

## Unlocking Efficiency Through Digitisation:

Plotting the Journey with Atlas for Clinical Trial Output Generation



# ATLAS

### Presenters :

**Eddy Foster**, Senior Director, Data Science Acceleration, Roche

**Daphné Grasselly**, Senior Data Scientist, Roche

# **Context**

## Digitisation and Digital Workflows

# The Ocean Platform

Designed for both GxP and Non GxP work, “FAIR by Design” and language agnostic

## NextGen Tools/Toolkits for R workflow

Additional tools will follow over time (ATLAS, TIDES)

## Legacy Tools / Business Tools & Workflows

## OCEAN

The backbone system that hosts the NextGen Tools under one roof for both regulatory and exploratory work.

Access to Rstudio/SAS Studio/Jupyter containers via OCEAN Workbench

Code Management: GitLab



# Digitisation and Digital Workflows

What do we mean?

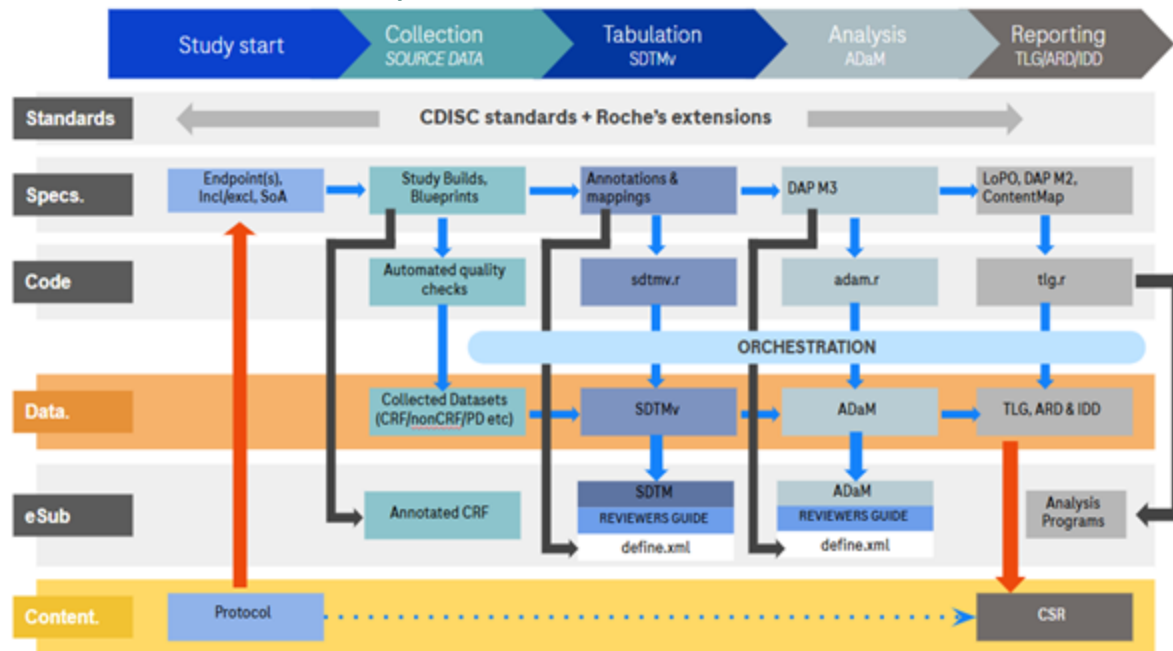
Product creation (e.g. **SDTMv, ADaM, TLG**) is supported by a workflow that has transitioned from a document-centric approach to one driven by digital apps that:

- Focus on Data Scientists needs with **usability at the center**
- Use rich, interconnected metadata that forms a portfolio-wide view
  - A Contextual Information **Backbone**
- Create **FAIR by design** specifications of product definitions and transformations
- Support efficient **use of standards and non-standard** definitions at all levels
- Integrates with programmer tooling (packages etc) to drive consistency and enable automation
  - For AI agents as well as HITL\*

# Clinical Trial Data and Analysis Flow

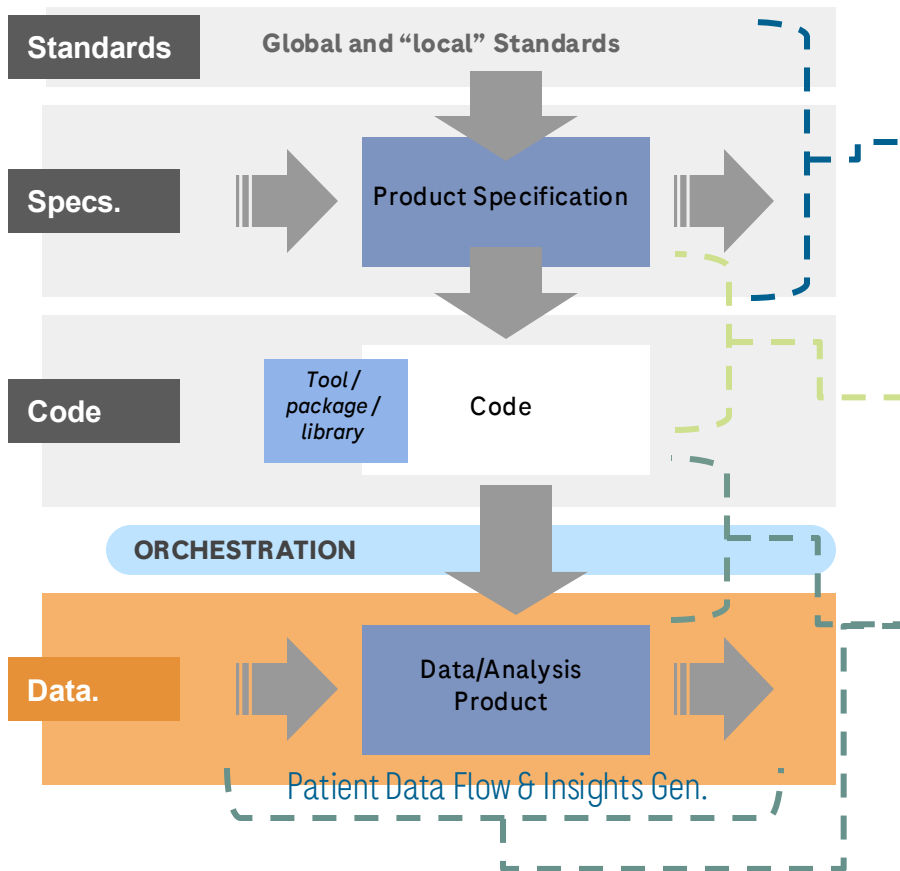
Generating Insights and Evidence about our portfolio

The general approach to **insights generation** is fairly consistent across the industry



- There is a *dual flow*; not only of the data but of the *contextual information* related to the data and analysis products
- By digitising this contextual information we can harness it to drive meaningful productivity gains and operational insights into the work across our portfolio

# Streamlined Digital Workflows



## Consistent and clear Product Definition

- Describes the intended state of the product
- Transitions between products
- Promotes standardisation and reuse
- Detailed portfolio-level view of planned work

## Faster Code Generation

- Automation of code generation and QC
- Deterministic (rules-based) or probabilistic (AI)

## Simplified Orchestration

- Easier to set up consistent and efficient pipelines with fewer execution issues at critical times
- Provides key information about actual delivery

Streamlined workflow for AI agents and HITL

# Introducing ATLAS

Web Application for Clinical Trial Output Planning & E2E Orchestration



## ATLAS



Enables the user to efficiently define the list of planned outputs for their analysis (SDTM, ADaMs, TLG)



Creates an Orchestration Plan that encapsulates the full analysis pipeline

*For the first time we'll have a detailed global picture of the work we are planning and executing across the portfolio*

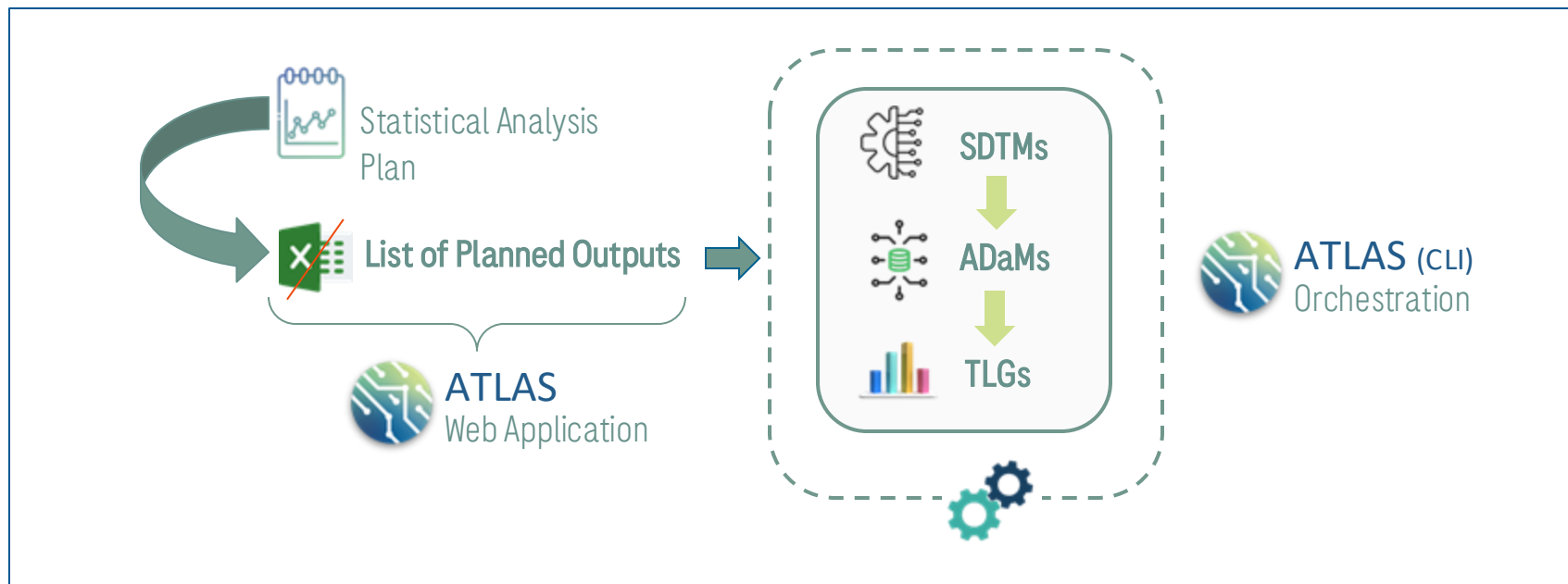
# ATLAS

## Deep Dive

# ATLAS components

Clinical trial programming plan and E2E Orchestration

From clinical trial plan to E2E outputs generation :



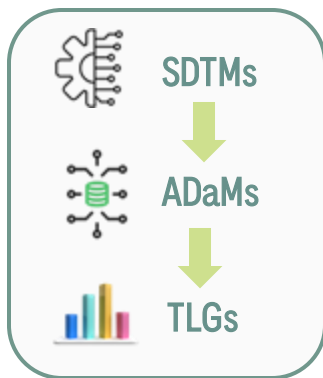


# ATLAS - Web Application (Metadata Definition)

Clinical Trial Plan

## 1) Users define clinical trial plan metadata

Clinical Trial Outputs and related programs



TLG filters (R/SAS expressions)  
TLG titles and footnotes

## 2) Activity configuration (Reporting Event workspace)



Gitlab Repository Creation (from specific templates)  
Data Area creation

## 3) Outputs assignments to the Activity

Selection of a subset of  
outputs to include in the  
report.

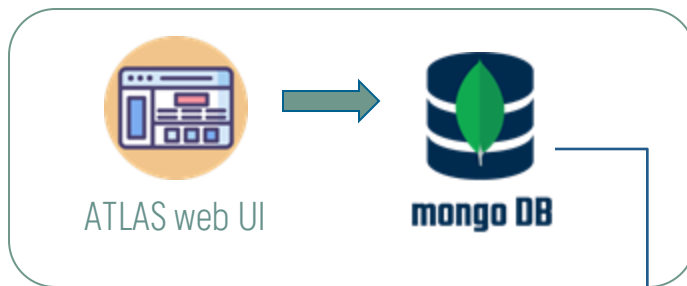




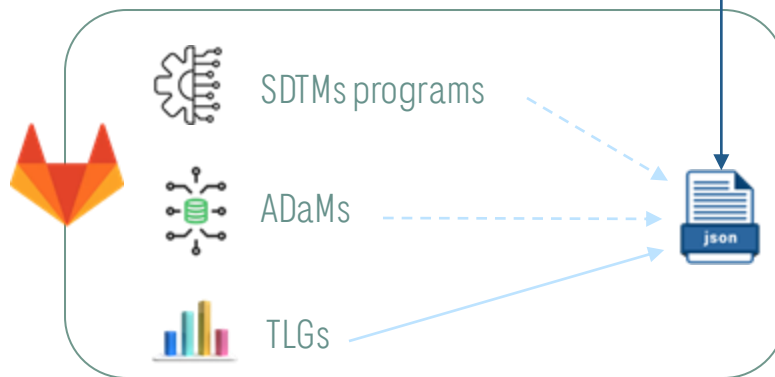
# ATLAS - Command Line Interface

Metadata extraction

## Metadata Definition



## Metadata Extraction



TLG code templates generation  
Access TLG metadata within the programs



# ATLAS - Command Line Interface

**Snakemake** : Reproducible and scalable data analyses

## What is Snakemake?

A workflow management system for **reproducible and scalable data analysis**.  
Combines ease of use with powerful features for complex workflows.

### Key Features:



- **Rules:** Define workflows as rules specifying **input**, **output**, and **shell** cmd.
- **Dependency Management:** Resolves task dependencies.
- **Parallel Execution:** Uses available resources (CPUs, GPUs, etc.).
- **Reproducibility:** Tracks all steps, ensuring workflows can be reproduced easily.
- **Containers:** Run jobs in **SAS/R Containers**.



# ATLAS - Command Line Interface

Snakemake: Reproducible and scalable data analyses

## What is Snakemake?



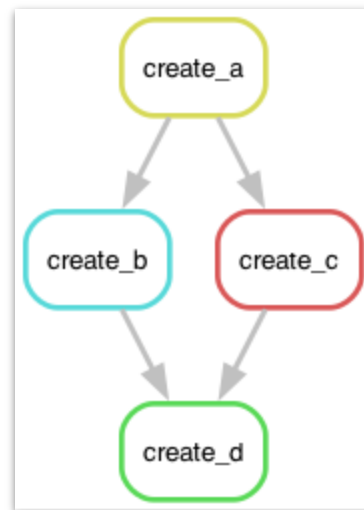
Snakefile

```
rule create_a:  
    output: "a.txt"  
    shell: "touch a.txt"  
  
rule create_b:  
    input: "a.txt"  
    output: "b.txt"  
    shell: "cat a.txt > b.txt"  
  
rule create_c:  
    input: "a.txt"  
    output: "c.txt"  
    shell: "cat a.txt > c.txt"  
  
rule create_d:  
    input: ["b.txt", "c.txt"]  
    output: "d.txt"  
    shell: "touch d.txt"
```



## DAG

(Directed Acyclic Graph)





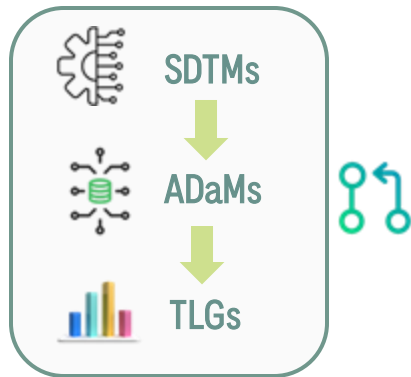
# ATLAS - Command Line Interface

## Metadata extraction and Snakefiles generation

Git Repository

### 1) Orchestration Dependencies

Define SDTM/ADaMs and TLGs inter-dependencies



### 2) Extract Metadata and Snakefiles



Extract metadata as Json files  
TLG standards **code templates** generation

### 3) Run your Analysis



Run **Snakemake command** (Specific rule).  
Orchestration is run within **CI pipelines**



# ATLAS - Command Line Interface

## Demo

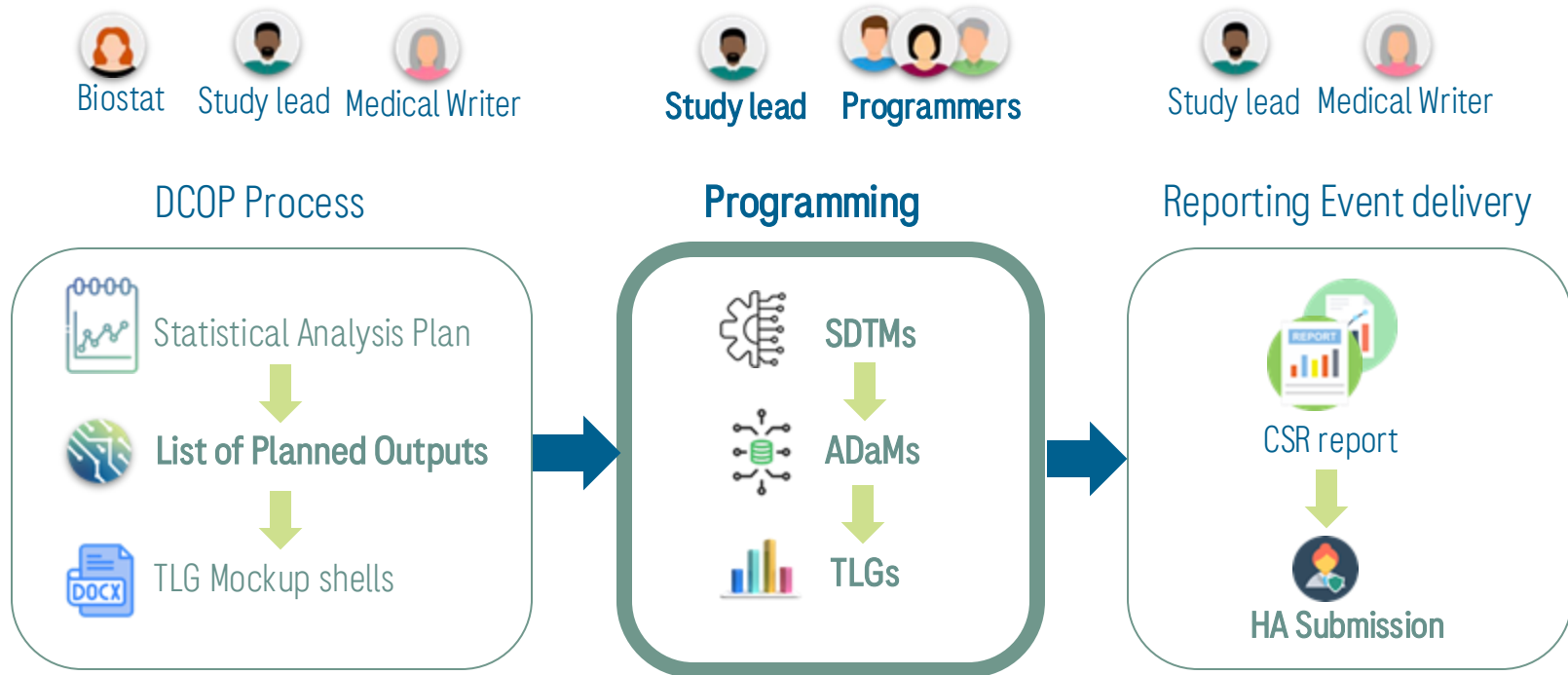


# Roadmap

## What's Next ?

# E2E Content generation

From SAP to Report generation



# Open Source TLG Catalog

## Catalog of Standard mockup shells

(Standards TLG)

### AET01

GSDR
ADVERSE EVENTS
TABLE

Overview of Deaths and Adverse Events Summary Table

Table
Programming Notes
Code

```
# Load Libraries -----
library(NEST.TLGs)
library(tidyverse)

# Load data frames from {syntheticadam} into named list
adam_db <- syntheticadam::get_data(c("adsl", "adae"))
# set paths for sample {citril} metadata
tlg.templates::set_citril_options()

# Get command line arguments -----
filter_abbrev <- get_cmdarg("filter", suggest_filter("t_ae_oview"))
tlg_name <- parse_tlg_name("t_ae_oview", filter_abbrev)
output <- get_cmdarg("output", "t_ae_oview_GEN.docx")

# Load metadata -----
reporting_info <- read_reporting_info(report_event = get_reporting_event())
lopo <- read_lopo(entry = tlg_name)
read_whiskers()

# Filter and reformat data -----
adam_db <- adam_db |>
  filter_with_suffix(suffix = filter_abbrev, suffix_list = read_filters()) |>
  dunlin::reformat(
    format = extract_rules(format_list = read_formats()[[tlg_name]],
                          rules_list = read_rules(),
                          default = read_formats()[["all"]])
  )
```

### AET01

GSDR
ADVERSE EVENTS
TABLE

Overview of Deaths and Adverse Events Summary Table

Table
Programming Notes
Code

≡ t\_ae\_oview\_GEN...
1 / 1
49%
+
🔍
🔄
📄
📄
📄

1

# Roadmap

## Next features



**Content Reuse** : Defining Metadata and Mockups shells libraries



Introducing Linkage with **R coding agents** for **non standards** ADaMs derivations and TLG



**Atlas content generation from SAP**



Enhanced **users experience** (web UI improvements)



# Thanks to our key contributors !



## Core team :

Andrii Parfenenko, Bartosz Kaszewski, Dinakar Kulkarni, Franciszek Walkowiak, Haoyang Ju, Kamil Bojarski, Mathieu Cayssol, Remigiusz Pulit, Rohan Parmar, Wojciech Alaszewski



## Acknowledgment :

Ali Mahdipour, Cheryl Jones, Christopher DiPietrantonio, Hannah Eddleston, Holger Langkabel, Ian Healy, Joel Laxamana, Kathrin Flunkert, Konstantina Koukourikou, Leena Khatri, Magdalena Krochmal, Nelia Lasierra, Nick Ramirez, Ran Ma, Reza Rezai, Ross Farrugia, Tatiana Alonso Amor, Tim Barnett, Valeria Visona, Wenyi Liu