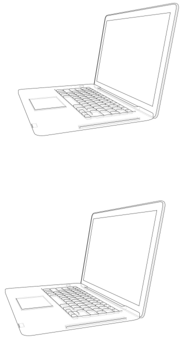
Moving towards Continuous Benchmarking (CB)

9th JLESC Workshop
April 15th-17th, 2019 | Knoxville, TN

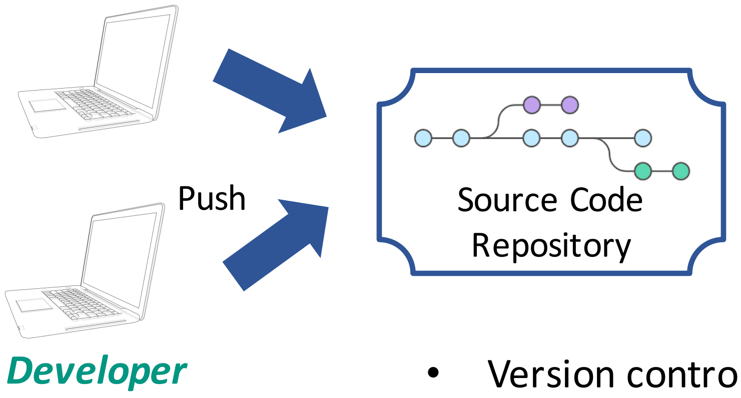
Anzt, Chen, Cojean, Dongarra, Flegar, Nayak, Quintana-Orti, Tsai, Wang

A Healthy Software Development Cycle



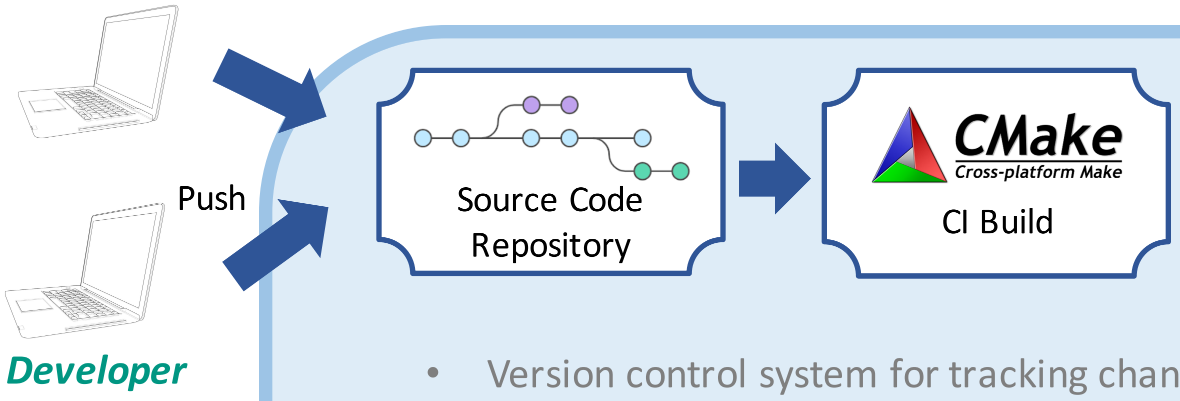
Developer

A Healthy Software Development Cycle



- Version control system for tracking changes and coordinating collaborative development.

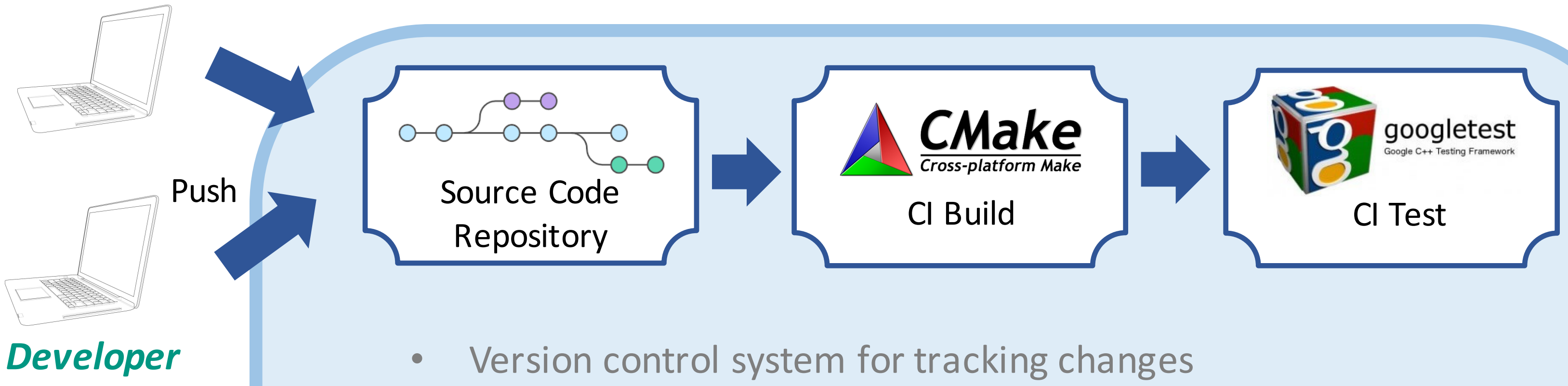
A Healthy Software Development Cycle



- Version control system for tracking changes and coordinating collaborative development.
- CI Build system continuously checks the compilation success on different architectures.

Continuous Integration (CI)

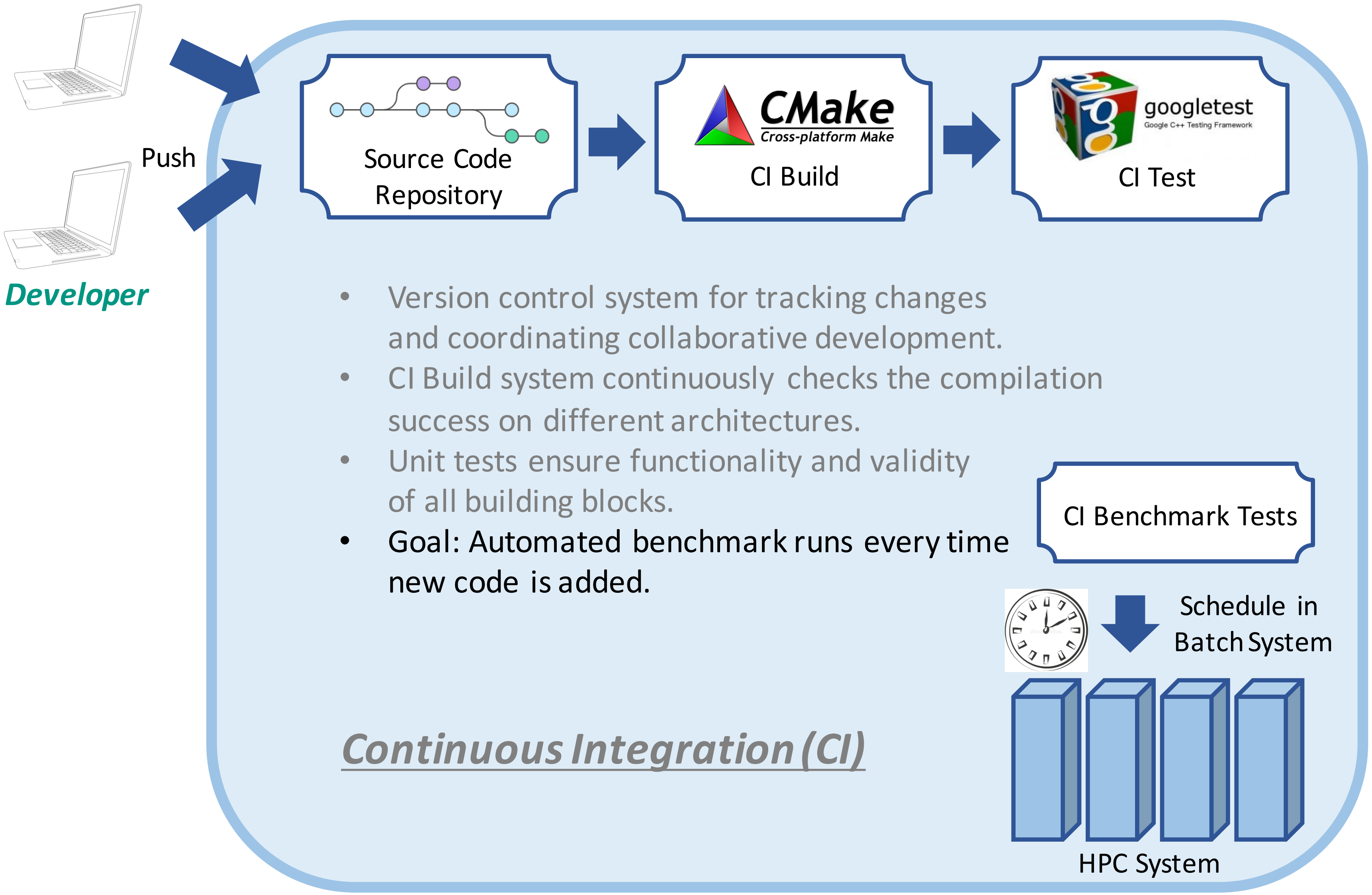
A Healthy Software Development Cycle



- Version control system for tracking changes and coordinating collaborative development.
- CI Build system continuously checks the compilation success on different architectures.
- Unit tests ensure functionality and validity of all building blocks.

Continuous Integration (CI)

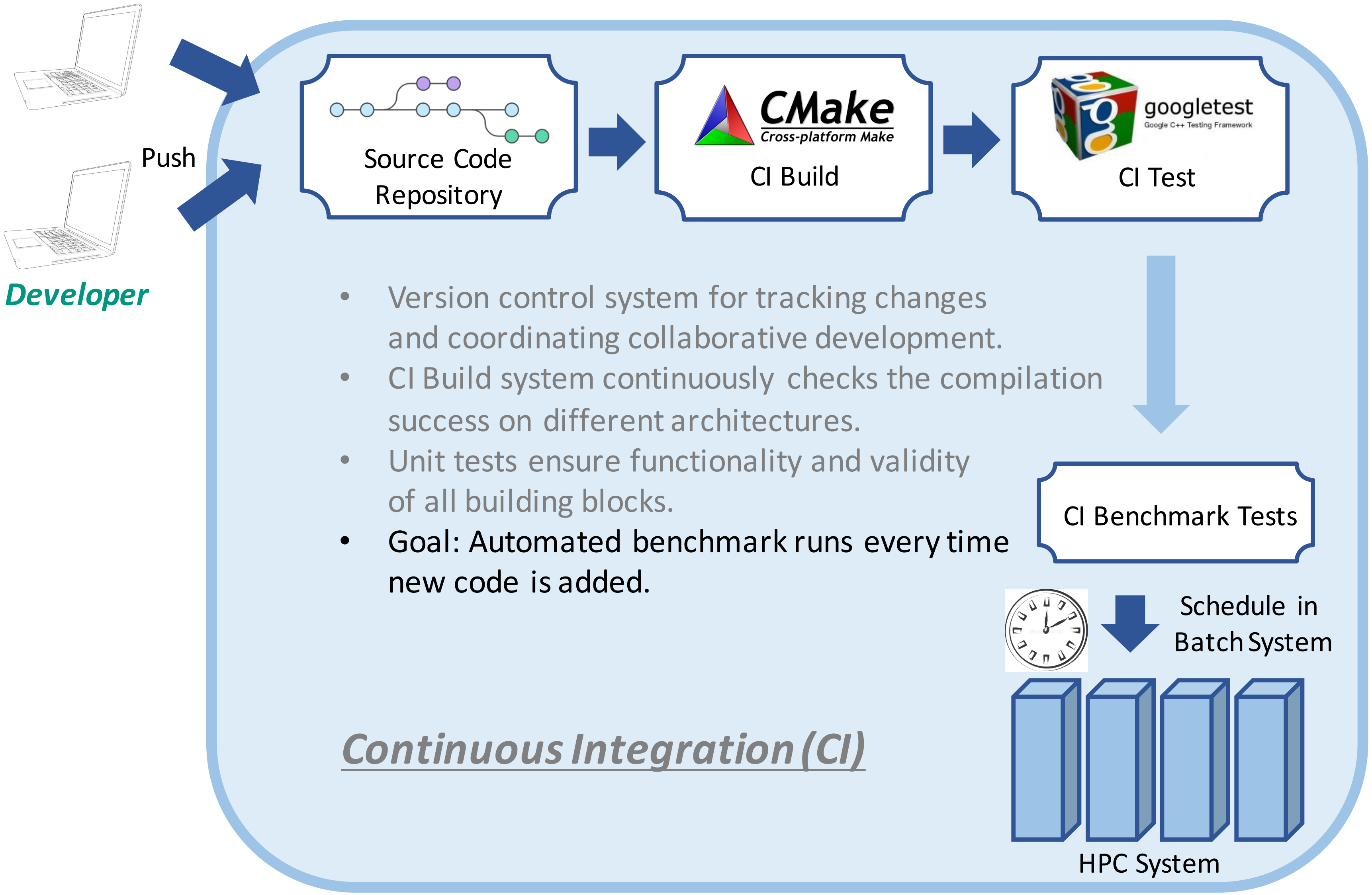
A Healthy Software Development Cycle



- Version control system for tracking changes and coordinating collaborative development.
- CI Build system continuously checks the compilation success on different architectures.
- Unit tests ensure functionality and validity of all building blocks.
- Goal: Automated benchmark runs every time new code is added.

Continuous Integration (CI)

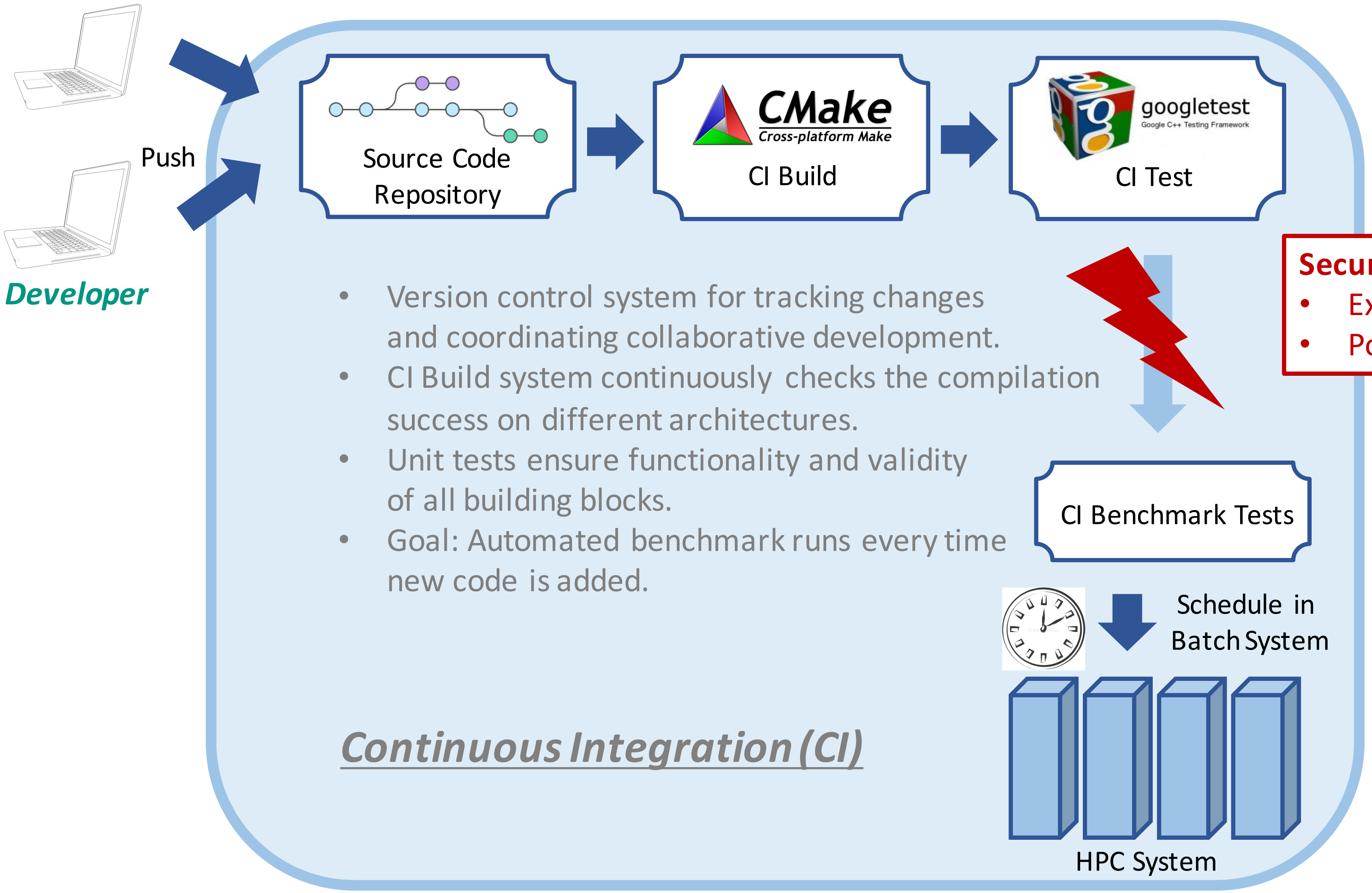
A Healthy Software Development Cycle



- Version control system for tracking changes and coordinating collaborative development.
- CI Build system continuously checks the compilation success on different architectures.
- Unit tests ensure functionality and validity of all building blocks.
- Goal: Automated benchmark runs every time new code is added.

Continuous Integration (CI)

A Healthy Software Development Cycle



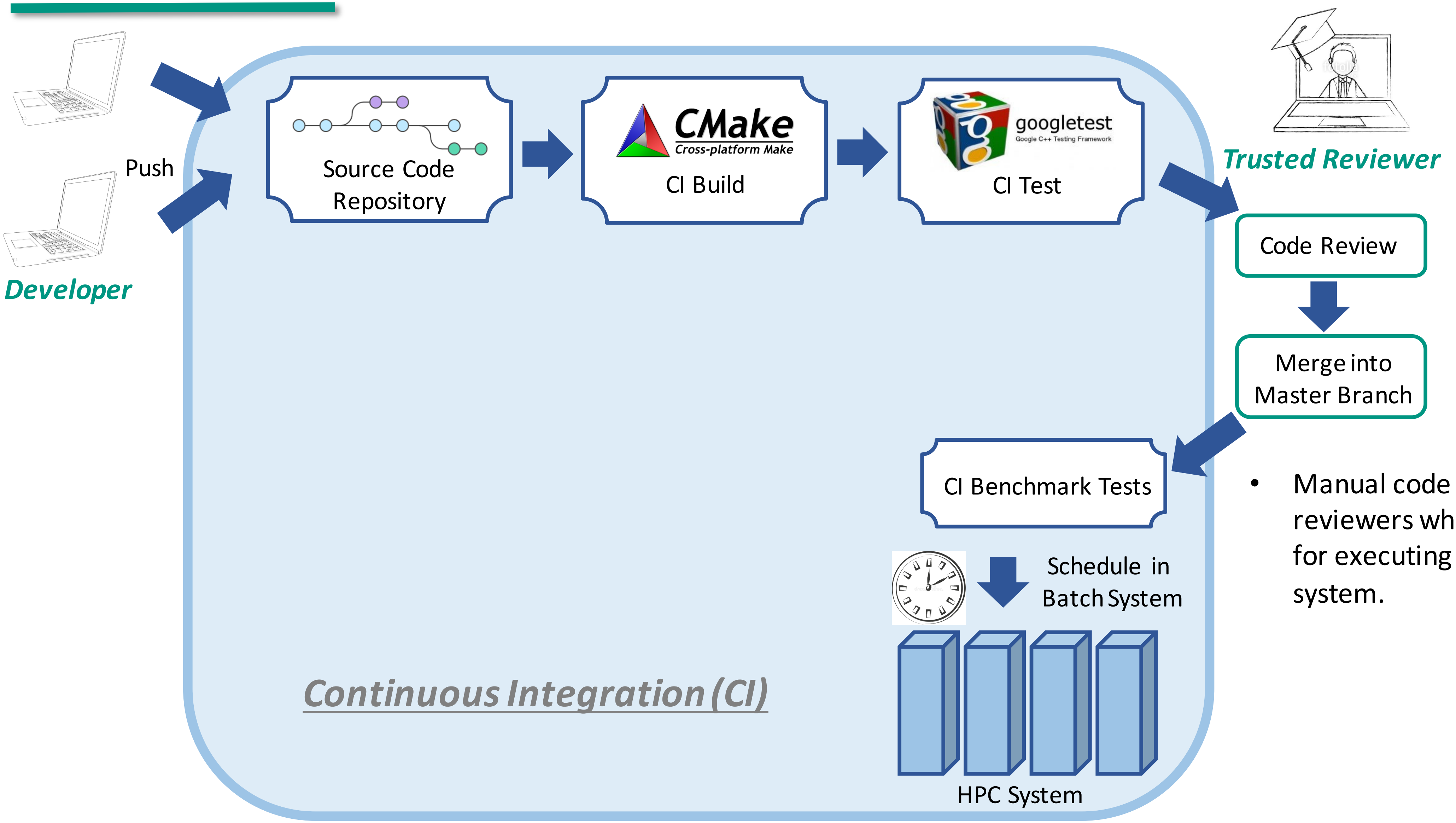
- Version control system for tracking changes and coordinating collaborative development.
- CI Build system continuously checks the compilation success on different architectures.
- Unit tests ensure functionality and validity of all building blocks.
- Goal: Automated benchmark runs every time new code is added.

Continuous Integration (CI)

Security Thrust!

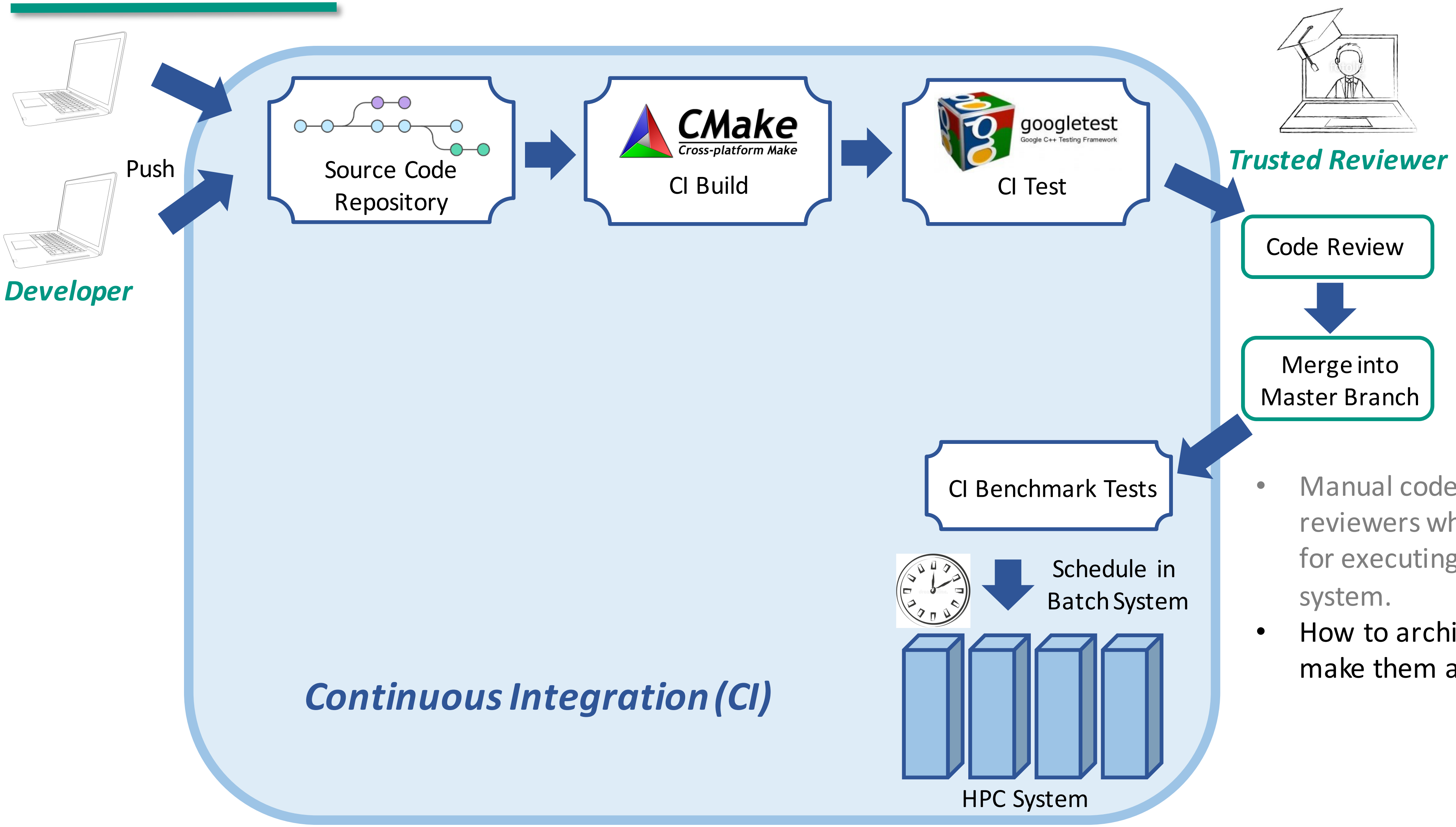
- External code is run on HPC system
- Possible point of attack

A Healthy Software Development Cycle



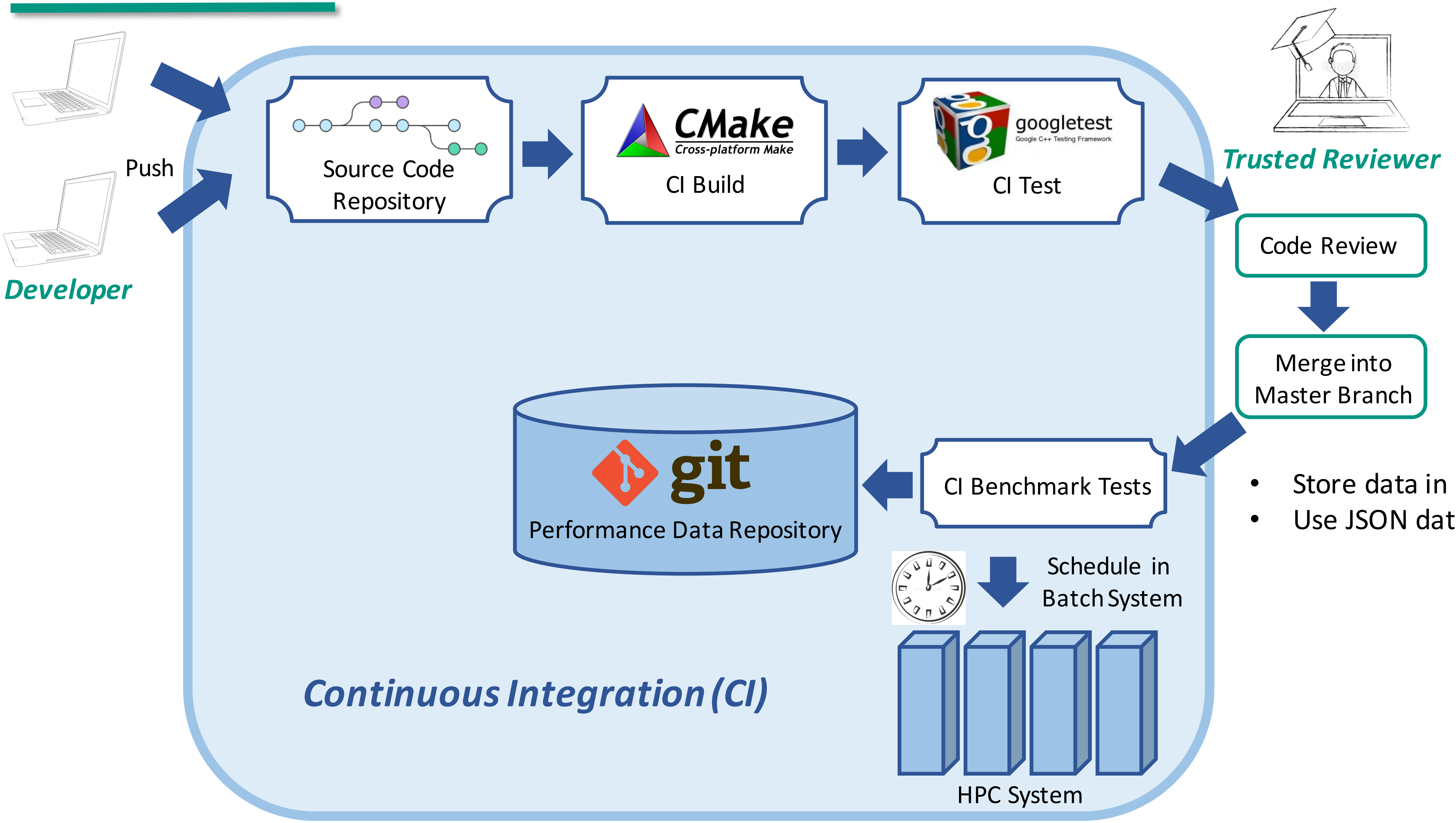
- Manual code reviews by trusted reviewers who take responsibility for executing code on the HPC system.

A Healthy Software Development Cycle



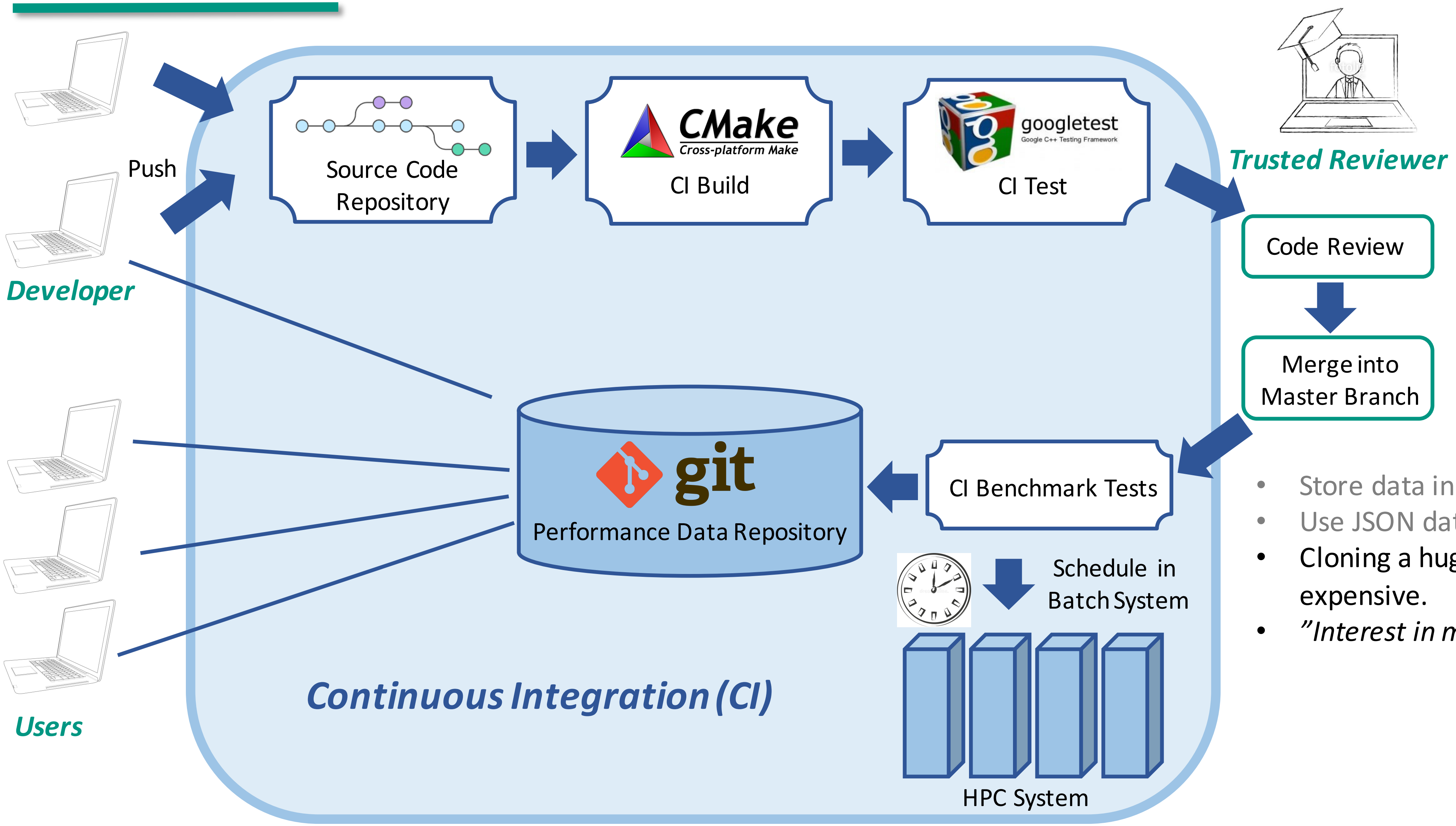
- Manual code reviews by trusted reviewers who take responsibility for executing code on the HPC system.
- How to archive the results and make them accessible?

A Healthy Software Development Cycle



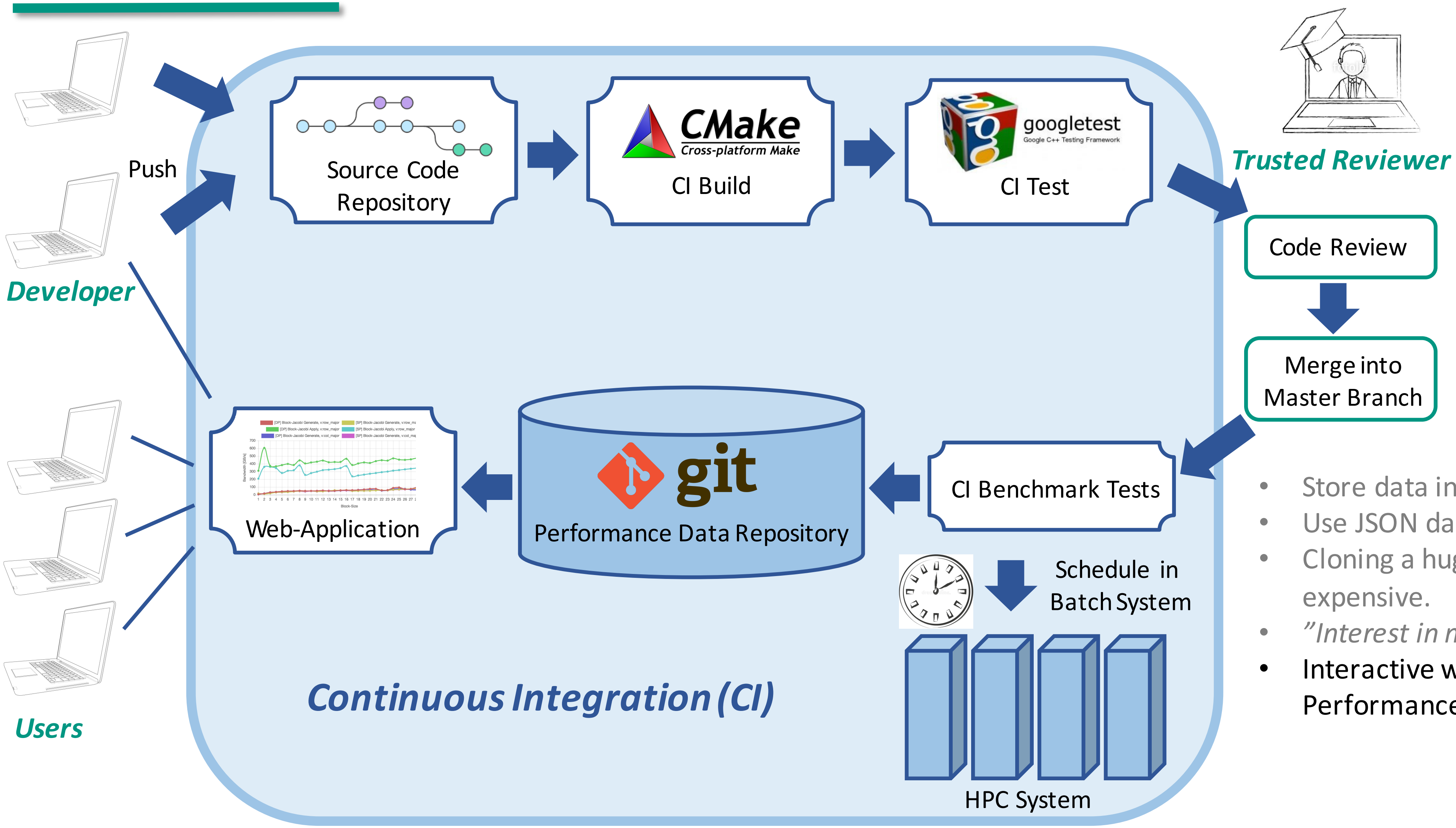
- Store data in separate repository.
- Use JSON data exchange format.

A Healthy Software Development Cycle



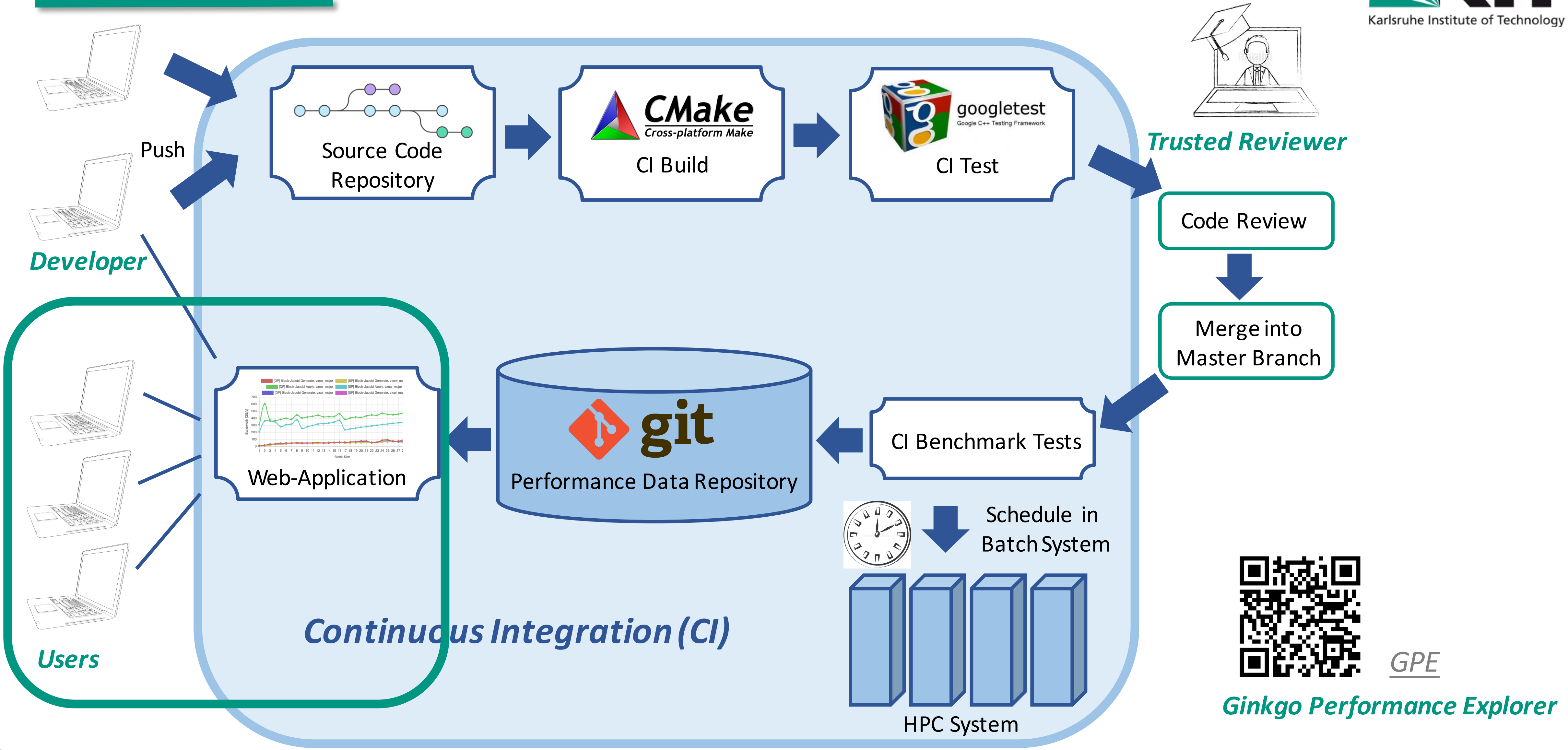
- Store data in separate repository.
- Use JSON data exchange format.
- Cloning a huge repository is expensive.
- *"Interest in my experiments, only."*

A Healthy Software Development Cycle



- Store data in separate repository.
- Use JSON data exchange format.
- Cloning a huge repository is expensive.
- *"Interest in my experiments, only."*
- Interactive web-based Ginkgo Performance Explorer (GPE).

A Healthy Software Development Cycle



Developer

Push

Source Code Repository

CI Build

CI Test

Trusted Reviewer

Code Review

Merge into Master Branch

CI Benchmark Tests

Schedule in Batch System

HPC System

Performance Data Repository

Web-Application

Users

Continuous Integration (CI)

GPE

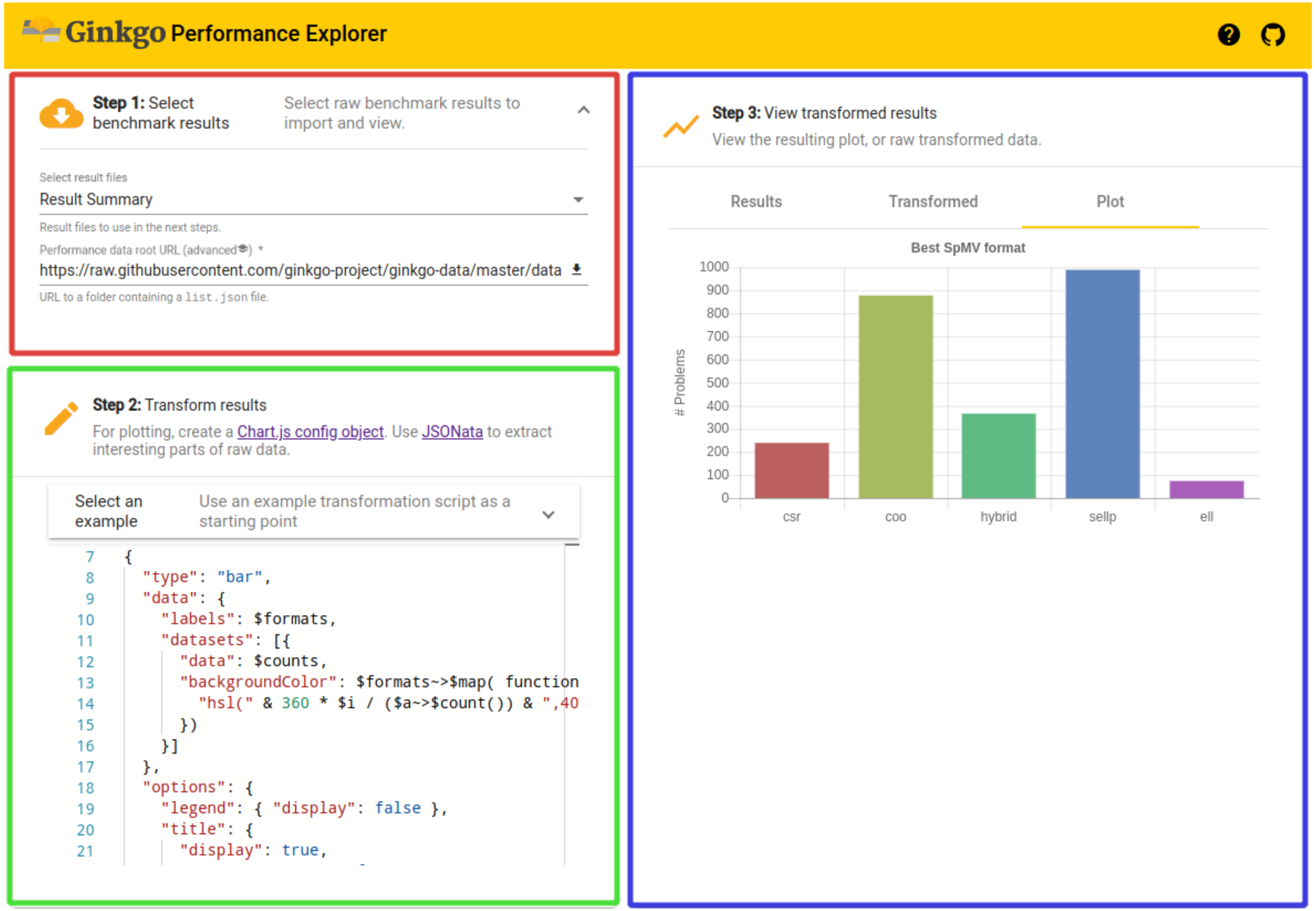
Ginkgo Performance Explorer

Ginkgo Performance Explorer (GPE)

Data Selection Tab

Transformation Script Editor

Data and Plot Viewer



Ginkgo Performance Explorer

Step 1: Select benchmark results

Select raw benchmark results to import and view.

Select result files

Result Summary

Result files to use in the next steps.

Performance data root URL (advanced) *

https://raw.githubusercontent.com/ginkgo-project/ginkgo-data/master/data

URL to a folder containing a `list.json` file.

Step 2: Transform results

For plotting, create a Chart.js config object. Use JSONata to extract interesting parts of raw data.

Select an example — Use an example transformation script as a starting point

```
{
  "type": "bar",
  "data": {
    "labels": $formats,
    "datasets": [{
      "data": $counts,
      "backgroundColor": $formats~>$map( function
        "hsl(" & 360 * $i / ($a~>$count()) & ",40
      })
    }]
  },
  "options": {
    "legend": { "display": false },
    "title": {
      "display": true,
```

Step 3: View transformed results

View the resulting plot, or raw transformed data.

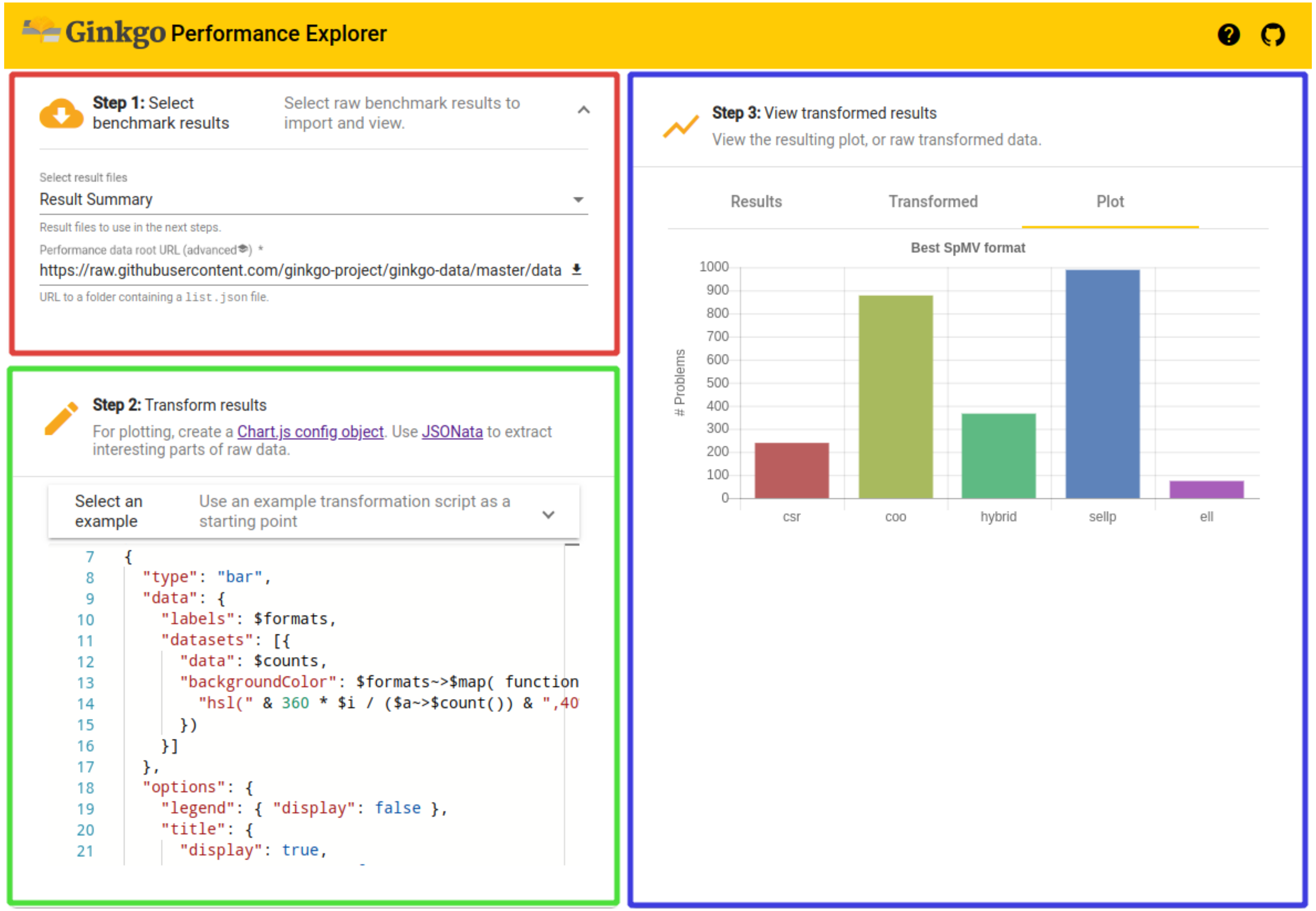
GPE

Ginkgo Performance Explorer (GPE)

Data Selection Tab

1.
Select Data in Git repository.

Transformation Script Editor



Ginkgo Performance Explorer

Step 1: Select benchmark results

Select raw benchmark results to import and view.

Select result files
Result Summary
Result files to use in the next steps.

Performance data root URL (advanced) *
https://raw.githubusercontent.com/ginkgo-project/ginkgo-data/master/data
URL to a folder containing a `list.json` file.

Step 2: Transform results

For plotting, create a Chart.js config object. Use JSONata to extract interesting parts of raw data.

Select an example — Use an example transformation script as a starting point

```
{
  "type": "bar",
  "data": {
    "labels": $formats,
    "datasets": [{
      "data": $counts,
      "backgroundColor": $formats~>$map( function
        "hsl(" & 360 * $i / ($a~>$count()) & ",40
      })
    }]
  },
  "options": {
    "legend": { "display": false },
    "title": {
      "display": true,
```

Step 3: View transformed results

View the resulting plot, or raw transformed data.

Data and Plot Viewer

GPE

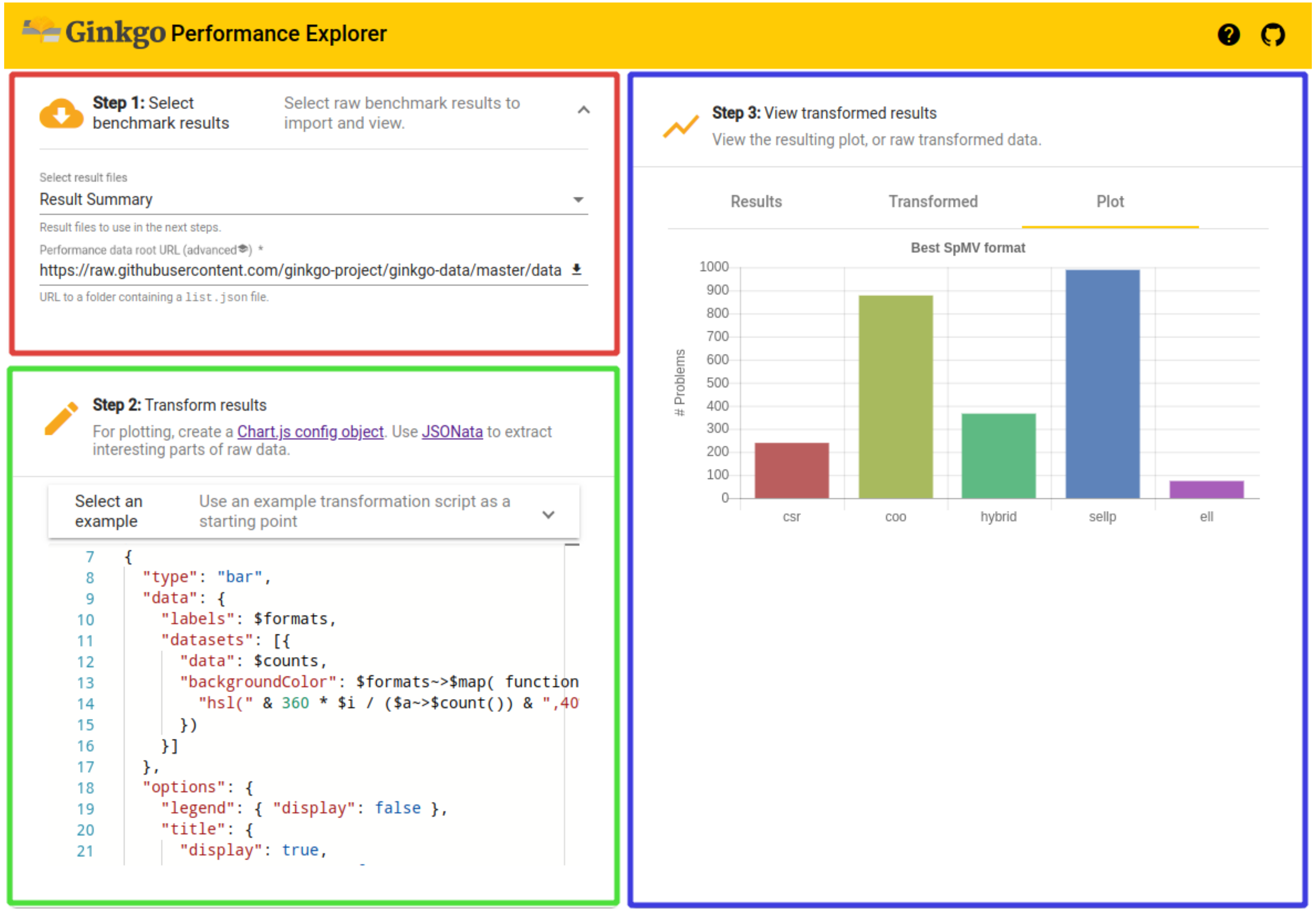
Ginkgo Performance Explorer (GPE)

Data Selection Tab

1. Select Data in Git repository.

Transformation Script Editor

2. Write JSONata script to visualize data (examples are provided).



Ginkgo Performance Explorer

Step 1: Select benchmark results

Select raw benchmark results to import and view.

Select result files: Result Summary

Result files to use in the next steps.

Performance data root URL (advanced) *: https://raw.githubusercontent.com/ginkgo-project/ginkgo-data/master/data

URL to a folder containing a `list.json` file.

Step 2: Transform results

For plotting, create a Chart.js config object. Use JSONata to extract interesting parts of raw data.

Select an example: Use an example transformation script as a starting point

```
{
  "type": "bar",
  "data": {
    "labels": $formats,
    "datasets": [{
      "data": $counts,
      "backgroundColor": $formats~>$map( function
        "hsl(" & 360 * $i / ($a~>$count()) & ",40
      })
    }]
  },
  "options": {
    "legend": { "display": false },
    "title": {
      "display": true,
```

Step 3: View transformed results

View the resulting plot, or raw transformed data.

Data and Plot Viewer

GPE

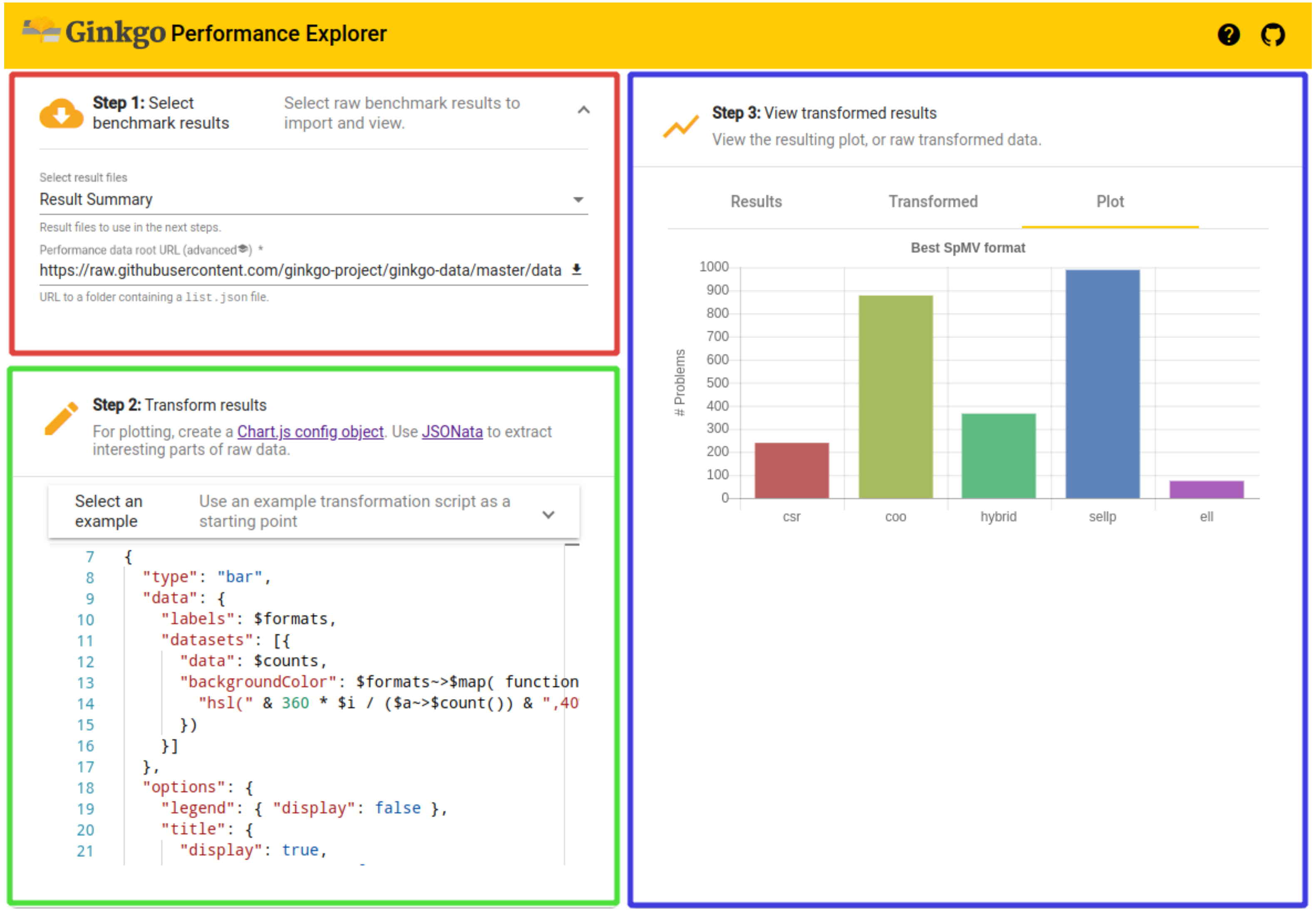
Ginkgo Performance Explorer (GPE)

Data Selection Tab

1.
Select Data in Git repository.

Transformation Script Editor

2.
Write JSONata script to visualize data (examples are provided).



Step 1: Select benchmark results

Select raw benchmark results to import and view.

Select result files: Result Summary

Result files to use in the next steps.

Performance data root URL (advanced) *: https://raw.githubusercontent.com/ginkgo-project/ginkgo-data/master/data

URL to a folder containing a `list.json` file.

Step 2: Transform results

For plotting, create a Chart.js config object. Use JSONata to extract interesting parts of raw data.

Select an example: Use an example transformation script as a starting point

```
{
  "type": "bar",
  "data": {
    "labels": $formats,
    "datasets": [{
      "data": $counts,
      "backgroundColor": $formats~>$map( function
        "hsl(" & 360 * $i / ($a~>$count()) & ",40
      })
    }]
  },
  "options": {
    "legend": { "display": false },
    "title": {
      "display": true,
```

Step 3: View transformed results

View the resulting plot, or raw transformed data.

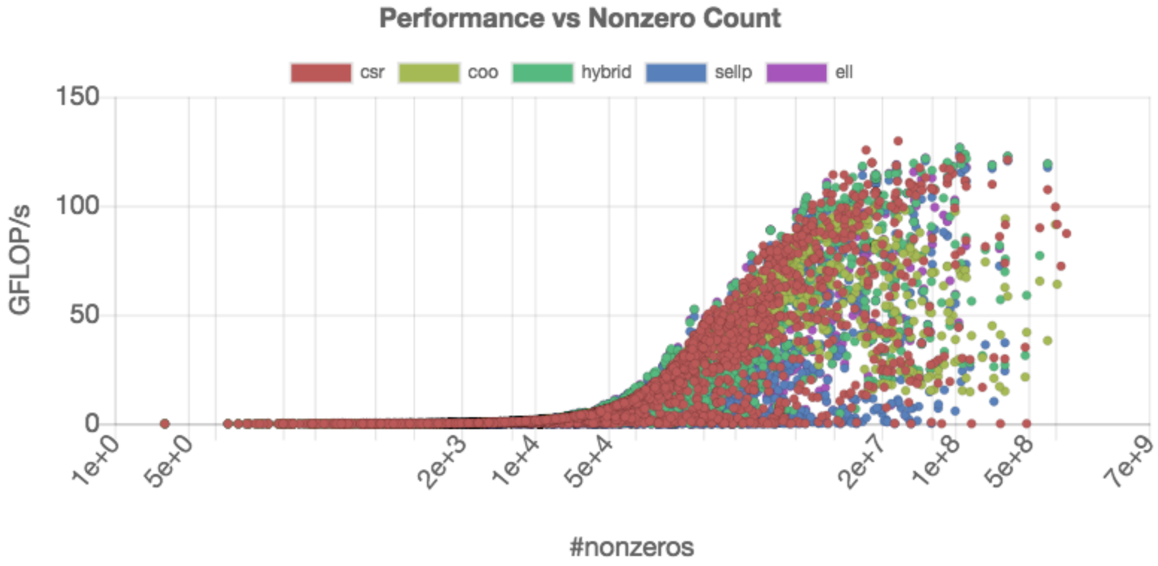
Results | Transformed | Plot

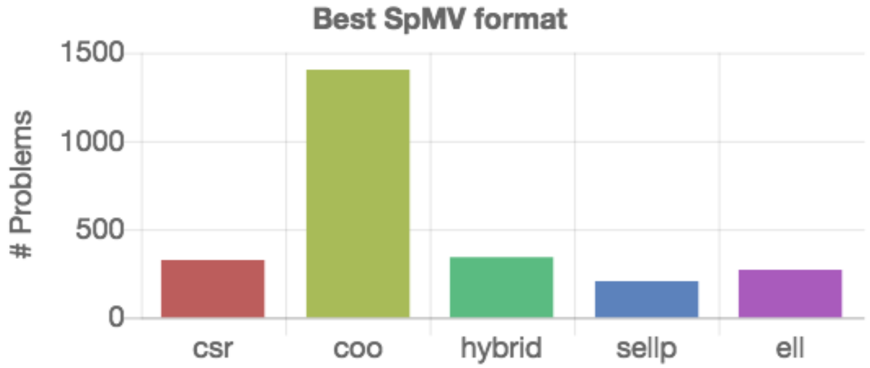
Data and Plot Viewer

3.
Analyze data visually.

GPE

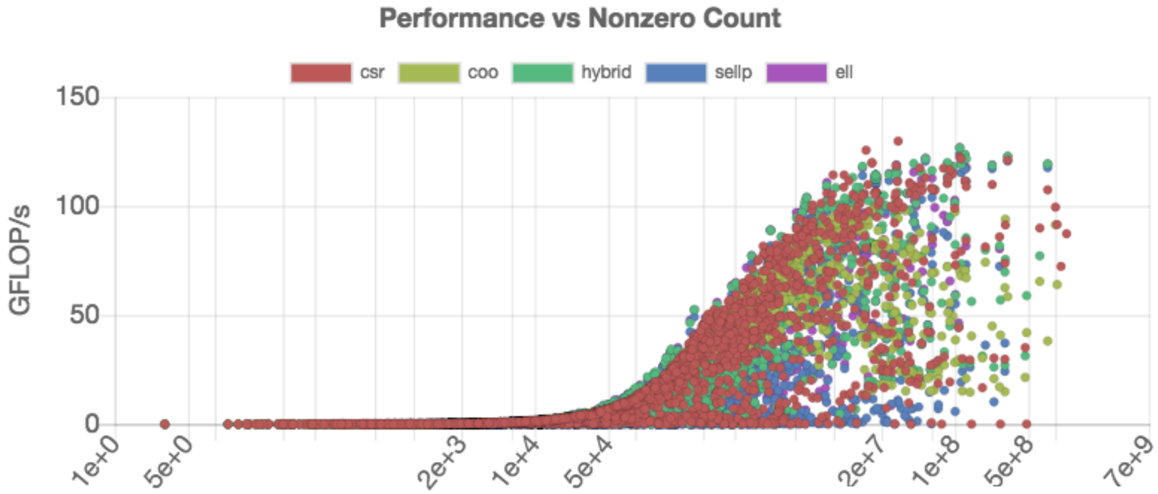
Ginkgo Performance Explorer (GPE)

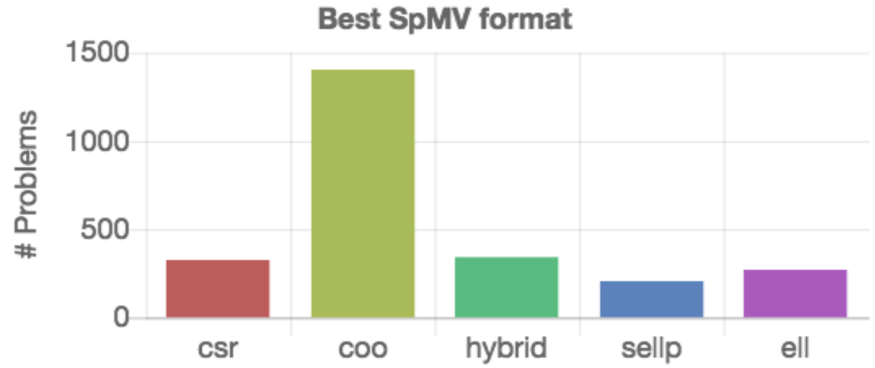


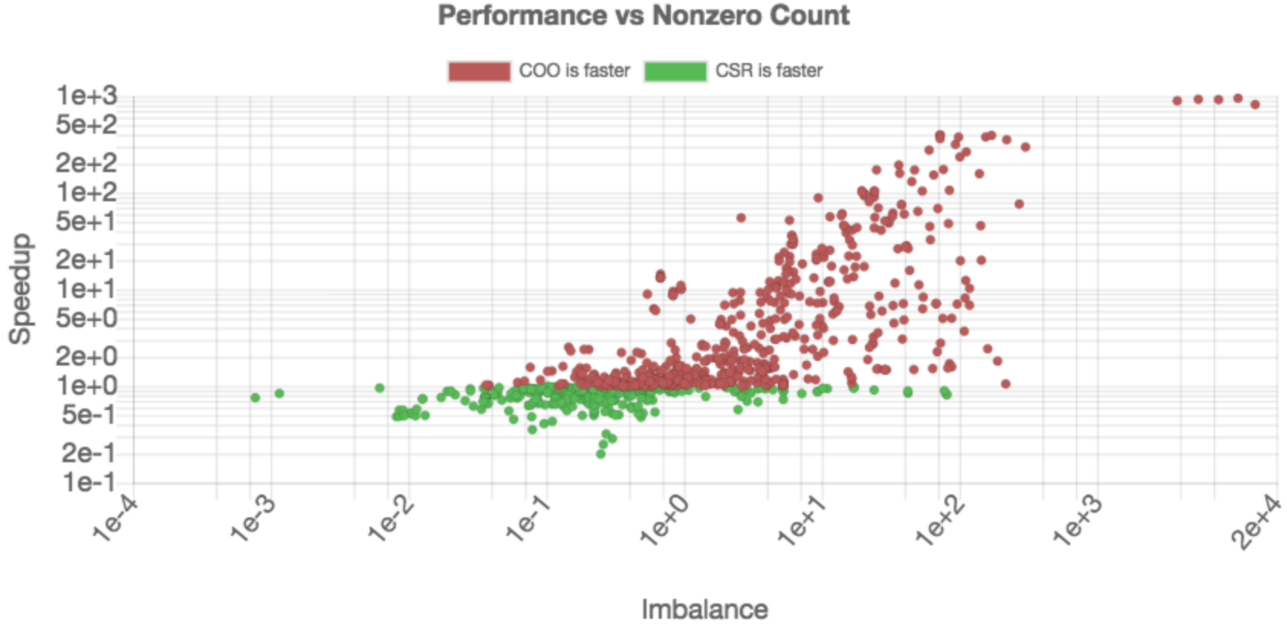


GPE

Ginkgo Performance Explorer (GPE)

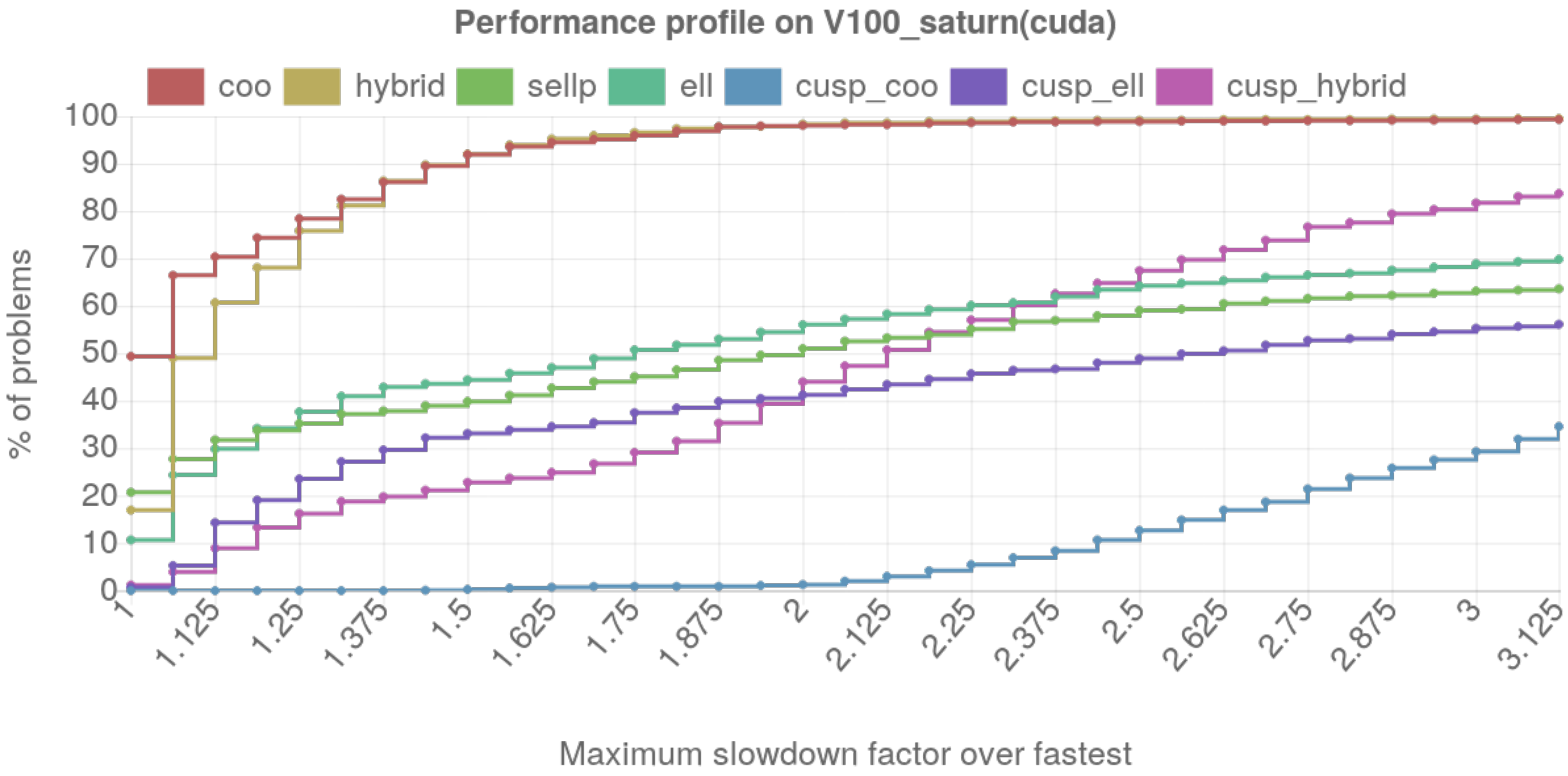






GPE

Ginkgo Performance Explorer (GPE)



Dolan & More: Benchmarking optimization software with performance profiles

GPE

Ginkgo’s Performance Evaluation Framework

Continuous Benchmarking Benefits

- Archiving performance data along with execution parameters ensures **full benchmark reproducibility**.
- **Comparing** the performance results over the code lifetime identifies **performance degradations**.
- **Ease of use**: the setup allows to launch benchmark with few clicks.

GPE

Ginkgo’s Performance Evaluation Framework

Continuous Benchmarking Benefits

- Archiving performance data along with execution parameters ensures **full benchmark reproducibility**.
- **Comparing** the performance results over the code lifetime identifies **performance degradations**.
- **Ease of use**: the setup allows to launch benchmark with few clicks.

Ginkgo Performance Explorer (GPE) Benefits

- The design of GPE efficiently realizes the analysis as **web service**, removing the need for downloading performance data to local disk or installing additional software.
- **External developers** without access to HPC systems can test and engineer **their codes on HPC resources**.
- **Extensibility**: Option to **compare** performance with **other software** libraries.

GPE

Anzt et al: **“Towards Continuous Benchmarking: An Automated Performance Evaluation Framework for High Performance Software”**, PASC 2019, accepted.

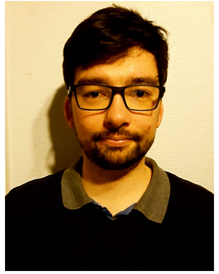
Learn More about Ginkgo

- Open-source C++ framework for sparse linear algebra.
- Sparse linear solvers, preconditioners, SpMV etc.
- Generic algorithm implementation:
  + reference kernels for checking correctness;
  + architecture-specific highly optimized kernels.
- Focused on GPU accelerators (i.e. NVIDIA GPUs).
- Software quality and sustainability efforts guided by xSDK community policies:
  - open source BSD
  - Doxygen
  - CMake Cross-platform Make
  - googletest Google C++ Testing Framework GitLab CI

https://xsdk.info/

Ginkgo

https://ginkgo-project.github.io/


Terry Cojean

Goran Flegar

Thomas Grützmacher

Pratik Nayak

Tobias Ribizel

better scientific software

https://bssw.io/