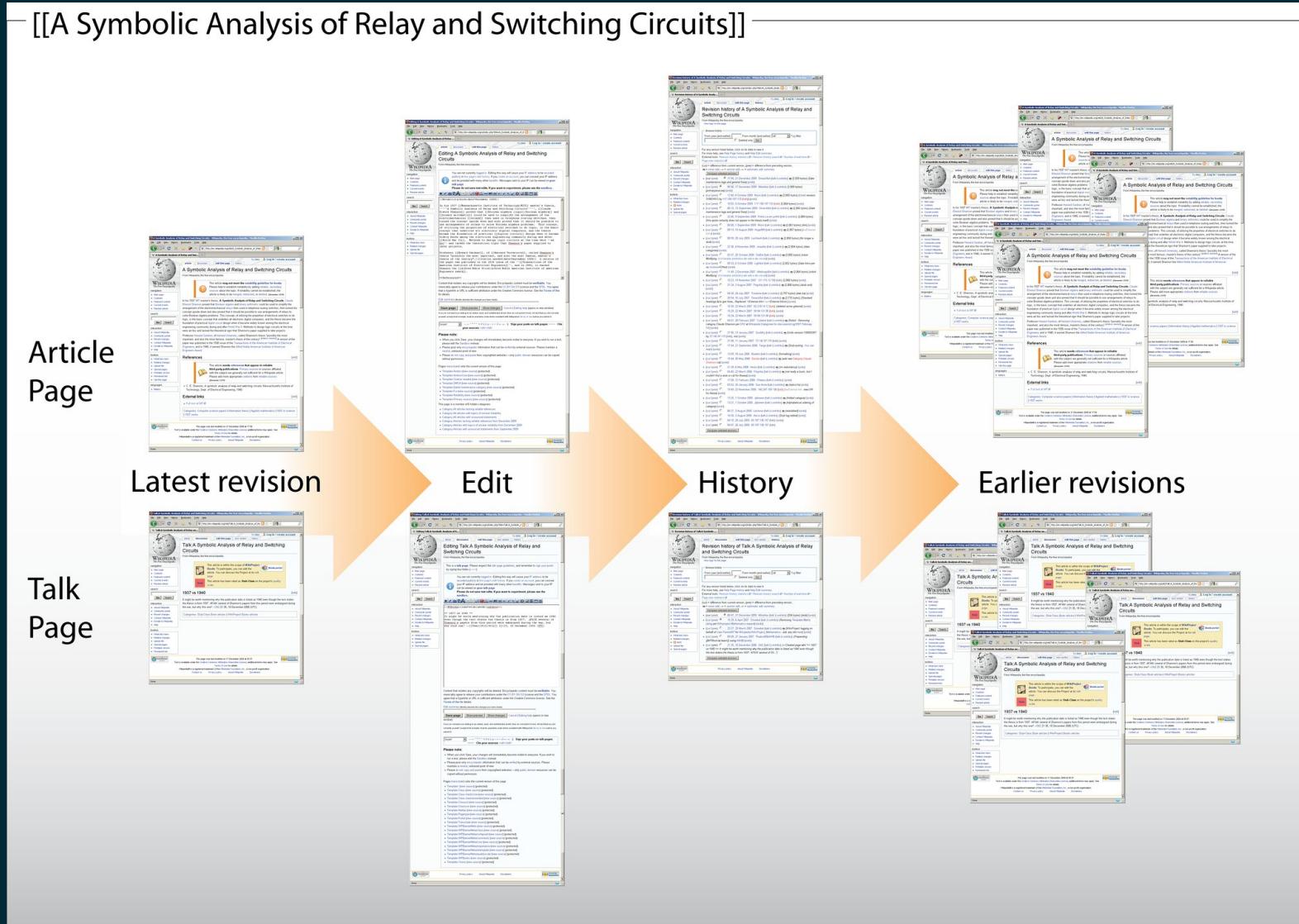
Duke University Libraries

Center for Data & Visualization Sciences

2024-02-12

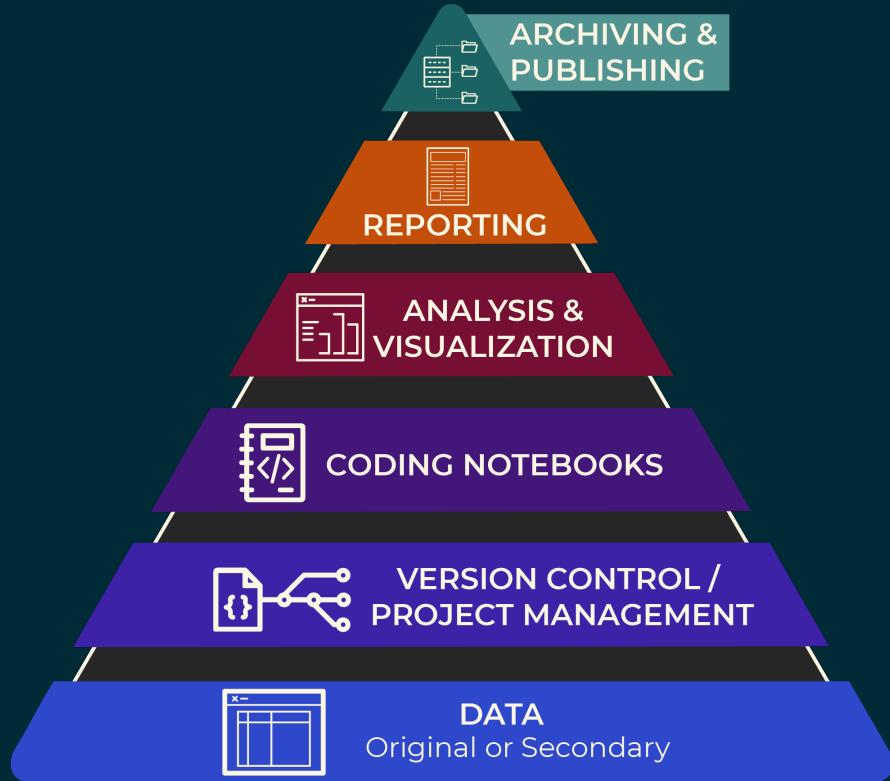


# ARTICLE PRODUCTION



# REPRODUCTION

Authoring and computation environment should enable the articulation of scholarship within a reproducible context



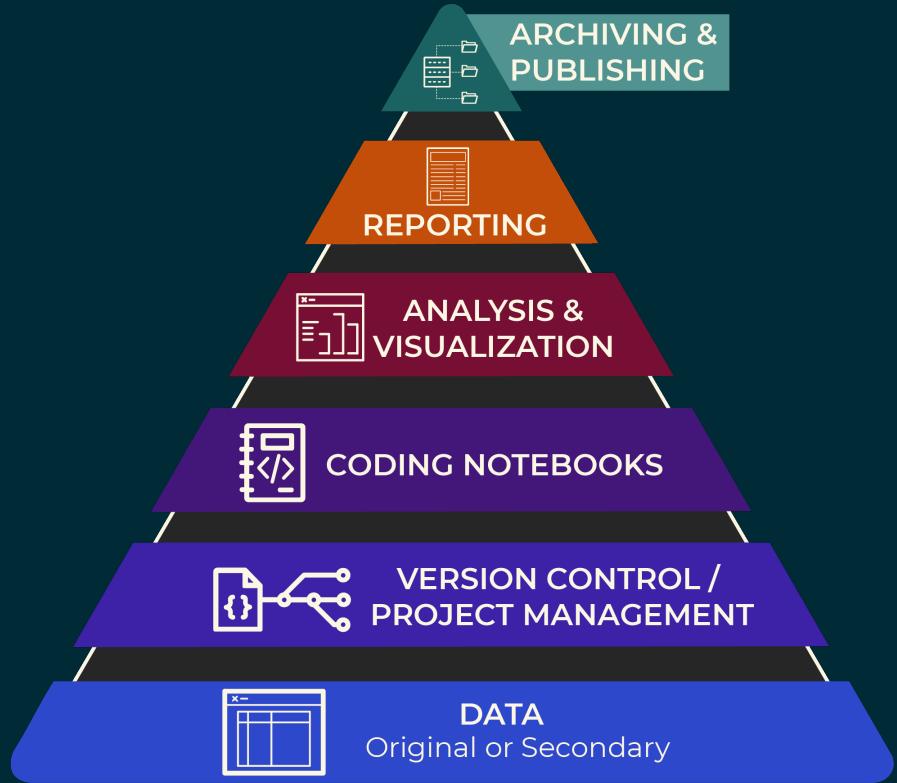
Reproducibility Pyramid • Little & Lafferty-Hess (2020)

# FEATURES

- Support composable recombination
- Accommodate multimedia expression
- Provide rich reporting expressions
- Support economical portability and degrade gracefully
- Support extensibility
- Ensure transparency
- Support a documentary-style project history
- Accommodate change and collaboration
- Be citable

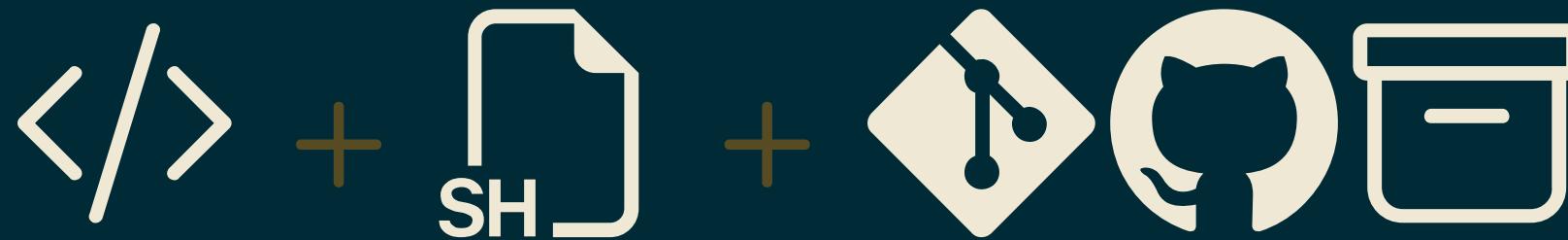
# THREE POINTS

1. Version Control (Git & GitHub)
2. Notebooks (Literate Coding)
3. Archiving & Publishing (Zenodo, Containers)



Reproducibility Pyramid • Little & Lafferty-Hess (2020)

# PREVIEW



# VERSION CONTROL



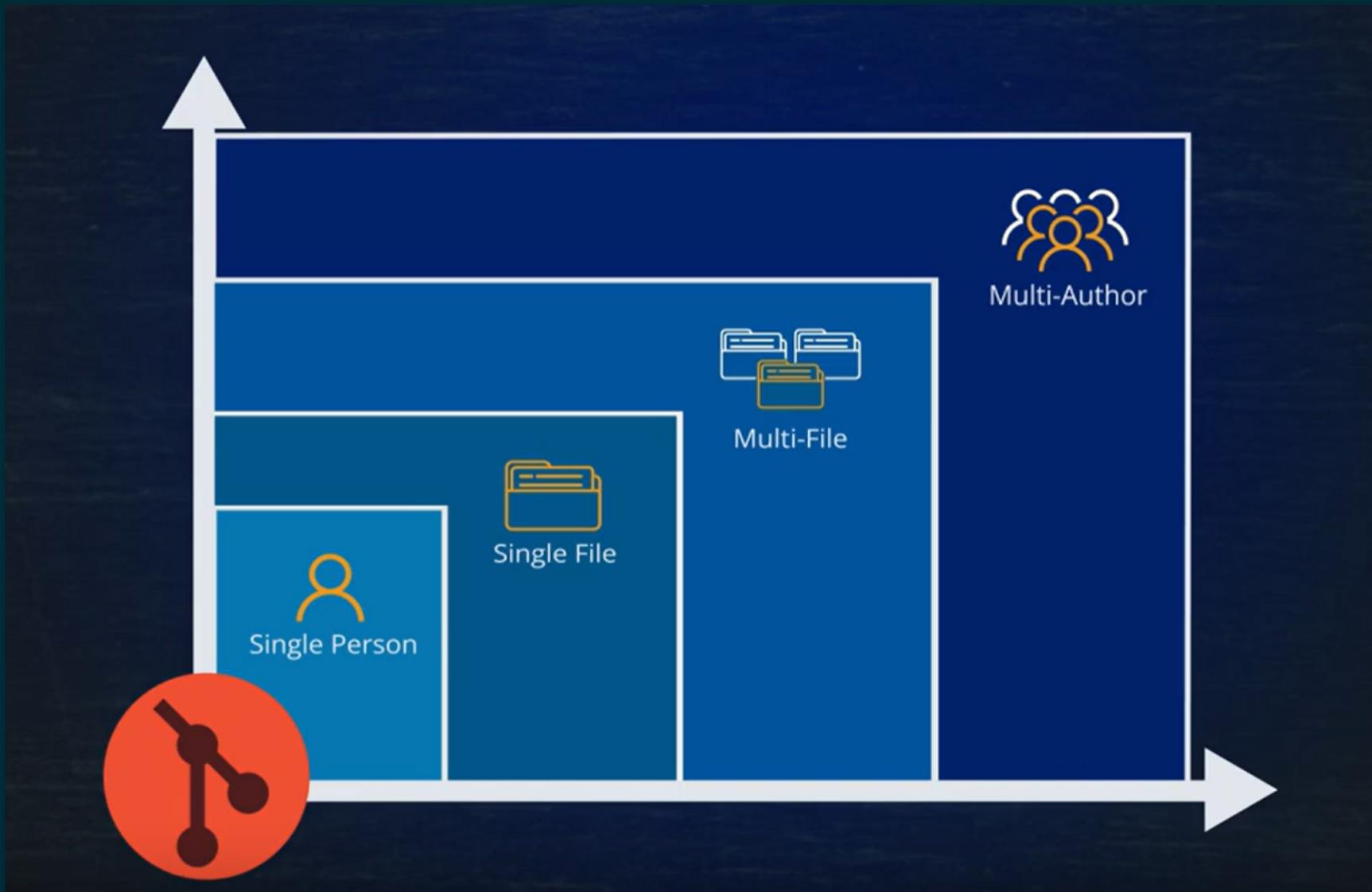
# DEFINITION

- A system to manage projects (repo)
- A system to track how computer files change over time
- A system that supports collaborative revision
- More than file synchronization
- Assists in project back-ups

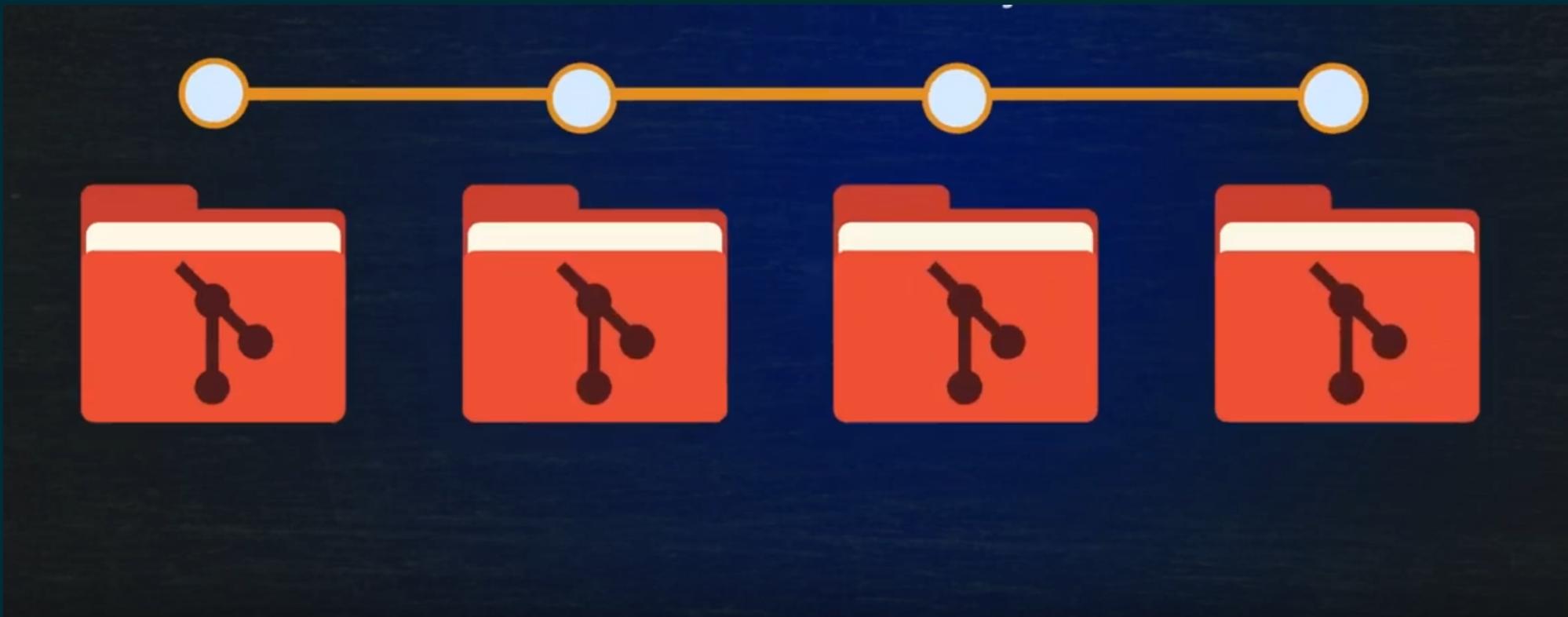
# GIT

- Free open source
- Wildly successful; most broadly implemented
- In use across the globe
- Use it on any file system
- Track any file
- Use it in any environment

# SCALABLE TO PROJECT SIZE



# PROJECT REPOSITORIES



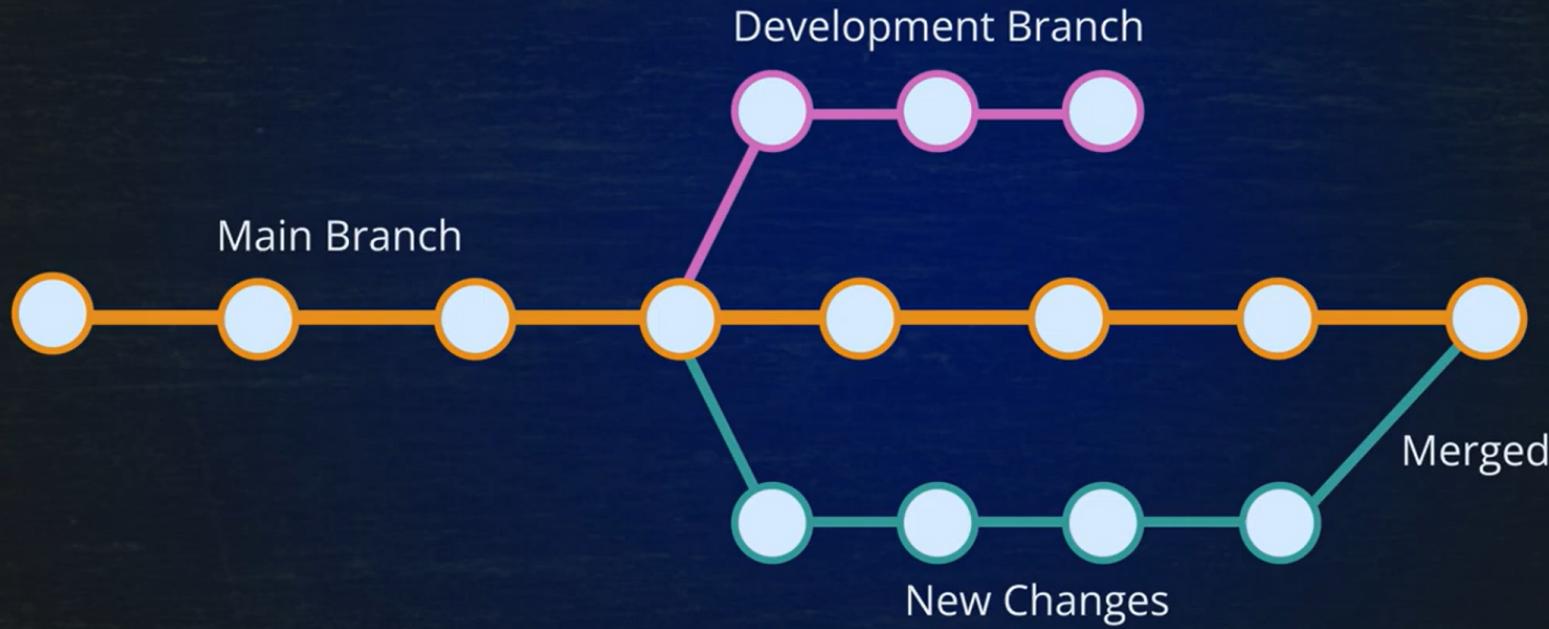
- Work on any file system
- Operates on at the folder level

# TRACK CHANGE



When, Who, Why, What

# BRANCHES



- Main branch
- Experimentation
- Developments
- Merging

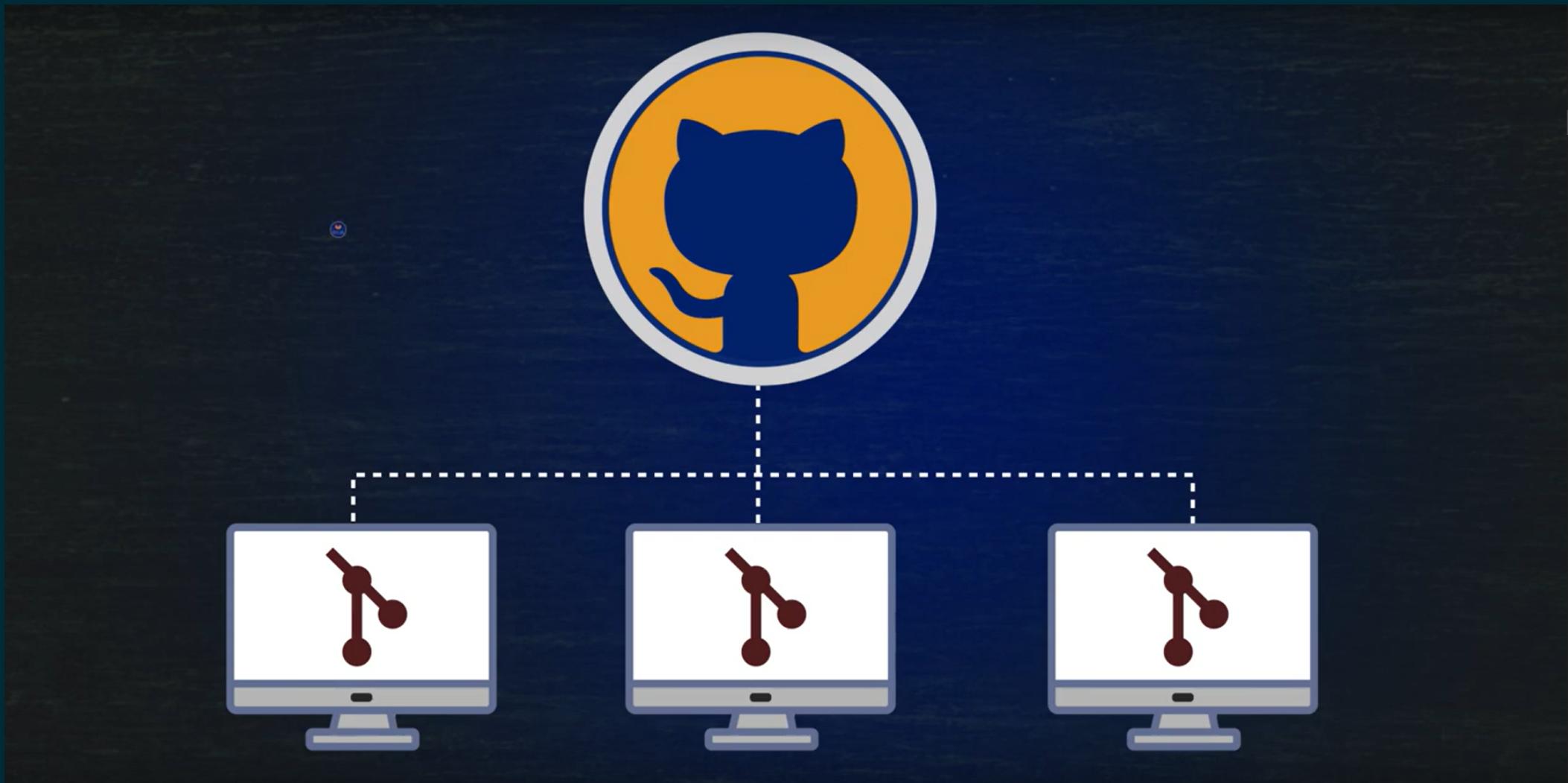
# GITHUB

- Profile (store and host) git repos
- Enable collaboration across the globe or private
- Editorial and fine-grain control



# GitHub

# GIT + GITHUB



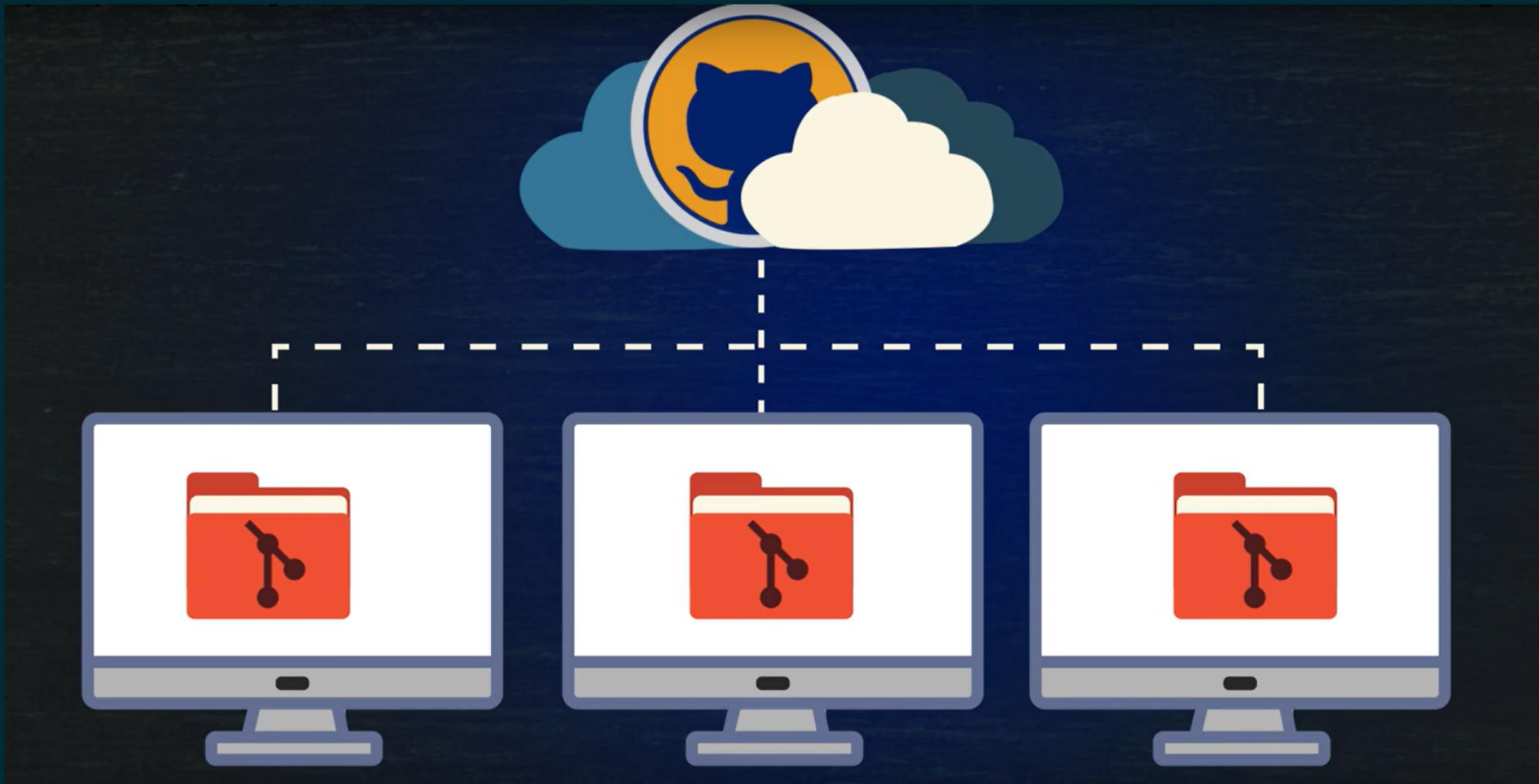
# HUBS

- GitHub
- GitLab
- BitBucket

# DUKE SPECIFIC HUBS

- gitlab.oit.duke.edu (NetID)
- PACE
- Anywhere that data and coding happens.

# FILE DISTRIBUTION AND COLLABORATION



# OTHER PROJECT MANAGEMENT FEATURES



Access  
Control



Task  
Distribution



Bug  
Tracking



Wiki  
Documentation



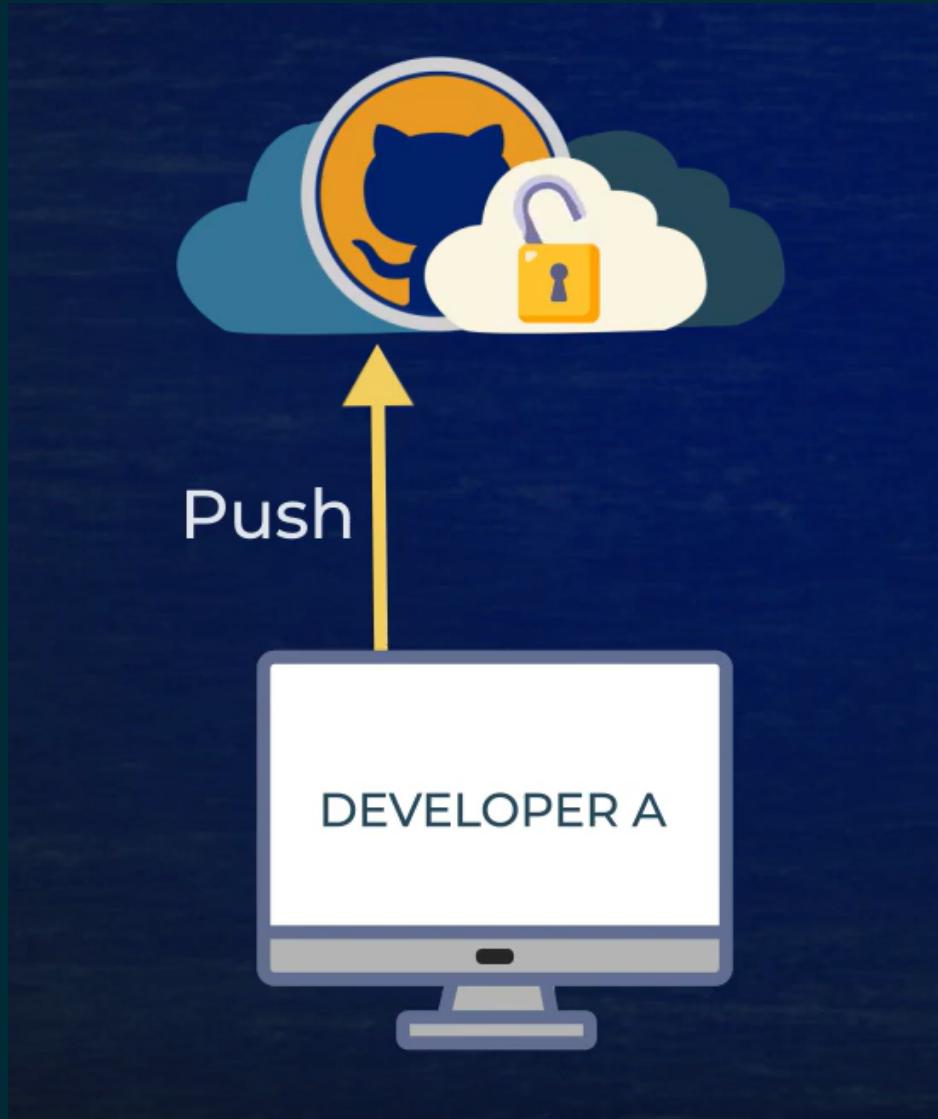
Kanban  
Planning

# BASIC FEATURES

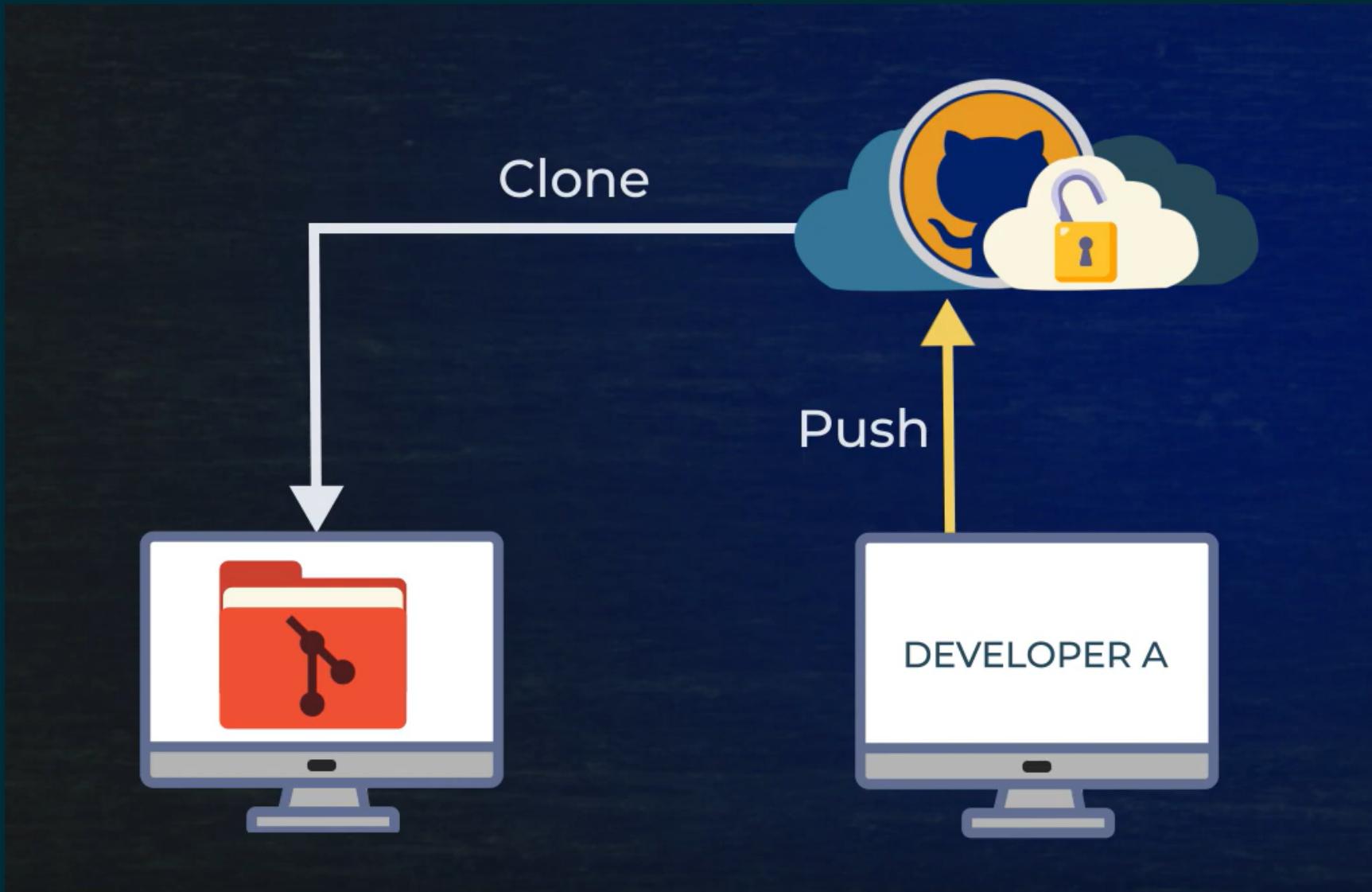
Git features implemented for distribution

- Push
- Public or Private
- Clone / Fork
- Pull Request
- Pull

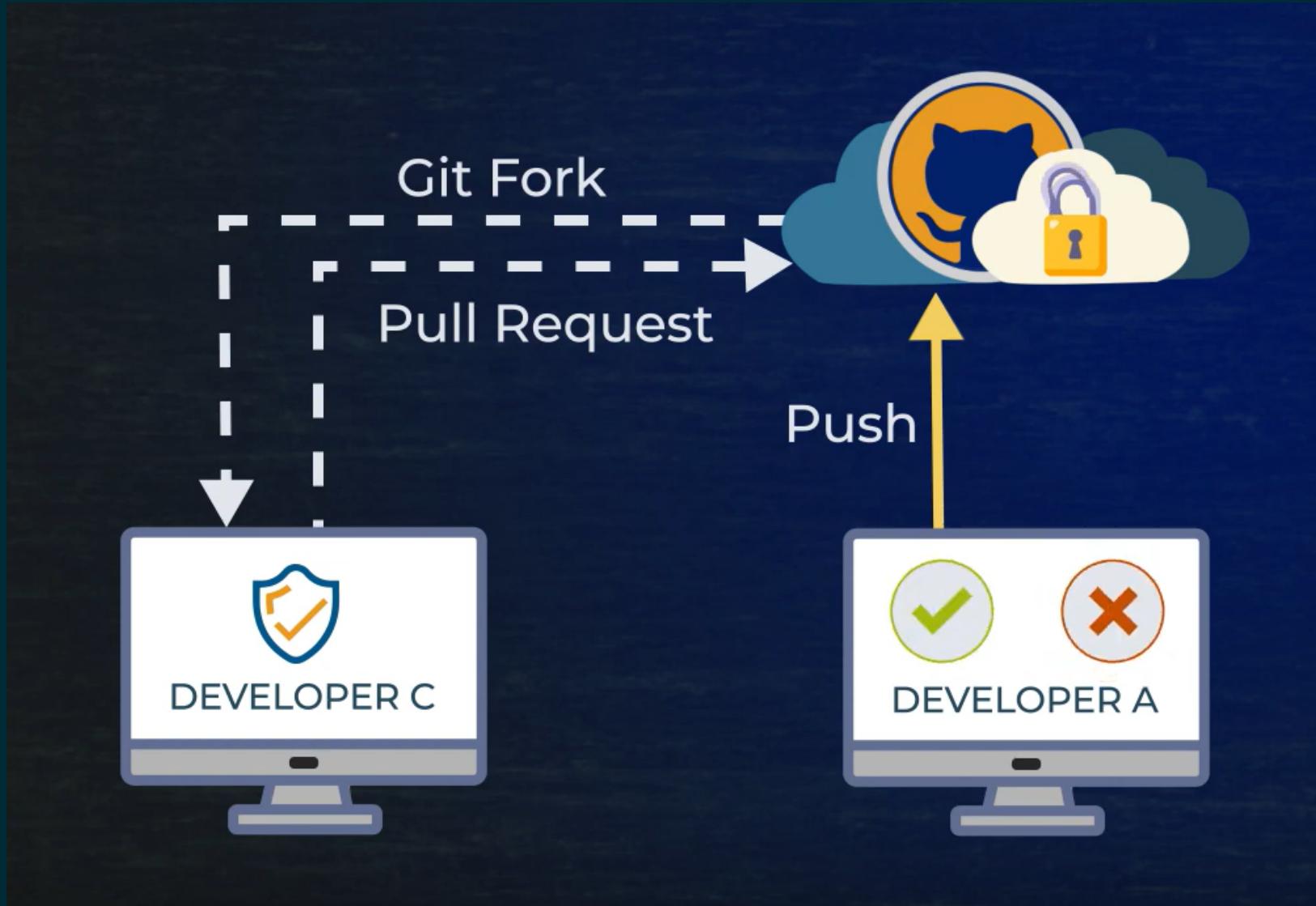
# PUSH



# CLONE



# FORK / PR



# SUMMARY

- Git is used to track changes to your repo
- GitHub is used to distribute your git repo and facilitate collaboration

# NOTEBOOKS



# REPRODUCIBILITY

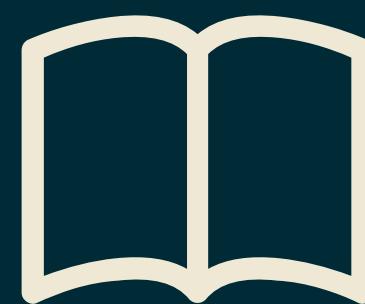
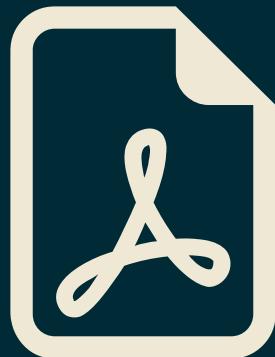
- Do everything with code!
  - Helps reduce repetition errors
  - Helps avoid copy/paste barriers
  - Orchestrate workflows

# COMPUTATIONAL NOTEBOOKS

- Authoring environment
  - Code chunks interspersed with natural language
  - aka *Literate Coding*
- Easy to read and compose
- Graceful degradation

# REPORTS AND EXPRESSIONS

Reports expressions are rendered at code execution



# INTERACTIVITY AND WEB APPLICATIONS

- Shiny
- Flask
- WebR
- Plotly Dash
- ObservableJS

Go to file/function Addins

# QUARTO NOTEBOOK IN RSTUDIO

```
title: "Quarto Computations"
```

This dataset contains a subset of the fuel economy data from the EPA. Specifically, we use the `mpg` dataset from the `ggplot2` package.

```
[r] | label: load-packages
```

```
library(ggplot2)
```

The visualization below shows a positive, strong, and linear relationship between the city and highway mileage of these cars. Additionally, mileage is higher for cars with fewer cylinders.

```
[r] | label: scatterplot
```

```
ggplot(mpg, aes(x = hwy, y = cty, color = cyl)) +  
  geom_point(alpha = 0.5, size = 2) +  
  scale_color_viridis_c() +  
  theme_minimal()
```

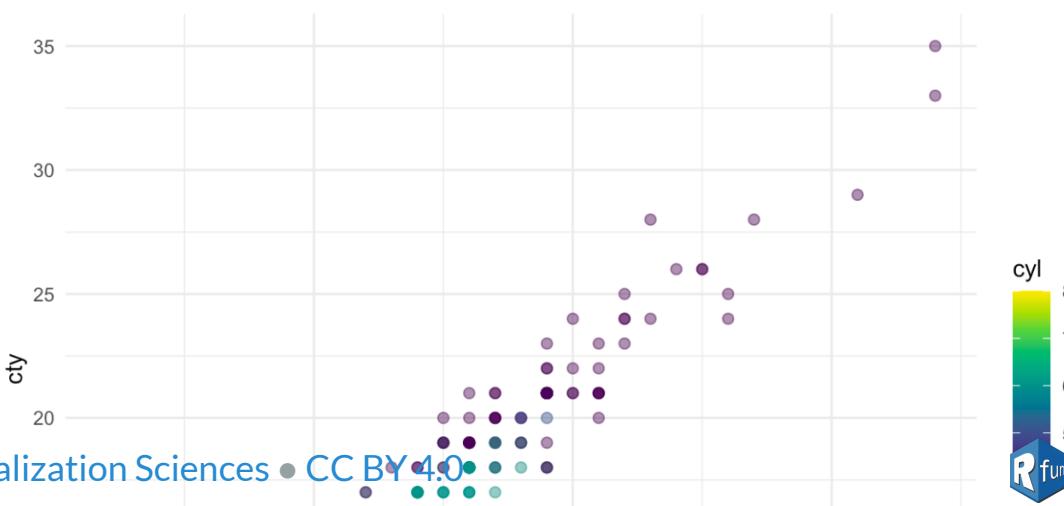
## Quarto Computations

This dataset contains a subset of the fuel economy data from the EPA. Specifically, we use the `mpg` dataset from the `ggplot2` package.

```
library(ggplot2)
```

The visualization below shows a positive, strong, and linear relationship between the city and highway mileage of these cars. Additionally, mileage is higher for cars with fewer cylinders.

```
ggplot(mpg, aes(x = hwy, y = cty, color = cyl)) +  
  geom_point(alpha = 0.5, size = 2) +  
  scale_color_viridis_c() +  
  theme_minimal()
```



# JUPYTER NOTEBOOKS

(yes Classification) is a good starting point for classification tasks, linear regression models are a good starting point for can be fit very quickly, and are very interpretable. You are probably familiar with the simplest form of a linear regression model extended to model more complicated data behavior.

- [Data.ipynb](#)
- [Fasta.ipynb](#)
- [Julia.ipynb](#)
- Linear Regression.ipynb**
- [Lorenz.ipynb](#)
- [lorenz.py](#)
- [R.ipynb](#)
- [untitled.dio](#)
- [untitled1.dio](#)
- [untitled2.dio](#)
- [untitled3.dio](#)
- [untitled4.dio](#)
- [untitled5.dio](#)
- [untitled6.dio](#)

In this section we will start with a quick intuitive walk-through of the mathematics behind this well-known problem, before seeing how before moving on to see how linear models can be generalized to account for more complicated patterns in data.

We begin with:

```
[1]: %matplotlib
import matplotlib.pyplot as plt
not
color="#f37626">e</font>book</h1>
```

**Simple**

We will start by considering where  $a$  is defined.

where  $a$  is defined.

Consider the following code:

```
[2]: rng = np.random.RandomState(42)
x = 10 * rng.rand(100)
y = 2 * x + 3 + rng.randn(100)
plt.scatter(x, y)
```

We can use the following code to fit a linear regression model:

```
[3]: from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(x[:, np.newaxis], y)
y_ = reg.predict(x[:, np.newaxis])
```

Notebook Metadata

```
{
  "kernelspec": {
    "display_name": "Python 3",
    "language": "python",
    "name": "python3"
  },
  "language_info": {
    "codemirror_mode": {
      "name": "ipython",
      "version": 3
    },
    "file_extension": ".py",
    "mimetype": "text/x-python",
    "name": "python",
    "nbconvert_exporter": "python",
    "pygments_lexer": "ipython3",
    "version": "3.6.7"
  },
  "toc-autonumbering": false,
  "toc-showcode": true,
  "toc-showmarkdowntxt": true
}
```

We begin with:

```
[1]: %matplotlib
import matplotlib.pyplot as plt
not
color="#f37626">e</font>book</h1>
```

**Simple**

We will start by considering where  $a$  is defined.

where  $a$  is defined.

Consider the following code:

```
[2]: rng = np.random.RandomState(42)
x = 10 * rng.rand(100)
y = 2 * x + 3 + rng.randn(100)
plt.scatter(x, y)
```

We can use the following code to fit a linear regression model:

```
[3]: from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(x[:, np.newaxis], y)
y_ = reg.predict(x[:, np.newaxis])
```

**Launcher**

**Notebook**

- Python 3
- C++11
- C++14
- C++17
- Julia 1.1.0
- phylogenetics (Python 3.7)
- R

**Console**

- Python 3
- C++11
- C++14
- C++17

**Seattle Weather: 2012-2015**

weather	Number of Records
drizzle	~50
fog	~400
rain	~250
snow	~20
sun	~700

**Lorenz.ipynb**

**Julia**

```
[10]: using RDatasets, Gadfly
plot(dataset("datasets","iris"), x="Sepal.Length", y="Sepal.Width", color="Species")
```

```
[10]: s = eigen(x)
```

```
[10]: Eigen{Complex{Float64},Complex{Float64},Array{Complex{Float64},2},Array{Complex{Float64},1}}
eigenvalues:
10-element Array{Complex{Float64},1}:
 4.793881566545466 + 0.0im
 -0.9445989635995898 + 0.0im
```

**python notebook**

We explore the Lorenz system of differential equations:

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Let's change  $(\sigma, \beta, \rho)$  with ipywidgets and examine the trajectories.

```
[1]: %matplotlib inline
from ipywidgets import interactive, fixed
```

```
[2]: from lorenz import solve_lorenz
w = interactive(solve_lorenz, sigma=(0.0,50.))
w
```

```
interactive(children=(FloatSlider(value=10.0, description='sigma', max=50.0), Flo
atSlider(value=2.666666666666666...))
```

**R**

```
[3]: ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) + geom_point()
```

```
[1]: head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length
5.1	3.5	1.4
4.9	3.0	1.4

John R Little • Center for Data & Visualization Sciences • CC BY 4.0

Mode: Command Ln 1, Col 1 Lorenz.ipynb

Rfun

# QUARTO

- A scientific publishing system
- R, Python, ObservableJS
- Compose with standard text editors, or basic IDEs
  - IDEs: RStudio, Jupyter, VSCode

# RENDERED OUTPUTS

- Artifacts that document a body of work
- Are reproducible and modifiable when data or techniques change
- Easy to update natural language explanations and re-render outputs
- Schedule emails based on report parameters

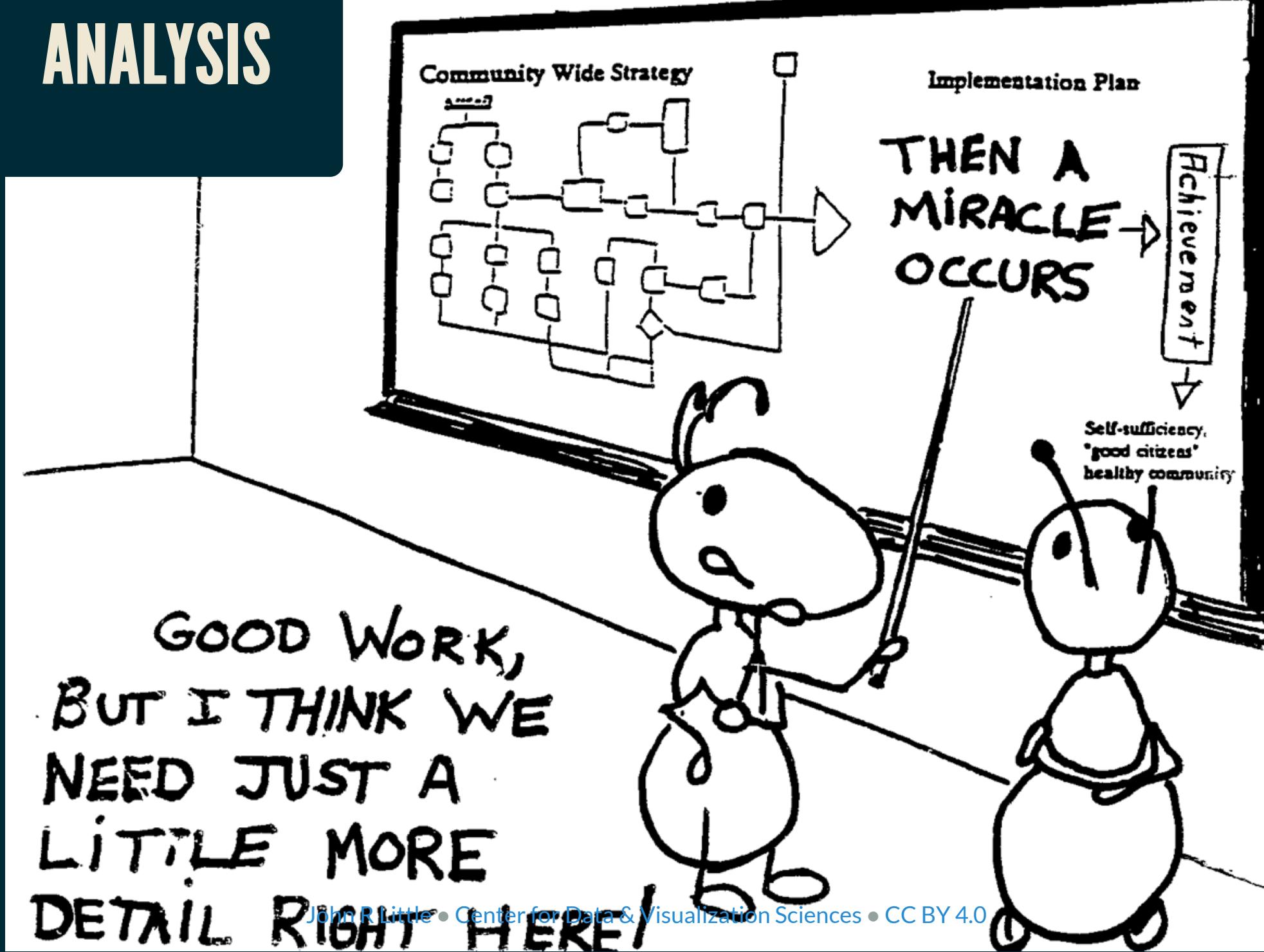
# SUMMARY OF BENEFITS

- Using natural language clearly explain data, models, and workflows
- Reduce dependencies on outside and undocumented steps
- Ability to expose technical code chunks depending on audience focus
- State of the art reproducibility
  - 21st century **container** for evidence-based, computationally-processed research

# ARCHIVE & PUBLISH



# ANALYSIS



# VISUALIZATION

Use graphics tools predicated on the grammar of graphics



# TYPES OF REPOSITORIES

Archival

Posteriority of milestones



Workflow

Versions / evolution of project



}

# HOW

- Generate report expressions from code
- Combine GitHub releases with Zenodo to archive your milestones and share the interactive computation in a binder Hub
- Zenodo: general, open repository to deposit research papers, data sets, code, reports and related artifacts and connect to a citable DOI.
- Binder: package and share reproducible computational environments
  - [mybinder.org](https://mybinder.org) (public BinderHub portal)



# SHARING YOUR WORKSPACE

Your computation workspace (i.e. your laptop, desktop, cloud)

Give someone else your laptop so they can play around with your projects

- the code, the data, the settings and configurations?
- Good idea?

Now you can share a copy of your computational environment



# BINDER HUB

- Easiest: mybinder.org open and public
  - `quarto use binder`
- Security demands may push you to use singularity

# STEPS

1. Make a GitHub Release at project milestone(s)
2. Connect GitHub to Zenodo
  1. Mint a DOI to a GitHub Release (persistent identifier: citation; milestones)
  2. With DOI, link to ORCID
3. Create a publicly launchable, fully functional computation container of your work

# CONTAINER EXAMPLES

- [https://github.com/libjohn/workshop\\_rfun\\_iterate?  
tab=readme-ov-file#readme](https://github.com/libjohn/workshop_rfun_iterate?tab=readme-ov-file#readme)
- [https://github.com/libjohn/workshop\\_webscraping?  
tab=readme-ov-file#readme](https://github.com/libjohn/workshop_webscraping?tab=readme-ov-file#readme)

# END TO END

Three minutes on what it looks like

## 1. Project with version control

- project folder with TIER organization
- data (raw)
- version control (git)

## 2. Coding notebook

- data cleaning
- natural language explanations
- analysis and modeling
- visualization
- generate report expressions from code

## 3. Publish

- workflow archived as
- Milestones linked to
- **Informal:** web, file share  
▪ Whitepapers, slides
- **Formal:** vis-a-vis peer review