

LMU

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

FACULTY OF PHYSICS

EXPERIMENTAL FLAVOR PHYSICS



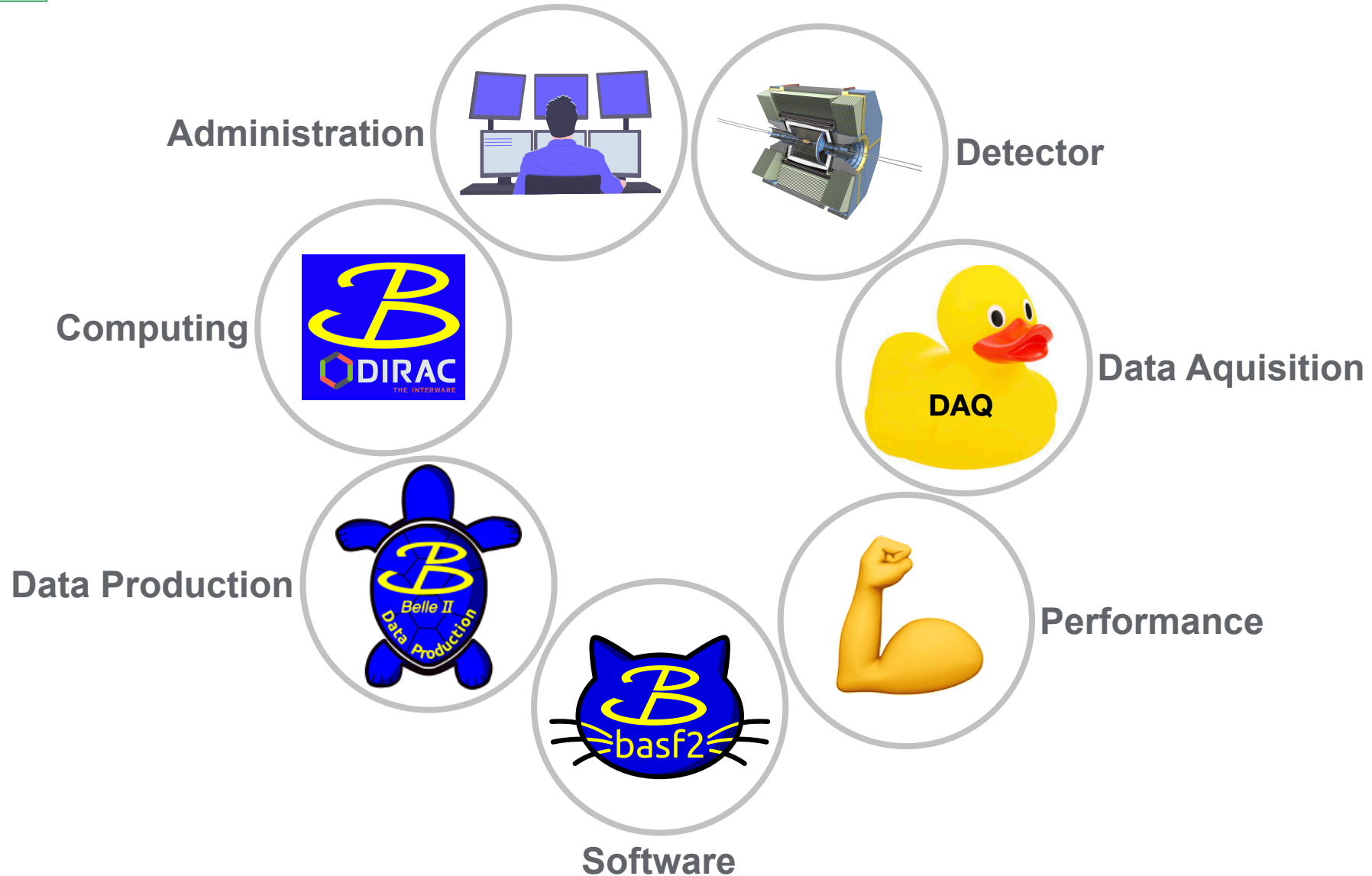
Software Development at Belle II

Belle II Academy 2026
04.05.2026

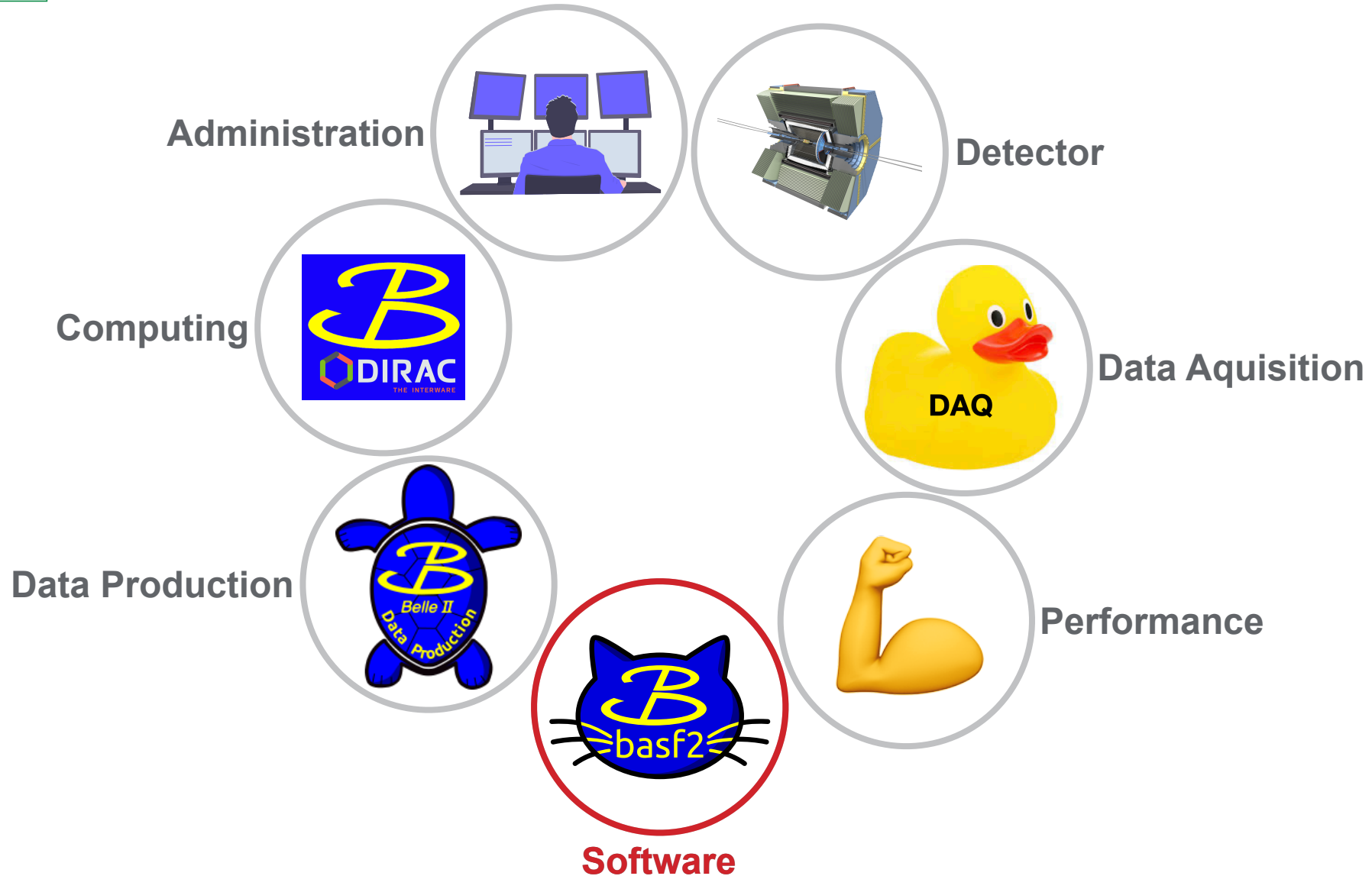
Alexander Heidelberg, Giacomo De Pietro, Thomas Kuhr



Where do we have Software in Belle II?



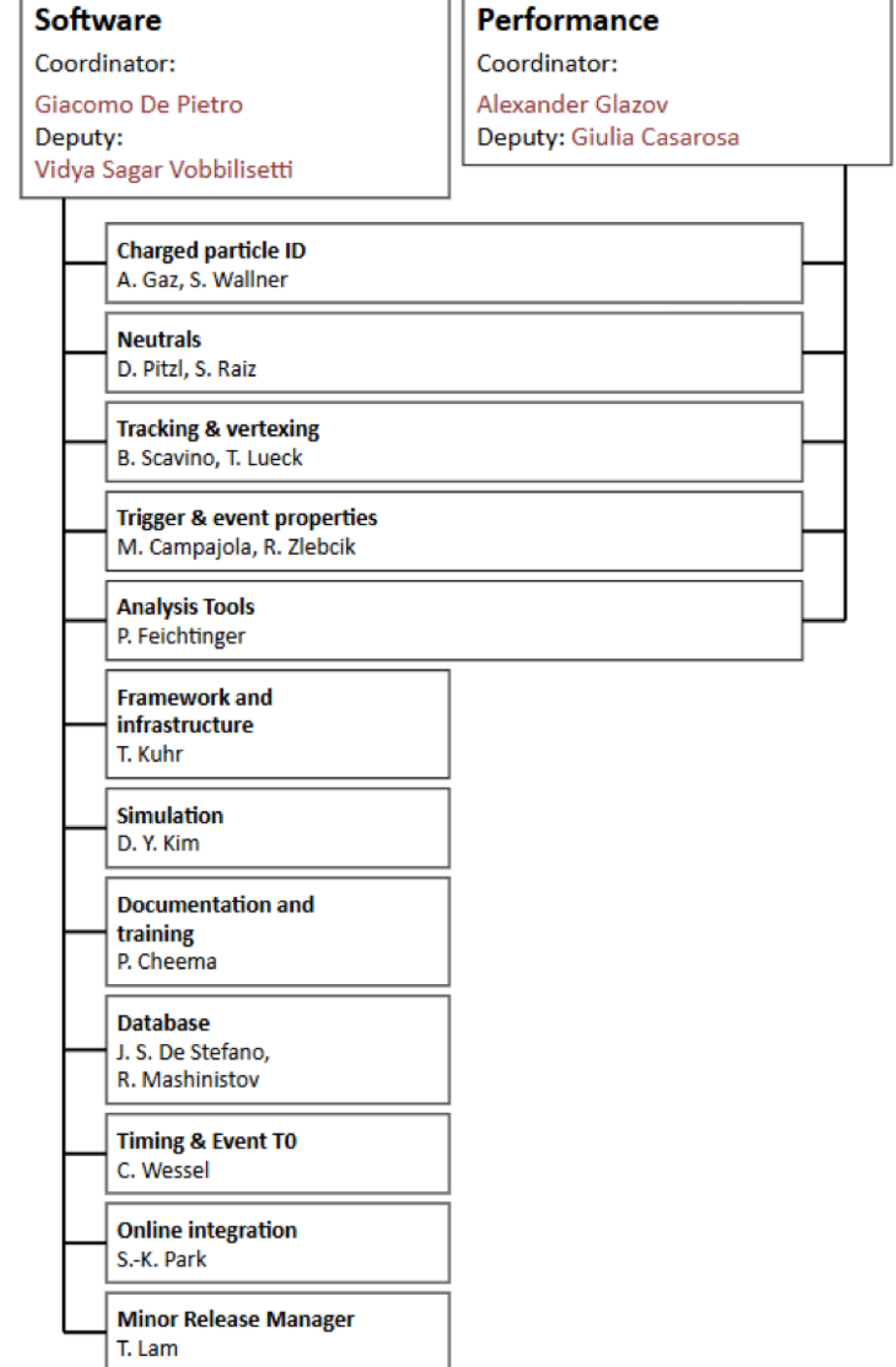
Where do we have Software in Belle II?





The Software Team

- Few **shared** groups between **software** and **performance**
- Few additional “software-only” groups
 - Mostly “one-man-band” than actual groups...
- basf2 librarians are also considered part of the software group
 - 37 packages: each package has between 1-3 librarians





LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

The Software Packages

Belle II / Software



Software



Subgroups and projects Shared projects Shared groups Inactive

Search (3 character minimum) Name

C Conditions Database	B belle_legacy	S SysVar
E examples-data-creation	D display	T tools
M MVA training scripts	D docker-images	T topoana
TRACKING Tracking	E examples-data	V validation-data
T Training	E externals	V versioning
b2luigi Owner b2luigi - bringing batch 2 luigi	F FastBDT	
B basf2	L LightReleaseValidation	
B belle2style	S starterkit-data	

Main Focus

- basf2
- externals
- tools
- b2luigi
- buildbot
- SysVar
- FastBDT
- ...



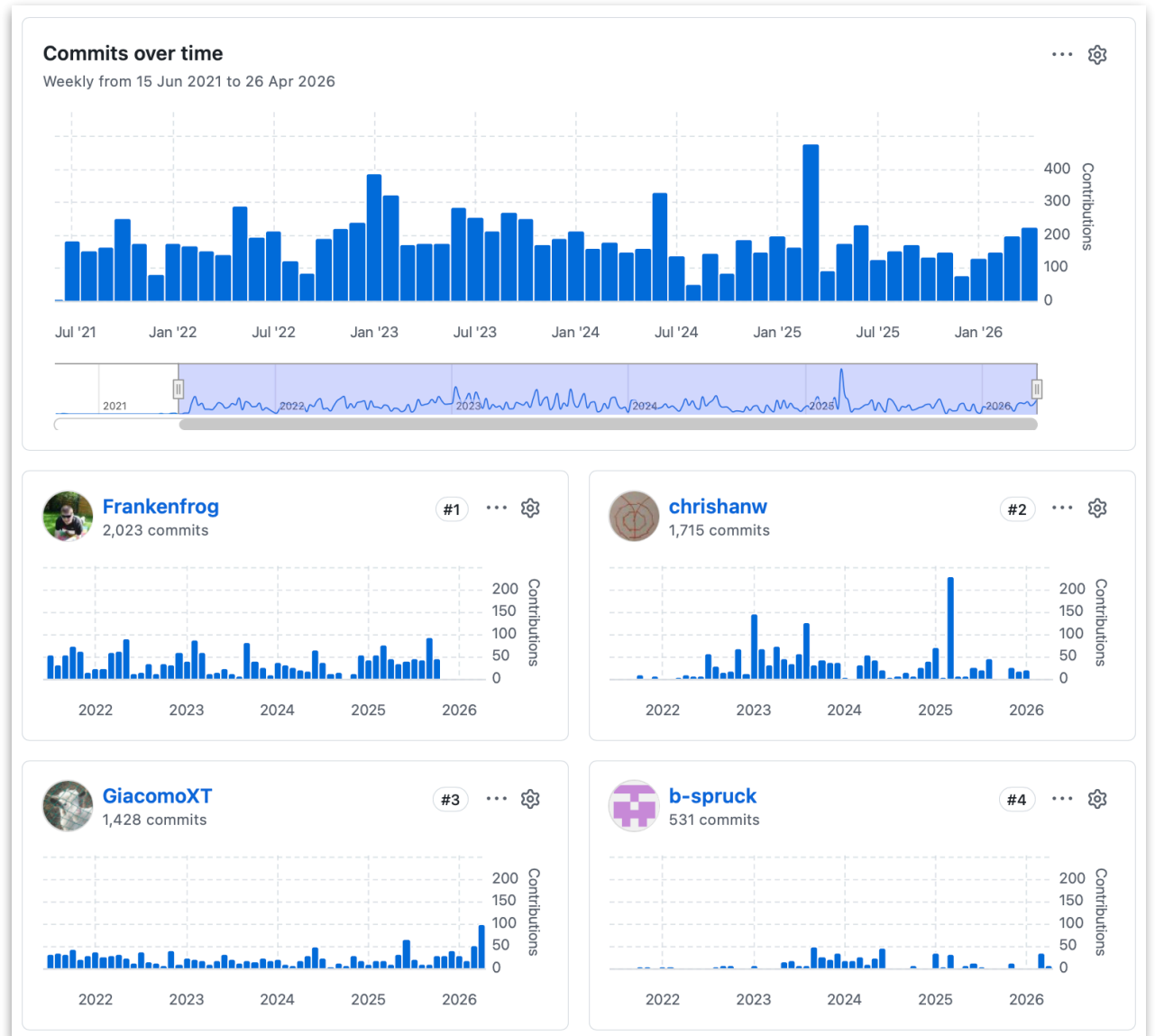
The largest software project in the collaboration

- Development started more than 10 years ago
- 37 packages:
 - **Core:** framework, site_scons
 - **Subdetector:** arich, cdc, ecl, klm, pxd, svd, top, vxd
 - **Data Taking:** daq, hlt, rawdata, trg
 - **Data Quality:** alignment, calibration, dqm, validation
 - **Data Storage:** mdst, skim
 - **MC:** decfiles, generators, geometry, simulation, structure
 - **Background:** background, beast, ir
 - **Offline Analysis:** analysis, b2bii, mva
 - **Documentation:** display, masterclass, online_book
 - **Reconstruction:** genfit2, reconstruction, tracking



basf2 in Numbers

- ~58k commits up to July 2021
- ~20k commits since July 2021
- Many lines of code...
 - ~54% of code written in C++
 - ~27% in Fortran
 - ~16% in Python
 - rest in C, RST, etc.
- Largely unit-tested
 - 1011 “Google” tests
 - 79 framework tests
 - 305 non-framework tests





basf2 Releases

Light Release

- It contains only a subset of the packages
- One tag every ~2 months

Full Release

- It contains all the packages
- One major tag per year

Upgrade Release

- Tag of the upgrade branch
- Used for developing and testing new features/geometries for the Belle II upgrade

Sphinx documentation

Latest light release

light-2603-ina (recommended)

light-2601-hyperion

light-2511-gacrux

light-2509-fornax

light-2507-europa

light-2505-deimos

release-09-00-15 (recommended)

release-08-03-00

release-08-02-06

release-08-01-10

release-08-00-10

release-06-02-00

release-06-01-16

release-06-00-14

prerelease-10-00-00a

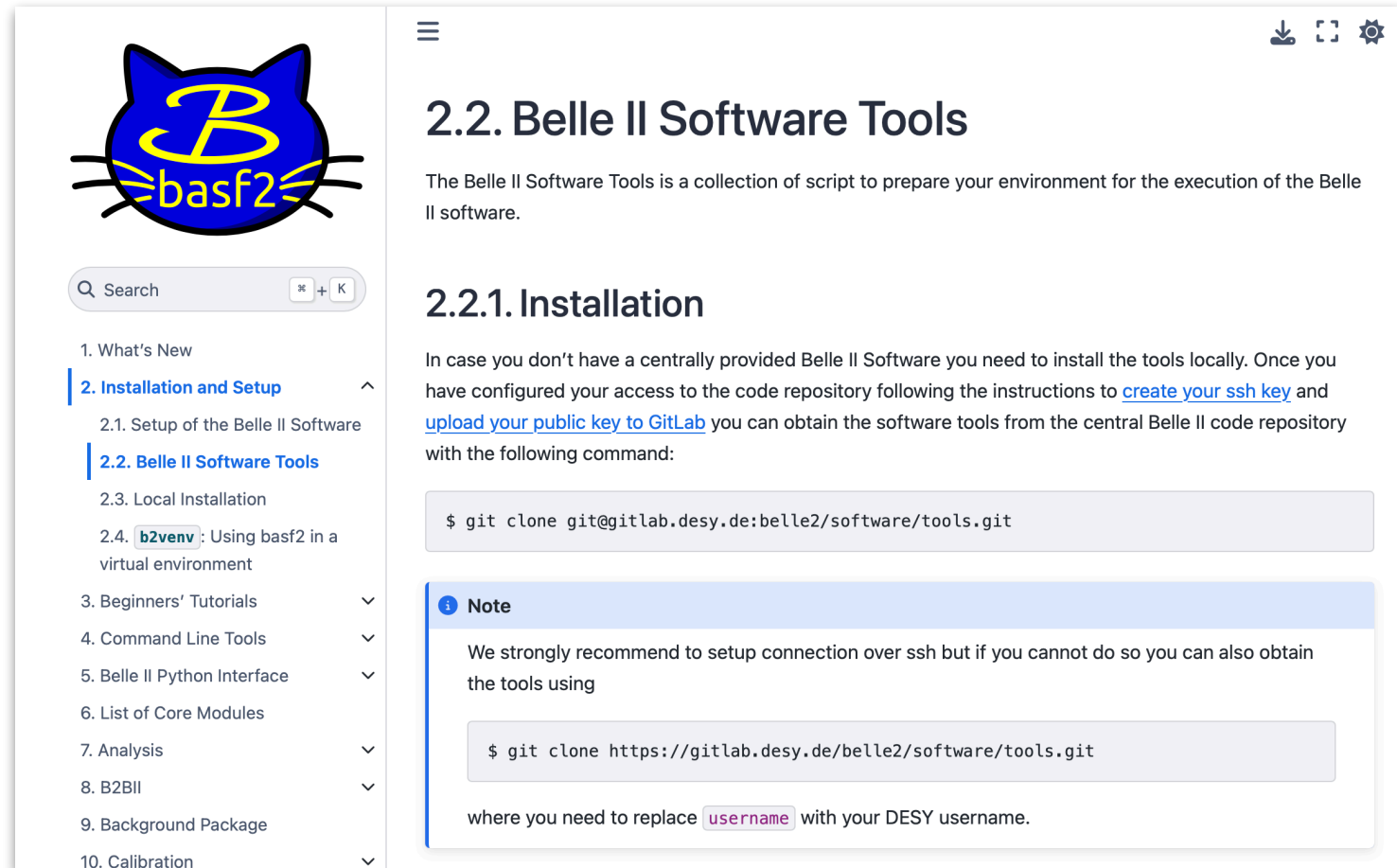
development


Set of basic tools for setting up basf2

- Analysis
- Development

Some Tools

- b2setup
- b2install-prepare
- b2execute
- b2venv
- ...





Q Search * + K

1. What's New
- 2. Installation and Setup**
 - 2.1. Setup of the Belle II Software
 - 2.2. Belle II Software Tools**
 - 2.3. Local Installation
 - 2.4. **b2venv**: Using basf2 in a virtual environment
3. Beginners' Tutorials
4. Command Line Tools
5. Belle II Python Interface
6. List of Core Modules
7. Analysis
8. B2BI
9. Background Package
10. Calibration

2.2. Belle II Software Tools

The Belle II Software Tools is a collection of script to prepare your environment for the execution of the Belle II software.

2.2.1. Installation

In case you don't have a centrally provided Belle II Software you need to install the tools locally. Once you have configured your access to the code repository following the instructions to [create your ssh key](#) and [upload your public key to GitLab](#) you can obtain the software tools from the central Belle II code repository with the following command:

```
$ git clone git@gitlab.desy.de:belle2/software/tools.git
```

Note

We strongly recommend to setup connection over ssh but if you cannot do so you can also obtain the tools using

```
$ git clone https://gitlab.desy.de/belle2/software/tools.git
```

where you need to replace `username` with your DESY username.

The code to download and compile the “external” packages on which basf2 depends

- We currently have 78 packages we compile ourselves:
 - compilers (gcc, clang)
 - common HEP software (ROOT, Geant4, CLHEP, ...)
 - generators (EvtGen, PYTHIA, PHOTOS, MadGraph, WHIZARD, Herwig, etc.)
 - many other libraries (curl, zlib, cmake, Git, Eigen, Boost, XRootD, etc.)
 - Python
- We also include 268 Python packages
 - Many packages are included by “popular demand” from users



buildbot is an open-source software for building CI/CD pipelines

- build of basf2 releases
- deployment on cvmfs of basf2 releases, externals builds and updated tools
- update of globaltags cache on CVMFS
- build and deployment of many documentations
- trigger of docker images build and their deployment on CVMFS
- 4 main workers + KEKCC + LMU

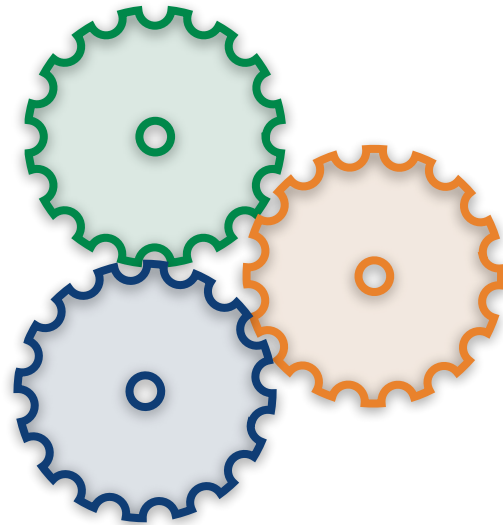
Builder Name	Builds	Tags	Workers
b2luigi			9
benchmark	1116 1115 1114		5
build-clang	2171 2170 2169 2168 2167 2166 2165 2164 2163 2162 2161 2160 2159 2158 2157		2
build-debug	2206 2205 2204 2203 2202 2201 2200 2199 2198 2197 2196 2195 2194 2193 2192		11
build-intel	2226 2224 2223 2222 2221 2220 2219 2218 2217 2216 2215 2214 2213 2212 2211		4
build-light	2227 2226 2225 2224 2223 2222 2221 2220 2219 2218 2217 2216 2215 2214 2213		9
cvmfs-conditions	1258 1257 1256 1255		3
cvmfs-externals			3
cvmfs-images			3
cvmfs-release	426 425 424		3
cvmfs-tools			3
cvmfs-versioning	76 75 74 73		3
development	1136 1135 1134		9
display			10
docker-images			1 2 3 4 5 9 10 11
externals			1

- Helper package constructed around the popular workflow manager luigi
 - Main purpose is to ease building the task trees in everyday scenarios
- Since November 2023, maintenance has been in the hands of Belle II

Workflow I/O

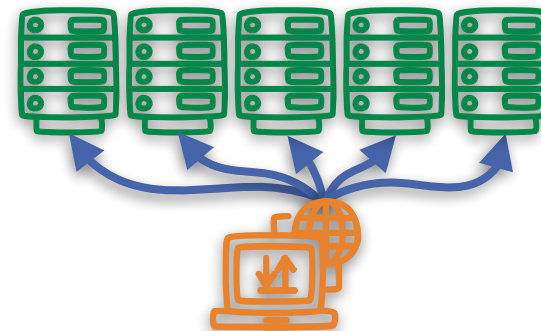
```

my_project=foo/
├─ parameter=1/
│   ├── other_parameter=a/
│   └── other_parameter=b/
├─ parameter=2/
│   ├── other_parameter=a/
│   └── other_parameter=b/
└─ parameter=3/
    
```



Workflow Steering

Distribute Workflow



User Interface

Python package that streamlines the treatment of systematic uncertainties

- Corrections from the performance groups will be provided in a format that SysVar can handle
 - SysVar will be the recommended tool for applying the necessary offline corrections!

SysVar: A tool enhancing consistency in the treatment of systematics

SysVar is a python based tool that allows to:



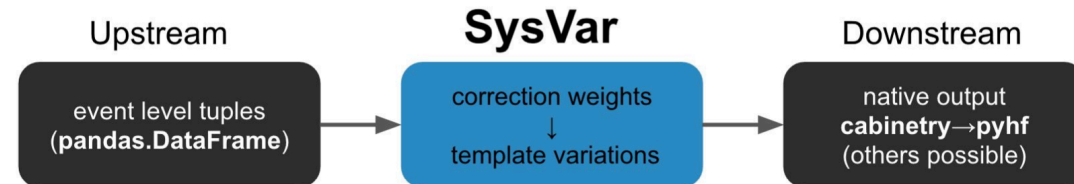
Anyone who corrects MC offline

Useful for



Anyone who runs template fits

1. Apply Data/MC corrections to a DataFrame
2. Generate Variations of Data/MC Corrections
3. Histogram MC to build templates and template variations for a non-parametric fit.
4. Diagonalize a cov matrix (channels x templates x bins) to produce eigenvariations and save them to implement Correlated Shape Variations
5. Identify the number of necessary eigendirections that the user should consider for an accurate analysis (different criteria are available)



SysVar sits between event-level data and the statistical model builder, converting correlated weight systematics into template variations

Stochastic gradient-boosted decision trees for multivariate classification, usable standalone and via Python interface

- Last summer, Belle II forked the original FastBDT repository from Thomas Keck and started maintaining its own version
- No new features or functional modifications are currently planned

```
for i in range(len(mean)):
    for j in range(i+1, len(mean)):
        cov[j][i] = cov[i][j]

N_train, N_test = 10000, 10000
data = np.random.multivariate_normal(mean, cov, N_train + N_test)
X_train, y_train = data[:N_train, 1:], data[:N_train, 0] > 0
X_test, y_test = data[N_train:, 1:], data[N_train:, 0] > 0

# Train FastBDT using its PythonInterface, which is based on the SKLearn classifiers
clf = FastBDT.Classifier()
clf.fit(X=X_train, y=y_train)
p = clf.predict(X_test)
global_auc = FastBDT.calculate_roc_auc(p, y_test)
print("Global AUC", global_auc)
```

A black rectangular sign with rounded corners is suspended by a yellow string. The sign features the words "HELP" and "WANTED" in a bold, orange, sans-serif font, stacked vertically. The background is a blurred indoor setting with warm lighting and a person's silhouette.

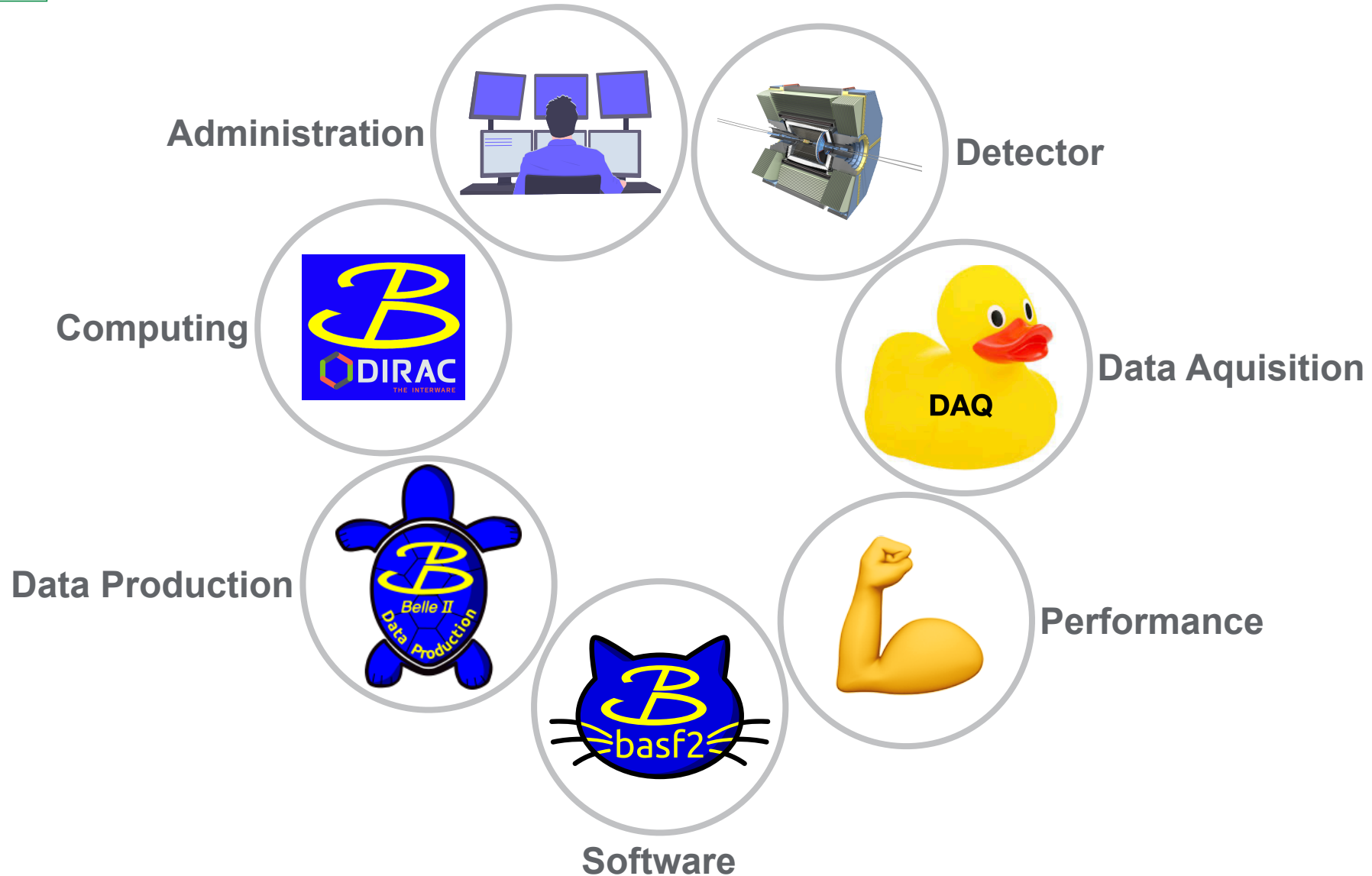
**HELP
WANTED**

A black rectangular sign hangs from a yellow string. The sign features the words "HELP WANTED" in large, bold, orange capital letters. A thick, red, brushstroke-like line is drawn across the sign, crossing out the word "WANTED".

**HELP
WANTED**

NEEDED

Where do we ~~have~~ need Software in Belle II?



What is Software Development?

Requirements

Testing

Design

CI/CD

Coding

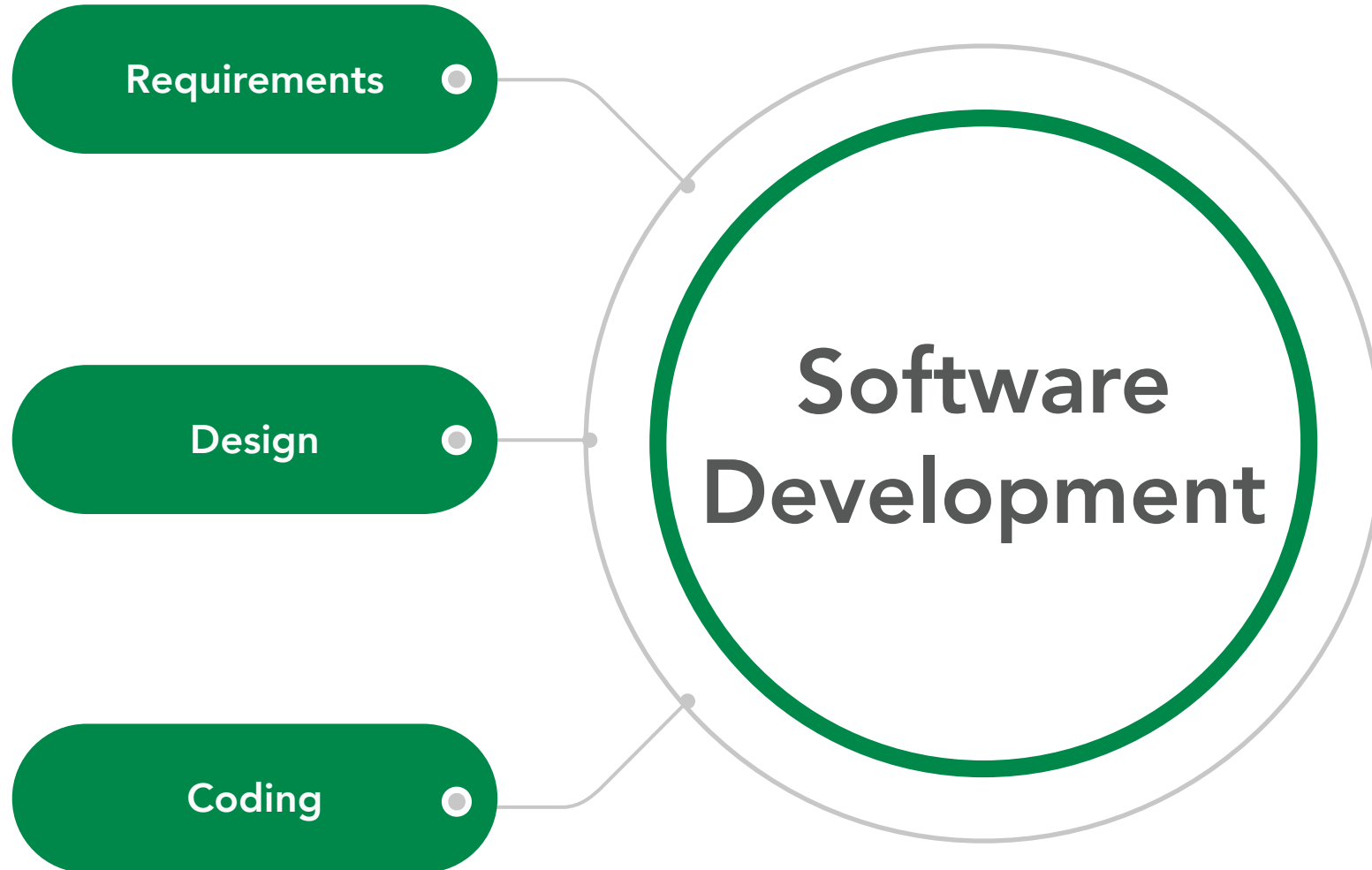
Documentation

**Software
Development**

What is Software Development?



What is Software Development?



Testing

CI/CD

Documentation

What is Software Development?



What is Software Development?



What is Software Development?



What is Software Development?



Not part of today

What is Software Development?



Not part of today

What is Software Development?



Does this sound familiar?

```

EXPLORER
  MYANALYSIS
    configs
    scripts
    src/myanalysis
      extraction
      fitting_tools
      mc_production
      mva_tools
      plotting
      selection
      utilities
        __pycache__
        base_task.py
        basics.py
        datahandler.py
        dataset.py
        modelparameters.py
        particles.py
        process.py
        variables.py
      validation
  modelparameters.py 2
    7
    8 @dataclass
    9 class ModelParameter:
    10     value: float = 0.0
    11     variable: Variable = Variable(name="m", latex_name="m", unit="GeV")
    12
    13     @property
    14     def as_dict(self) -> Dict[str, Union[float, Dict[str, str]]]:
    15         return {"value": self.value, "variable": self.variable.as_dict}
    16
    17     @classmethod
    18     def from_dict(
    19         cls, mp_dict: Dict[str, Union[float, Dict[str, str]]]
    20     ) -> "ModelParameter":
    21         value = mp_dict.get("value")
    22         variable = mp_dict.get("variable")
    23         assert isinstance(value, float)
    24         assert isinstance(variable, dict)
    25         return cls(value, Variable.from_dict(variable))
    26
    27
    28
    29
    30 class ModelParameterCollection:
    31     def __init__(
    32         self,
    33         variable: Variable = Variable(name="m", latex_name="m", unit="GeV"),
    34         values: Optional[List[float]] = None,
    35         range: Optional[List[float]] = None,
  
```



Does this sound familiar?

```

EXPLORER
  MYANALYSIS
    configs
    scripts
    src/myanalysis
      extraction
      fitting_tools
      mc_production
      mva_tools
      plotting
      selection
      utilities
      __pycache__
      base_task.py
      basics.py
      datahandler.py
      dataset.py
      modelparameters.py
      particles.py
      process.py
      variables.py
      validation
  modelparameters.py 2
    7
    8 @dataclass
    9 class ModelParameter:
    10     value: float = 0.0
    11     variable: Variable = Variable(name="m", latex_name="m", unit="GeV")
    12
    13     @property
    14     def as_dict(self) -> Dict[str, Union[float, Dict[str, str]]]:
    15         return {"value": self.value, "variable": self.variable.as_dict}
    16
    17     @classmethod
    18     def from_dict(
    19         cls, mp_dict: Dict[str, Union[float, Dict[str, str]]]
    20     ) -> "ModelParameter":
    21         value = mp_dict.get("value")
    22         variable = mp_dict.get("variable")
    23         assert isinstance(value, float)
    24         assert isinstance(variable, dict)
    25         return cls(value, Variable.from_dict(variable))
    26
    27
    28
    29
    30 class ModelParameterCollection:
    31     def __init__(
    32         self,
    33         variable: Variable = Variable(name="m", latex_name="m", unit="GeV"),
    34         values: Optional[List[float]] = None,
    35         range: Optional[List[float]] = None,
  
```



Does this sound familiar?

```

EXPLORER
  MYANALYSIS
    configs
    scripts
    src/myanalysis
      extraction
      fitting_tools
      mc_production
      mva_tools
      plotting
      selection
      utilities
        __pycache__
        base_task.py
        basics.py
        datahandler.py
        dataset.py
        modelparameters.py
        particles.py
        process.py
        variables.py
      validation
  modelparameters.py 2
    7
    8
    9 @dataclass
    10 class ModelParameter:
    11     value: float = 0.0
    12     variable: Variable = Variable(name="m", latex_name="m_{s1}", unit="GeV/
    13
    14     @property
    15     def as_dict(self) -> Dict[str, Union[float, Dict[str, str]]]:
    16         return {"value": self.value, "variable": self.variable.as_dict}
    17
    18     @classmethod
    19     def from_dict(
    20         cls, mp_dict: Dict[str, Union[float, Dict[str, str]]]
    21     ) -> "ModelParameter":
    22         value = mp_dict.get("value")
    23         variable = mp_dict.get("variable")
    24         assert isinstance(value, float)
    25         assert isinstance(variable, dict)
    26         return cls(value, Variable.from_dict(variable))
    27
    28
    29
    30 class ModelParameterCollection:
    31     def __init__(
    32         self,
    33         variable: Variable = Variable(name="m", latex_name="m_{s1}", unit="
    34         values: Optional[List[float]] = None,
    35         range: Optional[List[float]] = None,
  
```

There is a new parameter in the MC files. Can you propagate it through your workflow?



Does this sound familiar?

```

EXPLORER
  MYANALYSIS
    configs
    scripts
    src/myanalysis
      extraction
      fitting_tools
      mc_production
      mva_tools
      plotting
      selection
      utilities
      __pycache__
      base_task.py
      basics.py
      datahandler.py
      dataset.py
      modelparameters.py
      particles.py
      process.py
      variables.py
      validation
  modelparameters.py 2
    7
    8
    9 class ModelParameter:
    10     value: float = 0.0
    11     variable: Variable = Variable(name="m", latex_name="m_{as}", unit="GeV/
    12
    13     @property
    14     def as_dict(self) -> Dict[str, Union[float, Dict[str, str]]]:
    15         return {"value": self.value, "variable": self.variable.as_dict}
    16
    17     @classmethod
    18     def from_dict(cls, mp_dict: Dict[str, Union[float, Dict[str, str]]]
    19     ) -> "ModelParameter":
    20         value = mp_dict.get("value")
    21         variable = mp_dict.get("variable")
    22         assert isinstance(value, float)
    23         assert isinstance(variable, dict)
    24         return cls(value, Variable.from_dict(variable))
    25
    26
    27
    28
    29
    30 class ModelParameterCollection:
    31     def __init__(
    32         self,
    33         variable: Variable = Variable(name="m", latex_name="m_{as}", unit="
    34         values: Optional[List[float]] = None,
    35         range: Optional[List[float]] = None,
  
```

Sure! No problem. I just have to change the output of one method!



Does this sound familiar?

```

EXPLORER
  MYANALYSIS
    configs
    scripts
    src / myanalysis
      extraction
      fitting_tools
      mc_production
      mva_tools
      plotting
      selection
      utilities
      __pycache__
      base_task.py
      basics.py
      datahandler.py
      dataset.py
      modelparameters.py
      particles.py
      process.py
      variables.py
      validation
  modelparameters.py 2
    7
    8 @dataclass
    18
    19 def from_dict(
    20     cls, mp_dict: Dict[str, Union[float, Dict[str, str]]]
    21 ) -> "ModelProperty":
    22     value = mp_dict.get("value")
    23     variable = mp_dict.get("variable")
    24     assert isinstance(value, float)
    25     assert isinstance(variable, dict)
    26     return cls(value, Variable.from_dict(variable))
    27
    28
    29
    30 class ModelParameterCollection:
    31     def __init__(
    32         self,
    33         variable: Variable = Variable(name="m", latex_name="m", unit="")
    34         values: Optional[List[float]] = None,
    35         range: Optional[List[float]] = None,
  
```

TypeError: MyMethod() takes 2 positional arguments but 3 were given

Sure! No problem. I just have to change the output of one method!



Does this sound familiar?

```

EXPLORER
  MYANALYSIS
    configs
    scripts
    src / myanalysis
      extraction
      fitting_tools
      mc_production
      mva_tools
      plotting
      selection
      utilities
      __pycache__
      base_task.py
      basics.py
      datahandler.py
      dataset.py
      modelparameters.py
      particles.py
      process.py
      variables.py
      validation
  modelparameters.py 2
    7
    8 @dataclass
    18
    19 def from_dict(
    20     cls, mp_dict: Dict[str, Union[float, Dict[str, str]]]
    21     ) -> "ModelParameter":
    22     value = mp_dict.get("value")
    23     variable = mp_dict.get("variable")
    24     assert isinstance(value, float)
    25     assert isinstance(variable, dict)
    26     return cls(value, Variable.from_dict(variable))
    27
    28
    29
    30 class ModelParameterCollection:
    31     def __init__(
    32         self,
    33         variable: Variable = Variable(name="m", latex_name="m", unit="")
    34         values: Optional[List[float]] = None,
    35         range: Optional[List[float]] = None,
  
```

TypeError: MyMethod() takes 2 positional arguments but 3 were given

Sure! No problem. I just
have to change the
one method.

IndexError: list index out of range



Does this sound familiar?

```

EXPLORER
  MYANALYSIS
    configs
    scripts
    src / myanalysis
      extraction
      fitting_tools
      mc_production
      mva_tools
      plotting
      selection
      utilities
      __pycache__
      base_task.py
      basics.py
      datahandler.py
      dataset.py
      modelparameters.py 2
      particles.py
      process.py
      variables.py
      validation
  modelparameters.py 2
    7
    8 @dataclass
    18
    19 def from_dict(
    20     cls, mp_dict: Dict[str, Union[float, Dict[str, str]]]
    21 ) -> "ModelParameter":
    22     value = mp_dict.get("value")
    23     variable = mp_dict.get("variable")
    24     assert isinstance(value, float)
    25     assert isinstance(variable, dict)
    26     return cls(value, Variable.from_dict(variable))
    27
    28
    29
    30 class ModelParameterCollection:
    31     def __init__(
    32         self,
    33         variable: Variable = Variable(name="m", latex_name="m", unit="")
    34         values: Optional[List[float]] = None,
    35         range: Optional[List[float]] = None,
  
```

TypeError: MyMethod() takes 2 positional arguments but 3 were given

Sure! No problem. I just
have to change the
one method.

IndexError: list index out of range

TypeError: object of type 'float' has no len()



Does this sound familiar?

```

EXPLORER
  MYANALYSIS
    configs
    scripts
    src/myanalysis
      extraction
      fitting_tools
      mc_production
      mva_tools
      plotting
      selection
      utilities
      __pycache__
      base_task.py
      basics.py
      datahandler.py
      dataset.py
      modelparameters.py
      particles.py
      process.py
      variables.py
      validation
  modelparameters.py 2
    7
    8 @dataclass
    18
    19 def from_dict(
    20     cls, mp_dict: Dict[str, Union[float, Dict[str, str]]]
    21 ) -> "ModelProperty":
    22     value = mp_dict.get("value")
    23     variable = mp_dict.get("variable")
    24     assert isinstance(value, float)
    25     assert isinstance(variable, dict)
    26     return cls(value, Variable.from_dict(variable))
    27
    28
    29
    30 class ModelParameterCollection:
    31     def __init__(
    32         self,
    33         variable: Variable = Variable(name="m", latex_name="m", unit="")
    34         values: Optional[List[float]] = None,
    35         range: Optional[List[float]] = None,
  
```

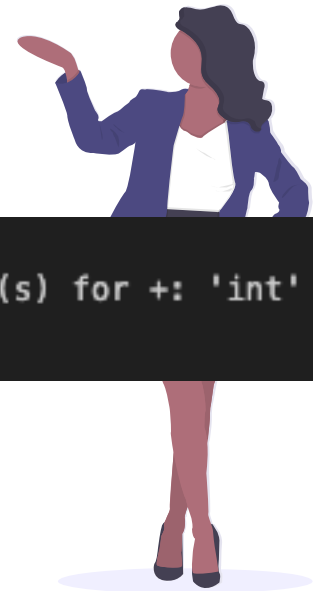
TypeError: MyMethod() takes 2 positional arguments but 3 were given

Sure! No problem. I just
have to change the
one method.

IndexError: list index out of range

TypeError: object of type 'float' has no len()

TypeError: unsupported operand type(s) for +: 'int' and 'str'



Does this sound familiar?

```

7
8 @dataclass
18
19 def from_dict(
20     cls, mp_dict: Dict[str, Union[float, Dict[str, str]]]
21 ) -> "ModelProperty":
22     value = mp_dict.get("value")
23     variable = mp_dict.get("variable")
24     assert isinstance(value, float)
25     assert isinstance(variable, dict)
26     return cls(value, Variable.from_dict(variable))
27
28
29
30 class ModelParameterCollection:
31     def __init__(
32         self,
33         variable: Variable = Variable(name="m", latex_name="m", unit="")
34         values: Optional[List[float]] = None,
35         range: Optional[List[float]] = None,

```

TypeError: MyMethod() takes 2 positional arguments but 3 were given

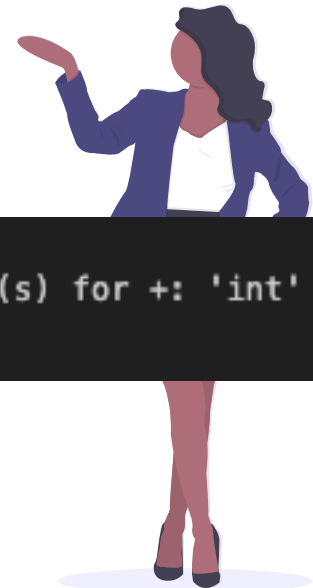
Sure! No problem. I just
have to change the
one method.

IndexError: list index out of range

TypeError: object of type 'float' has no len()

TypeError: unsupported operand type(s) for +: 'int' and 'str'

ZeroDivisionError: division by zero



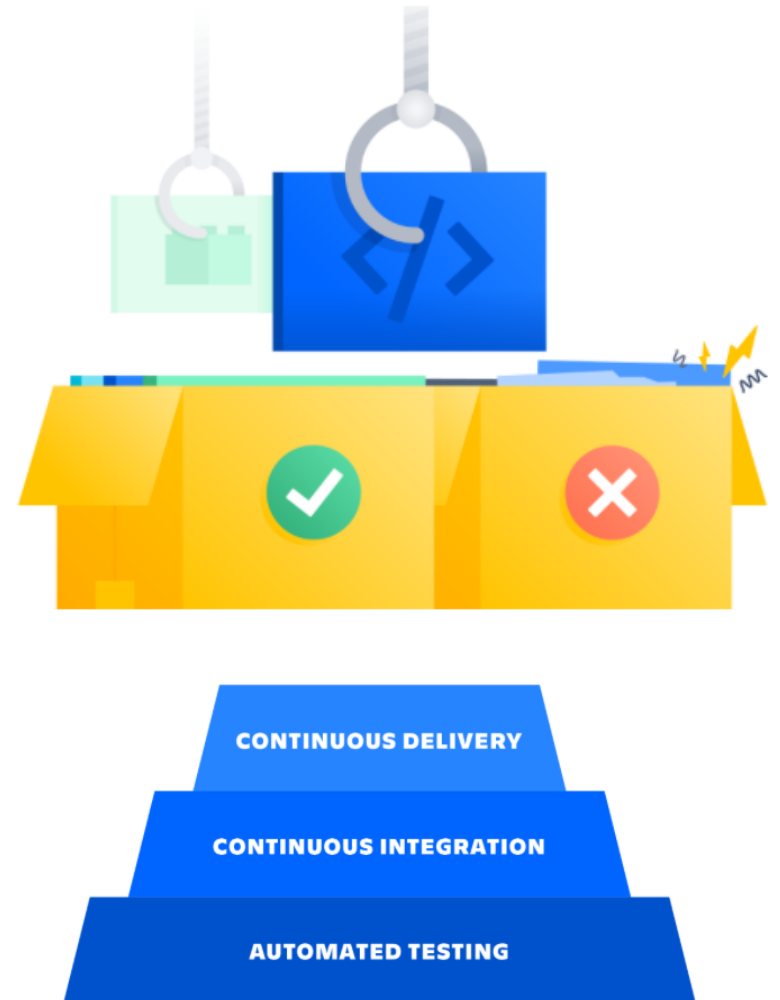
Testing is Common Practice

Testing code is as common as writing the code itself

- Before ~2000, companies had dedicated teams (much like Q&A) to make sure things worked manually

Automation in Industry

- Test-Driven Development
- Continuous Integration and Continuous Testing
- Artificial Intelligence in Testing



Testing is Common Practice

Testing code is as common as writing the code itself

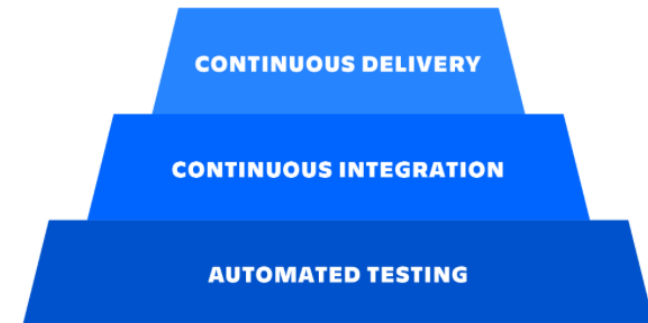
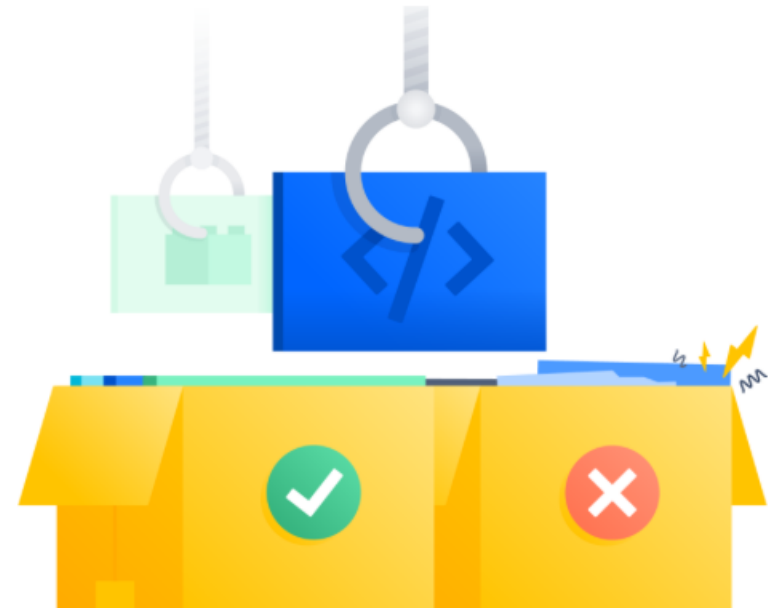
- Before ~2000, companies had dedicated teams (much like Q&A) to make sure things worked manually

Automation in Industry

- Test-Driven Development
- Continuous Integration and Continuous Testing
- Artificial Intelligence in Testing

Belle II

- Over time, almost every core software package is at least functionally tested





Benefits of Testing Code

Benefits of Testing Code

Time is Money!

- Complex code that grows over time
- Several weeks or months between connected projects
- Think of tests as an investment:
 - Each minute spent writing tests can save hours of frustration

Benefits of Testing Code

Time is Money!

- Complex code that grows over time
- Several weeks or months between connected projects
- Think of tests as an investment:
 - Each minute spent writing tests can save hours of frustration

Collaboration

- Several people are working on the same project
- Multiple branches add to the main
 - There is a high chance of adding a commit that breaks everything

Benefits of Testing Code

Time is Money!

- Complex code that grows over time
- Several weeks or months between connected projects
- Think of tests as an investment:
 - Each minute spent writing tests can save hours of frustration

Collaboration

- Several people are working on the same project
- Multiple branches add to the main
 - There is a high chance of adding a commit that breaks everything

Experience

- Learning to write tests is a valuable skill

Benefits of Testing Code

Time is Money!

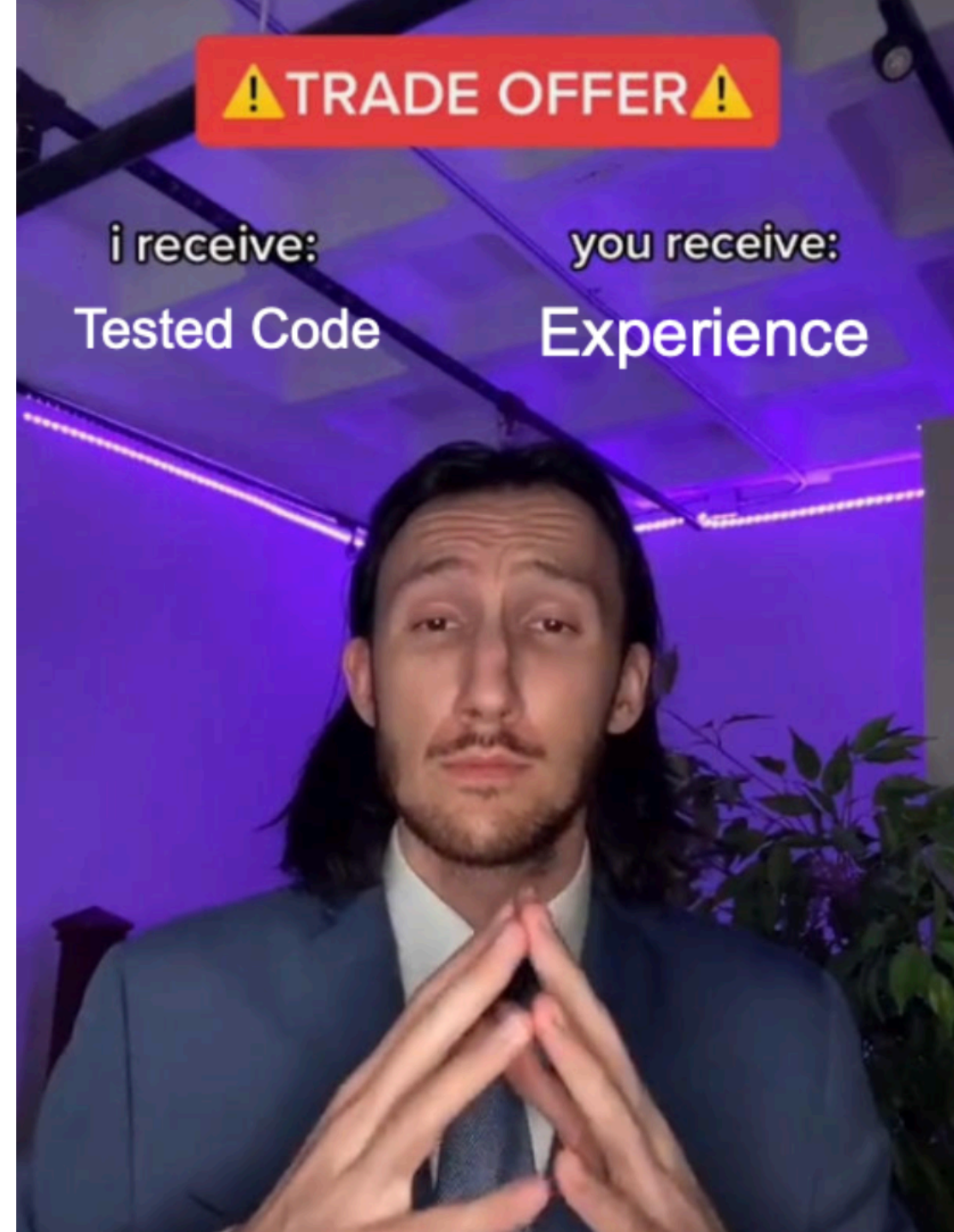
- Complex code that grows over time
- Several weeks or months between connected projects
- Think of tests as an investment:
 - Each minute spent writing tests can save hours of frustration

Collaboration

- Several people are working on the same project
- Multiple branches add to the main
 - There is a high chance of adding a commit that breaks everything

Experience

- Learning to write tests is a valuable skill





LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

What is a “Test”?



What is a “Test”?

“Software testing is the act of checking whether software meets its intended objectives and satisfies expectations.”

What is a “Test”?

“Software testing is the act of checking whether software meets its intended objectives and satisfies expectations.”

“Does the software do what it is supposed to do and what it needs to do?”

What is a “Test”?

“Software testing is the act of checking whether software meets its intended objectives and satisfies expectations.”

“Does the software do what it is supposed to do and what it needs to do?”

Tests happen...

What is a “Test”?

“Software testing is the act of checking whether software meets its intended objectives and satisfies expectations.”

“Does the software do what it is supposed to do and what it needs to do?”

Tests happen...

- ... automatic!

What is a “Test”?

“Software testing is the act of checking whether software meets its intended objectives and satisfies expectations.”

“Does the software do what it is supposed to do and what it needs to do?”

Tests happen...

- ... automatic!
- ... on different levels!
 - Unit, Integration, System

What is a “Test”?

“Software testing is the act of checking whether software meets its intended objectives and satisfies expectations.”

“Does the software do what it is supposed to do and what it needs to do?”

Tests happen...

- ... automatic!
- ... on different levels!
 - Unit, Integration, System
- ... in different formats!
 - Static, dynamic, passive,...

```
1  import unittest
2
3  def add(a, b):
4      return a + b
5
6  class TestMath(unittest.TestCase):
7      def test_add(self):
8          self.assertEqual(add(2, 3), 5)
9
10 if __name__ == "__main__":
11     unittest.main()
```

Hello World!



```

1  import unittest
2
3  def add(a, b):
4      return a + b
5
6  class TestMath(unittest.TestCase):
7      def test_add(self):
8          self.assertEqual(add(2, 3), 5)
9
10 if __name__ == "__main__":
11     unittest.main()

```

```

> pytest simple.py
===== test session starts =====
platform darwin -- Python 3.12.3, pytest-9.0.3, pluggy-1.6.0
rootdir: /Users/alexanderheidelbach/Local/BelleII_Academy
collected 1 item

simple.py . [100%]

===== 1 passed in 0.01s =====

```

Integration Test

```
1 import tempfile
2 from pathlib import Path
3
4 def save_result(filename, value):
5     Path(filename).write_text(str(value))
6
7 def load_result(filename):
8     return int(Path(filename).read_text())
9
10 def test_save_and_load_result():
11     with tempfile.NamedTemporaryFile() as tmp:
12         save_result(tmp.name, 42)
13         result = load_result(tmp.name)
14
15     assert result == 42
```

```
● ● ●  
1 import sys  
2  
3 def main():  
4     name = sys.argv[1]  
5     print(f"Hello, {name}!")  
6  
7 if __name__ == "__main__":  
8     main()
```

```
● ● ●  
1 import subprocess  
2  
3 def test_app():  
4     result = subprocess.run(  
5         ["python", "app.py", "BelleII"],  
6         capture_output=True,  
7         text=True  
8     )  
9  
10    assert result.stdout.strip() == "Hello, BelleII!"  
11    assert result.returncode == 0
```

Test Formats



```
1 def divide(a, b):  
2     """Divide two numbers"""  
3     return a / b
```

```
1 def divide(a, b):  
2     """Divide two numbers"""  
3     return a / b
```

Static Testing is like proofreading
But there are also tools to check structure, syntax

Test Formats

```
1 def divide(a, b):  
2     """Divide two numbers"""  
3     return a / b
```

Dynamic Testing executes the code

```
7 class TestDivide(unittest.TestCase):  
8  
9     def test_divide_success(self):  
10         self.assertEqual(divide(6, 2), 3)
```

Static Testing is like proofreading
But there are also tools to check structure, syntax

Test Formats

```
1 def divide(a, b):  
2     """Divide two numbers"""  
3     return a / b
```

Dynamic Testing executes the code

```
7 class TestDivide(unittest.TestCase):  
8  
9     def test_divide_success(self):  
10         self.assertEqual(divide(6, 2), 3)  
11  
12     def test_divide_by_zero(self):  
13         with self.assertRaises(ZeroDivisionError):  
14             divide(1, 0)
```

Static Testing is like proofreading
But there are also tools to check structure, syntax

Test Formats

```
1 def divide(a, b):  
2     """Divide two numbers"""  
3     return a / b
```

Static Testing is like proofreading
But there are also tools to check structure, syntax

Dynamic Testing executes the code

```
7 class TestDivide(unittest.TestCase):  
8  
9     def test_divide_success(self):  
10         self.assertEqual(divide(6, 2), 3)  
11  
12     def test_divide_by_zero(self):  
13         with self.assertRaises(ZeroDivisionError):  
14             divide(1, 0)
```

```
17 result = divide(10, 2)  
18 print(f"Result: {result}")
```

Passive Testing observes the operation

Tests in basf2 - Unittest

In C++, a popular framework is the so-called Google Test framework

```
1  #include <gtest/gtest.h>
2
3  TEST(HitOrdererTest, OrderNoHits)
4  {
5      HitOrderer orderer;
6
7      std::vector<KDTHit> hits = {};
8      auto result = orderer.orderHits(2.0, 1.0, hits);
9
10     EXPECT_TRUE(result.empty());
11 }
```



Tests in basf2 - Integration

```
7 def run_simulation(output_file):
8     """Main function to be executed if this script"""
9     set_random_seed(12345)
10    main = create_path()
11
12    # specify number of events to be generated
13    main.add_module('EventInfoSetter', expList=[0], evtNumList=[5], runList=[1])
14    main.add_module('Progress')
15    main.add_module('ParticleGun',
16                    pdgCodes=[211],
17                    nTracks=1,
18                    momentumGeneration='fixed',
19                    momentumParams=[1.618034],
20                    phiGeneration='fixed',
21                    phiParams=[27.182818],
22                    thetaGeneration='fixed',
23                    thetaParams=[62.83185])
24    add_simulation(main, bkgfiles=None)
25    main.add_module('RootOutput', outputFileNames=output_file)
26    process(main)
```

```
29 def run_tracking(input_file, mode):
30     set_random_seed(12345)
31     main = create_path()
32     main.add_module('RootInput', inputFileName=input_file)
33
34     main.add_module('Gearbox')
35     main.add_module('Geometry')
36     add_tracking_reconstruction(
37         main,
38         pruneTracks=False,
39         svd_standalone_mode=mode)
40     process(main)
```

Tests in basf2 - Integration

```

7 def run_simulation(output_file):
8     """Main function to be executed if this script"""
9     set_random_seed(12345)
10    main = create_path()
11
12    # specify number of events to be generated
13    main.add_module('EventInfoSetter', expList=[0], evtNumList=[5], runList=[1])
14    main.add_module('Progress')
15    main.add_module('ParticleGun',
16                    pdgCodes=[211],
17                    nTracks=1,
18                    momentumGeneration='fixed',
19                    momentumParams=[1.618034],
20                    phiGeneration='fixed',
21                    phiParams=[27.182818],
22                    thetaGeneration='fixed',
23                    thetaParams=[62.83185])
24    add_simulation(main, bkgfiles=None)
25    main.add_module('RootOutput', outputFileName=output_file)
26    process(main)

```

```

29 def run_tracking(input_file, mode):
30     set_random_seed(12345)
31     main = create_path()
32     main.add_module('RootInput', inputFileName=input_file)
33
34     main.add_module('Gearbox')
35     main.add_module('Geometry')
36     add_tracking_reconstruction(
37         main,
38         pruneTracks=False,
39         svd_standalone_mode=mode)
40     process(main)

```

```

43 if __name__ == "__main__":
44
45     with b2tu.clean_working_directory():
46         dst_file = 'dst.root'
47         b2tu.run_in_subprocess(target=run_simulation, output_file=dst_file)
48         for mode in [
49             'VXDTF2',
50             'SVDHough',
51             'VXDTF2_and_SVDHough',
52             'SVDHough_and_VXDTF2'
53         ]:
54             b2tu.run_in_subprocess(target=run_tracking, input_file=dst_file, mode=mode)
55

```



Tests in basf2 - Passive Testing

```
1 import b2test_utils
2 from basf2 import set_random_seed, create_path, process
3
4 # make logging more reproducible by replacing some strings
5 b2test_utils.configure_logging_for_tests()
6 set_random_seed("1337")
7 testinput = [b2test_utils.require_file('analysis/tests/mdst.root')]
8
9 fsps = ['pi-:all', 'gamma:all']
10
11 # a new ParticleLoader for each fsp
12 testpath = create_path()
13 testpath.add_module('RootInput', inputFileNames=testinput)
14 testpath.add_module('ParticleLoader', decayStrings=fsps)
15 testpath.add_module('ParticleSelector', decayString='gamma:all', cut='isFromECL')
16
17 # Variables created by event kinematics module
18 event_kinematics = [
19     "missingMomentumOfEvent",
20     ...
21 ]
22
23 testpath.add_module('EventKinematics', particleLists=fsps)
24 # Print the variables to log
25 testpath.add_module('ParticlePrinter', listName='', fullPrint=False,
26                     variables=event_kinematics)
27 process(testpath, 1)
28
```



Tests in basf2 - Passive Testing

```

1 import b2test_utils
2 from basf2 import set_random_seed, create_path, process
3
4 # make logging more reproducible by replacing some strings
5 b2test_utils.configure_logging_for_tests()
6 set_random_seed("1337")
7 testinput = [b2test_utils.require_file('analysis/tests/mdst.root')]
8
9 fsps = ['pi-:all', 'gamma:all']
10
11 # a new ParticleLoader for each fsp
12 testpath = create_path()
13 testpath.add_module('RootInput', inputFileNames=testinput)
14 testpath.add_module('ParticleLoader', decayStrings=fsps)
15 testpath.add_module('ParticleSelector', decayString='gamma:all', cut='isFromECL')
16
17 # Variables created by event kinematics module
18 event_kinematics = [
19     "missingMomentumOfEvent",
20     ...
21 ]
22
23 testpath.add_module('EventKinematics', particleLists=fsps)
24 # Print the variables to log
25 testpath.add_module('ParticlePrinter', listName='', fullPrint=False,
26                     variables=event_kinematics)
27 process(testpath, 1)
28

```

```

1 [INFO] Steering file: analysis/tests/event_kinematics_variables.py
2 [INFO] The random number seed is set to "1337"
3 [INFO] Starting event processing, random seed is set to '1337'
4 [INFO] Added file ${BELLE2_SOFTWARE_DIR}/analysis/tests/mdst.root
5 [INFO] ParticleLoader's Summary of Actions:
6 [INFO]   o) creating (anti-)ParticleList with name: pi-:all (pi+:all)
7 [INFO]     -> MDST source: Tracks
8 [INFO]   o) creating ParticleList with name: gamma:all
9 [INFO]     -> MDST source: ECLClusters and KLMClusters
10 [INFO] ParticleSelector: gamma:all
11 [INFO]     -> With cuts : isFromECL
12 [INFO] Conditions Database: found working metadata provider
13     provider = ${BELLE2_CONDB_METADATA}
14 [INFO] Conditions data: configured globaltags
15     (highest priority first) are release-05-01-03, Legacy_CollisionAxisCMS
16 [INFO]   o) missingMomentumOfEvent = 1.09551
17

```

Tests in basf2 - Summary

Belle II / Software / basf2 / Pipelines / #306917

#306917

Passed Created 23 hours ago by **Vidya Sagar Vobbilisetti**, finished 22 hours ago

For commit `77d8faf7` [Fix shadow declaration of evtTOCDC](#)

Related merge request [!5530](#) to merge [bugfix/fix-svd-shadow-declaration](#)

latest merge request 3 jobs 40 minutes 19 seconds, queued for 4 seconds

Pipeline Jobs 3 Tests 1432

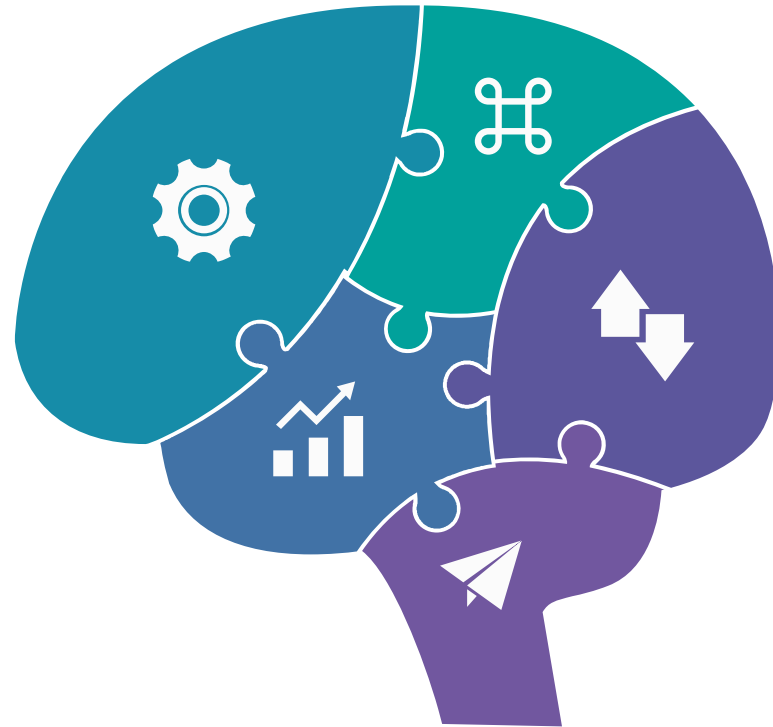
[run-tests](#)

1432 tests 0 failures 0 errors 100% success rate 8m 7s

Tests

Suite	Name	Filename	Status	Duration	Details
tracking	tracking_doxygen_check.py	📄	✓	10m 1s	View details
framework	sphinx_full_release.py	📄	✓	5m 10s	View details
framework	sphinx_light_release.py	📄	✓	3m 18s	View details
hlt	test_phase3_hlt_beam.py	📄	✓	3m 15s	View details

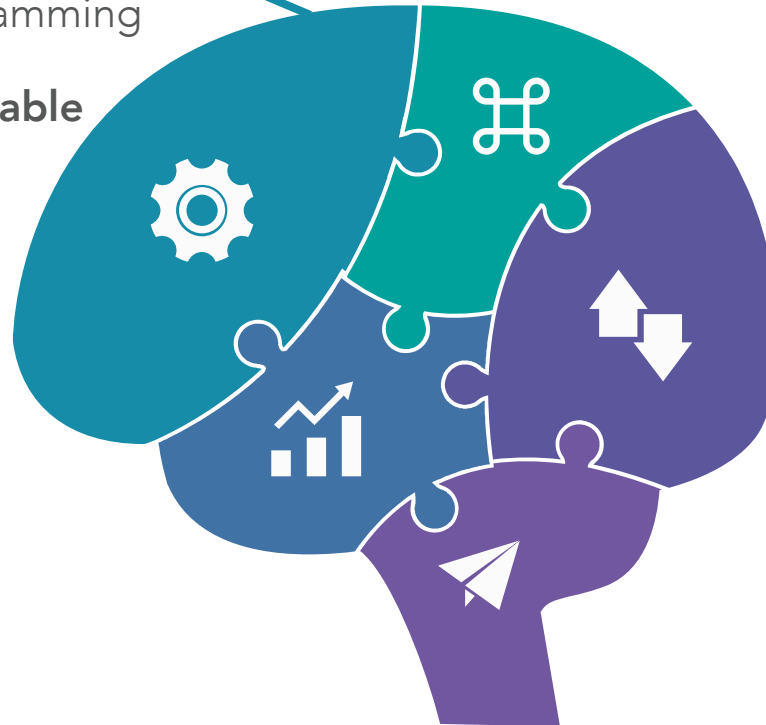
A Few Remarks



A Few Remarks

Functional Programming

- **Object-orientated** and **functional** programming
- Test **small** and independent **units**
- Tests: **independent**, **isolated**, and **repeatable**



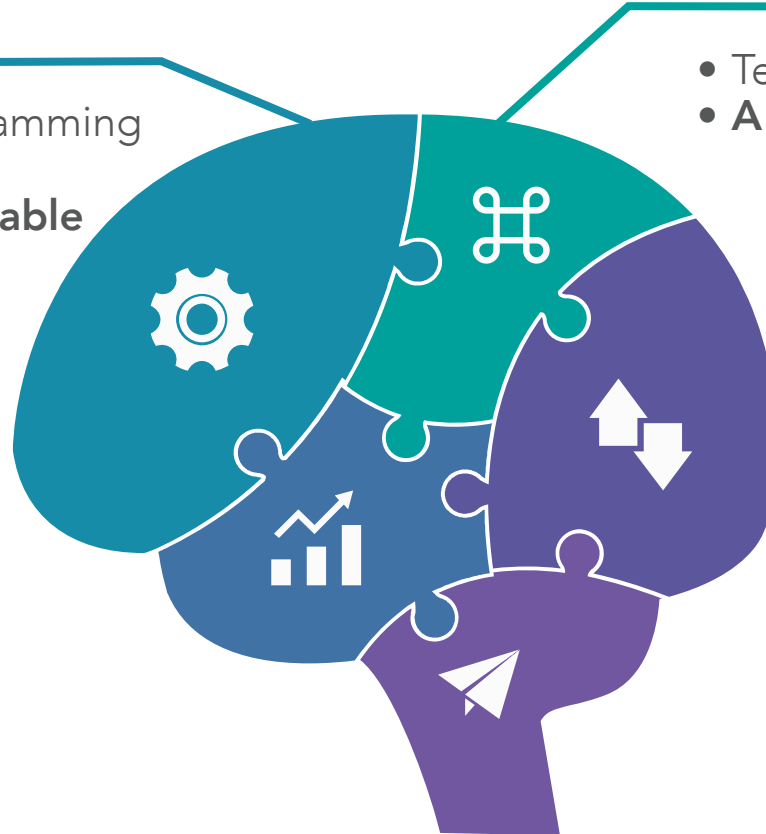
A Few Remarks

Functional Programming

- Object-orientated and functional programming
- Test **small** and independent **units**
- Tests: **independent**, **isolated**, and **repeatable**

"Good" Tests

- Test depicts the **desired functionality**
- A **bad test** can be more harmful than no test



A Few Remarks

Functional Programming

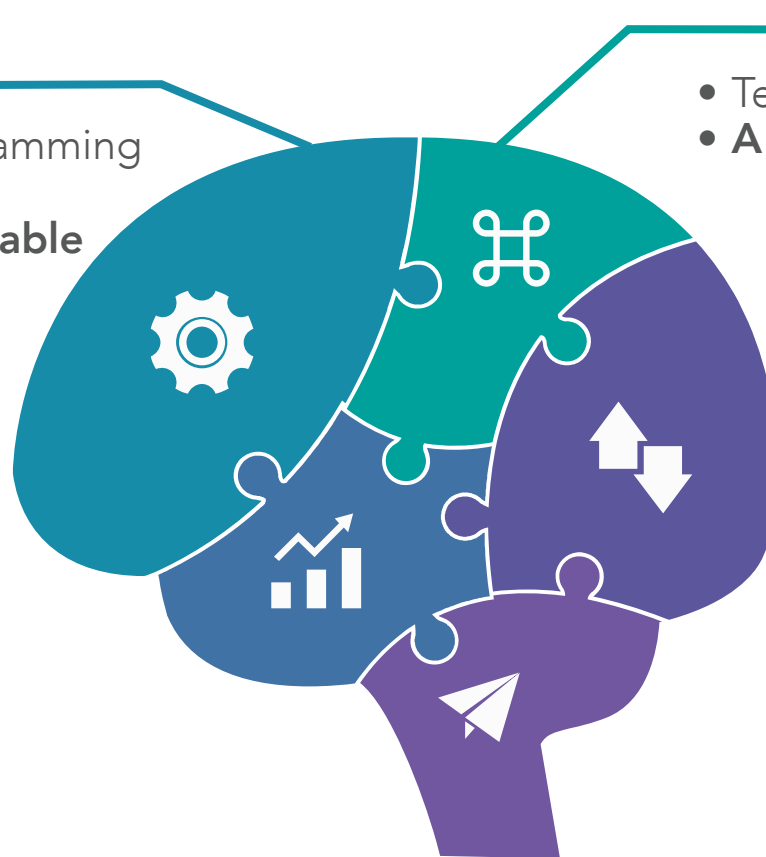
- Object-orientated and functional programming
- Test **small** and independent **units**
- Tests: **independent**, **isolated**, and **repeatable**

"Good" Tests

- Test depicts the **desired functionality**
- A **bad test** can be more harmful than no test

Bottom Up and Top Down

- Two approaches:
 - Write a unit and **test for its functionality**
 - Write a **test first** with your desired behaviour and then the function that passes the test



A Few Remarks

Functional Programming

- Object-orientated and functional programming
- Test **small** and independent **units**
- Tests: **independent**, **isolated**, and **repeatable**

"Good" Tests

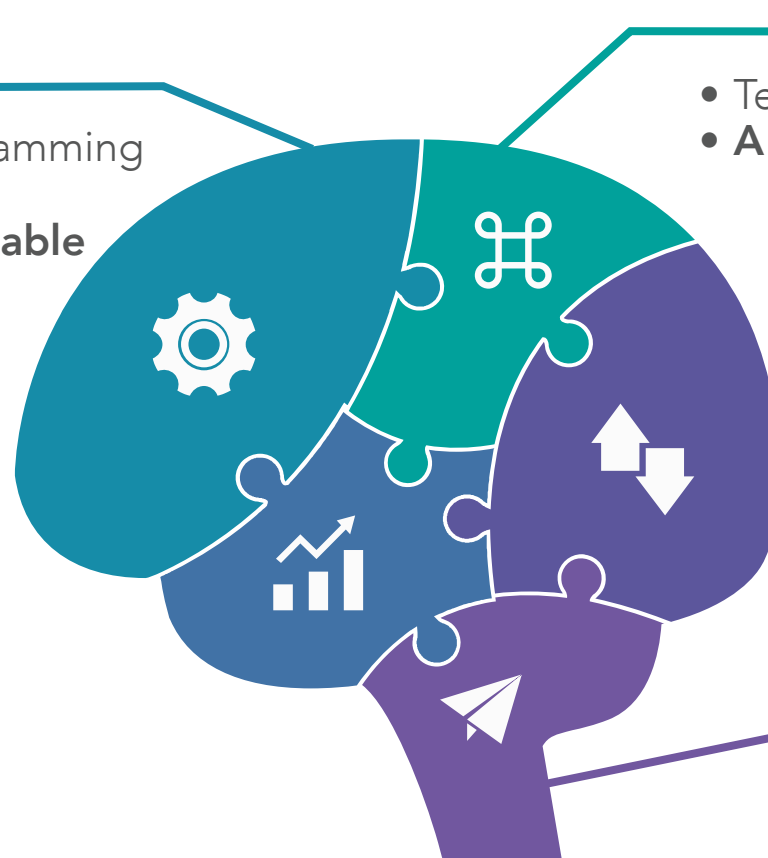
- Test depicts the **desired functionality**
- A **bad test** can be more harmful than no test

Bottom Up and Top Down

- Two approaches:
 - Write a unit and **test for its functionality**
 - Write a **test first** with your desired behaviour and then the function that passes the test

Start Early

- Write tests **early** and **continuously** throughout development
- Let it become an **unconscious habit** while you're writing code



A Few Remarks

Functional Programming

- Object-orientated and functional programming
- Test **small** and independent **units**
- Tests: **independent**, **isolated**, and **repeatable**

"Good" Tests

- Test depicts the **desired functionality**
- A **bad test** can be more harmful than no test

Bottom Up and Top Down

- Two approaches:
 - Write a unit and **test for its functionality**
 - Write a **test first** with your desired behaviour and then the function that passes the test

Modern Tools

- **Ready-to-use libraries** for testing
- **AI** tools to write tests
- **Coverage** tools
- Integrate testing in an **automated workflow** (e.g. GitLab CI/CD)

Start Early

- Write tests **early** and **continuously** throughout development
- Let it become an **unconscious habit** while you're writing code

What is Software Development?



Continuous Integration (CI)

- Frequently merge code into a shared branch
- Each change is automatically built and tested

Continuous Delivery (CD)

- Software is built, tested, and prepared for release continuously
- Changes are delivered in small, frequent increments
- The system is always ready to be released

Continuous Integration (CI)

- Frequently merge code into a shared branch
- Each change is automatically built and tested

Continuous Delivery (CD)

- Software is built, tested, and prepared for release continuously
- Changes are delivered in small, frequent increments
- The system is always ready to be released



Continuous Integration (CI)

- Frequently merge code into a shared branch
- Each change is automatically built and tested

Continuous Delivery (CD)

- Software is built, tested, and prepared for release continuously
- Changes are delivered in small, frequent increments
- The system is always ready to be released

Automate everything that ensures software quality














Framework to automatically start the CI(/CD) process

Pipeline is split into

- Stages - Collection of jobs
- Jobs - Single task

Example: Test stage

- Prepare the environment
- Execute the tests
- Upload the test results

Status	Pipeline	Created by	Stages	Actions
✓ Passed 00:32:28 22 hours ago	#306997 Merge branch 'bugfix/avoid-closing-t... main → 55f4b995 latest branch		✓ ✓ ✓ ✓	
✓ Passed 00:28:07 23 hours ago	#306980 Fix variable name m_HLTAccepted -> ... 5527 → 08ef34b9 latest merge request		✓ ✓ ✓	
✗ Failed 00:03:33 1 day ago	#306968 Declare HLTAccepted as local variable 5527 → f200ba80 merge request		✓ ✗ ➔	 
✓ Passed 00:40:31 1 day ago	#306953 Add test case for calling create_valida... 5532 → 7d4f5187 latest merge request		✓ ✓ ✓	
✓ Passed 00:32:10 1 day ago	#306949 Merge branch 'bugfix/update-bench... main → 06682e43 branch		✓ ✓ ✓ ✓	

Code

Build

Test

Documentation

Integrate



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Automation

GitLab Docs

Search topics, API and CI/CD reference



What's new?

Use GitLab GitLab Duo Extend Install Administer Subscribe Contribute Solutions

Getting started

GitLab Docs / Use GitLab / Use CI/CD to build your application

Use CI/CD to build your application

Use CI/CD to generate your application.

- Tutorials >
- Manage your organization >
- Organize work with projects >
- Plan and track work >
- Manage authentication and authorization >
- Use Git >
- Manage your code >
- Use CI/CD to build your application >

 - Getting started
 - Tutorials >
 - CI/CD YAML syntax reference >
 - Runners >
 - Pipelines >
 - Jobs >
 - CI/CD components >
 - CI/CD inputs >

Getting started

Build and test your application.

CI/CD YAML syntax reference

Pipeline configuration keywords, syntax, examples, and inputs.

Runners

Configuration and job execution.

Pipelines

Configuration, automation, stages, schedules, and efficiency.

Jobs

Configuration, rules, caching, artifacts, and logs.

CI/CD components

Reusable, versioned CI/CD components for pipelines.

CI/CD variables

Configuration, usage, and security.

Pipeline security

Secrets management, job tokens, secure files, and cloud security.

Debugging

Configuration validation, warnings, errors, and troubleshooting.

GitHub Docs

Version: Free, Pro, & Team

Search or ask Copilot



← Home

GitHub Actions

- Get started >
- Concepts >
- How-tos >
- Reference >
- Tutorials >

GitHub Actions documentation

Automate, customize, and execute your software development workflows right in your repository with GitHub Actions. You can discover, create, and share actions to perform any job you'd like, including CI/CD, and combine actions in a completely customized workflow.

Overview

Quickstart

Recommended

Quickstart for GitHub Actions

Try out the core features of GitHub Actions in minutes.

Understanding GitHub Actions

Learn the basics of core concepts and essential terminology in GitHub Actions.

Using GitHub-hosted runners

You can assign a job to run on a virtual machine hosted by GitHub.



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Automation

We will focus on GitLab

But the ideas stay the same



The screenshot shows the GitLab Docs website. The main heading is "Use CI/CD to build your application". Below the heading, there are several cards for different topics: "Getting started", "CI/CD YAML syntax reference", "Runners", "Pipelines", "Jobs", "CI/CD components", "CI/CD variables", "Pipeline security", and "Debugging".

The screenshot shows the GitHub Actions documentation page. The main heading is "GitHub Actions documentation". Below the heading, there is a description of GitHub Actions and two buttons: "Overview" and "Quickstart". At the bottom, there is a "Recommended" section with three cards: "Quickstart for GitHub Actions", "Understanding GitHub Actions", and "Using GitHub-hosted runners".

Hello World!

```
1  stages:
2    - build
3    - test
4
5  build_job:
6    stage: build
7    script:
8      - echo "Building the project"
9
10 test_job:
11   stage: test
12   script:
13     - echo "Running tests"
```

Hello World!

Defines the order of execution
Pipelines run stage by stage

```
1  stages:
2    - build
3    - test
4
5  build_job:
6    stage: build
7    script:
8      - echo "Building the project"
9
10 test_job:
11   stage: test
12   script:
13     - echo "Running tests"
```

Hello World!

Defines the order of execution
Pipelines run stage by stage

Arbitrary names you choose
Each job = one task

```
1  stages:
2    - build
3    - test
4
5  build_job:
6    stage: build
7    script:
8      - echo "Building the project"
9
10 test_job:
11   stage: test
12   script:
13     - echo "Running tests"
```

Hello World!

Defines the order of execution
Pipelines run stage by stage

Arbitrary names you choose
Each job = one task

Assigns a job to a stage
Controls when it runs

```
1  stages:
2    - build
3    - test
4
5  build_job:
6    stage: build
7    script:
8      - echo "Building the project"
9
10 test_job:
11  stage: test
12  script:
13    - echo "Running tests"
```

Hello World!

Defines the order of execution
Pipelines run stage by stage

Arbitrary names you choose
Each job = one task

Assigns a job to a stage
Controls when it runs

The actual commands executed
Runs in a shell on the runner

```
1  stages:
2    - build
3    - test
4
5  build_job:
6    stage: build
7    script:
8      - echo "Building the project"
9
10 test_job:
11   stage: test
12   script:
13     - echo "Running tests"
```

Some More Features

```
1  image: python:3.11
2
3  stages:
4    - build
5    - test
6
7  before_script:
8    - python --version
9
10 build_job:
11   stage: build
12   script:
13     - mkdir build
14     - echo "Build output" > build/result.txt
15   artifacts:
16     paths:
17       - build/
18
19 test_job:
20   stage: test
21   script:
22     - cat build/result.txt
23     - python -m unittest discover
```

Some More Features

Defines the container used to run the jobs

```
1 image: python:3.11
2
3 stages:
4   - build
5   - test
6
7 before_script:
8   - python --version
9
10 build_job:
11   stage: build
12   script:
13     - mkdir build
14     - echo "Build output" > build/result.txt
15   artifacts:
16     paths:
17       - build/
18
19 test_job:
20   stage: test
21   script:
22     - cat build/result.txt
23     - python -m unittest discover
```

Some More Features

Defines the container used to run the jobs

Runs setup commands before each job

```

1  image: python:3.11
2
3  stages:
4    - build
5    - test
6
7  before_script:
8    - python --version
9
10 build_job:
11   stage: build
12   script:
13     - mkdir build
14     - echo "Build output" > build/result.txt
15   artifacts:
16     paths:
17       - build/
18
19 test_job:
20   stage: test
21   script:
22     - cat build/result.txt
23     - python -m unittest discover

```

Some More Features

Defines the container used to run the jobs

Runs setup commands before each job

Stores files from one job so later jobs can use them

```

1  image: python:3.11
2
3  stages:
4    - build
5    - test
6
7  before_script:
8    - python --version
9
10 build_job:
11   stage: build
12   script:
13     - mkdir build
14     - echo "Build output" > build/result.txt
15   artifacts:
16     paths:
17       - build/
18
19 test_job:
20   stage: test
21   script:
22     - cat build/result.txt
23     - python -m unittest discover

```

Even More Features

```

1  image: python:3.11
2
3  stages:
4    - build
5    - test
6
7  variables:
8    OUTPUT_FILE: "build/result.txt"
9
10 before_script:
11   - python --version
12
13 build_job:
14   stage: build
15   script:
16     - mkdir -p build
17     - echo "Build output" > $OUTPUT_FILE
18   artifacts:
19     paths:
20       - build/
21
22 test_job:
23   stage: test
24   needs: [build_job]
25   script:
26     - cat $OUTPUT_FILE
27     - python -m unittest discover
28
29 lint_job:
30   stage: test
31   script:
32     - pip install flake8
33     - flake8 .
34   rules:
35     - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'

```

Even More Features

Define reusable values
Avoid hardcoding paths

```
1 image: python:3.11
2
3 stages:
4   - build
5   - test
6
7 variables:
8   OUTPUT_FILE: "build/result.txt"
9
10 before_script:
11   - python --version
12
13 build_job:
14   stage: build
15   script:
16     - mkdir -p build
17     - echo "Build output" > $OUTPUT_FILE
18   artifacts:
19     paths:
20       - build/
21
22 test_job:
23   stage: test
24   needs: [build_job]
25   script:
26     - cat $OUTPUT_FILE
27     - python -m unittest discover
28
29 lint_job:
30   stage: test
31   script:
32     - pip install flake8
33     - flake8 .
34   rules:
35     - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'
```

Even More Features

Define reusable values
Avoid hardcoding paths

Controls order of the jobs

```

1  image: python:3.11
2
3  stages:
4    - build
5    - test
6
7  variables:
8    OUTPUT_FILE: "build/result.txt"
9
10 before_script:
11   - python --version
12
13 build_job:
14   stage: build
15   script:
16     - mkdir -p build
17     - echo "Build output" > $OUTPUT_FILE
18   artifacts:
19     paths:
20       - build/
21
22 test_job:
23   stage: test
24   needs: [build_job]
25   script:
26     - cat $OUTPUT_FILE
27     - python -m unittest discover
28
29 lint_job:
30   stage: test
31   script:
32     - pip install flake8
33     - flake8 .
34   rules:
35     - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'

```

Even More Features

Define reusable values
Avoid hardcoding paths

Controls order of the jobs

Control when jobs run
More: workflows

```

1  image: python:3.11
2
3  stages:
4    - build
5    - test
6
7  variables:
8    OUTPUT_FILE: "build/result.txt"
9
10 before_script:
11   - python --version
12
13 build_job:
14   stage: build
15   script:
16     - mkdir -p build
17     - echo "Build output" > $OUTPUT_FILE
18   artifacts:
19     paths:
20       - build/
21
22 test_job:
23   stage: test
24   needs: [build_job]
25   script:
26     - cat $OUTPUT_FILE
27     - python -m unittest discover
28
29 lint_job:
30   stage: test
31   script:
32     - pip install flake8
33     - flake8 .
34   rules:
35     - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'

```



One Real Example

```
1 stages:
2   - pre_commit
3   - test
4   - docs
5   - publish
6
7 variables:
8   FLIT_ROOT_INSTALL: "1"
9   PYTHON_VERSION: "3.8.8"
10  ARTIFACTS_TEST_PATH: "junit/test-results.xml"
11  ARTIFACTS_DOCS_PATH: "docs-build"
12  ARTIFACTS_COV_PATH: "coverage/"
13  GITLAB_BELLE2_TOOLS_B2SETUP: "/cvmfs/belle.cern.ch/tools/b2setup"
14  FLIT_INDEX_URL: "https://upload.pypi.org/legacy/"
15  FLIT_USERNAME: __token__
16  BASF2_RELEASE_VERSION: "light-2409-toyger"
17
18 workflow:
19  rules:
20  - if: $CI_PIPELINE_SOURCE == 'merge_request_event'
21  - if: $CI_PIPELINE_SOURCE == 'push' && ($CI_COMMIT_BRANCH == 'main') && $CI_COMMIT_TITLE =~ /Merge branch.*/
22  - if: $CI_COMMIT_TAG
23  - if: $GITLAB_FORCE_RUNNING_PIPELINE == 'yes'
24
25 default:
26  image: belle2/belle2-base-el9:latest
27  tags:
28  - gitlab-runner12
29  before_script:
30  # Preliminary operations with Git
31  - git config user.email "something@something.com"
32  - git config user.name "someone"
33  - git config --global --add safe.directory $(dirname ${GITLAB_BELLE2_TOOLS_B2SETUP})
34  # Setup of basf2
35  - source ${GITLAB_BELLE2_TOOLS_B2SETUP}
36  - echo ${BASF2_RELEASE_VERSION}
37  - b2venv ${BASF2_RELEASE_VERSION}
38  - source venv/bin/activate
39  - pip install flit
40  interruptible: true # All the jobs can be interrupted by newer pipelines
41
42 pre_commit:
43  stage: pre_commit
44  script:
45  - pip install pre-commit
46  - pre-commit run --all-files
```

```
48 test:
49  stage: test
50  script:
51  - flit install -s --deps=develop
52  - echo "Running tests"
53  - pytest -v --cov b2luigi --junitxml=${ARTIFACTS_TEST_PATH} --cov-report=xml --cov-report=html --cov-report=term tests
54  - mv htmlcov ${ARTIFACTS_COV_PATH}
55  coverage: '/(?!total.*? (100(?:\.\d+)?)%|[1-9]?[0-9]?(\.\d+)?)?/'
56  parallel:
57  matrix:
58  - BASF2_RELEASE_VERSION: [$BASF2_PYTHON38_RELEASE, $BASF2_PYTHON311_RELEASE]
59  artifacts:
60  reports:
61  junit: $ARTIFACTS_TEST_PATH
62  coverage_report:
63  coverage_format: cobertura
64  path: coverage.xml
65  paths:
66  - junit/test-results.xml
67  - coverage/
68  expose_as: 'test result and coverage'
69
70 docs:
71  stage: docs
72  needs:
73  - job: test
74  artifacts: false
75  script:
76  - flit install -s --deps=develop
77  - sphinx-build docs/ $ARTIFACTS_DOCS_PATH
78  artifacts:
79  paths:
80  - docs-build/
81  expire_in: 1 week
82  expose_as: 'sphinx documentation'
83  when: on_success
84
85 publish:
86  stage: publish
87  needs:
88  - job: test
89  artifacts: false
90  - job: docs
91  artifacts: false
92  script:
93  - pip install setuptools wheel twine
94  - flit publish
95  only:
96  - tags
```

```
98 prepare_release:
99  stage: publish
100  rules:
101  - if: '$CI_COMMIT_TAG =~ /^v?\d+\.\d+\.\d+$/'
102  script:
103  - yum install -y jq
104  - 'curl --header "JOB-TOKEN: ${CI_JOB_TOKEN}"
105    "${CI_API_V4_URL}/projects/${CI_PROJECT_ID}/repository/
106    changelog?version=${CI_COMMIT_TAG}" | jq -r .notes > release_notes.md'
107  artifacts:
108  paths:
109  - release_notes.md
110
111 publish_release:
112  stage: publish
113  image: registry.gitlab.com/gitlab-org/release-cli:latest
114  before_script: []
115  needs:
116  - job: prepare_release
117  artifacts: true
118  rules:
119  - if: '$CI_COMMIT_TAG =~ /^v?\d+\.\d+\.\d+$/'
120  script:
121  - echo "Creating release"
122  release:
123  name: 'Release $CI_COMMIT_TAG'
124  description: release_notes.md
125  tag_name: '$CI_COMMIT_TAG'
126  ref: '$CI_COMMIT_SHA'
```

How does this look in practice?

Belle II / Software / b2luigi / Merge requests / !163

Use StrEnum

Open Vidya Sagar Vobbilisetti requested to merge `feature/use-StrEnum` into `feature/add-configuring-st...` 2 weeks ago

Overview 9 Commits 2 Pipelines 1 Changes 3

2 ^ v ⋮ + 🔔 Your review

Description

Use StrEnum instead of Enum in SLURM job statuses

Merge Request Commit Message

- Title: Use StrEnum instead of Enum in SLURM job statuses
- Changelog: Changed

👍 0 👎 0 😊

✔ Merge request pipeline #301107 passed ✔✔✔ 📄

Merge request pipeline passed for `ed90e634` 2 weeks ago ?


Test coverage 57.00% from 2 jobs ?

> [View 3 exposed artifacts](#)


8 Approve Approval is optional ?

✔ Test summary: **no** changed test results, **394** total tests Full report

Assignee Edit

 Vidya Sagar Vobbilisetti

Reviewer Edit

 Alexander Heidelberg 🔄

Labels Edit

enhancement x


Milestone Edit

None

Time tracking 🕒 +

No estimate or time spent



4 Participants



How does this look in practice?

 **Merge request pipeline #301107 passed**



Merge request pipeline passed for `ed90e634` 2 weeks ago 
 Test coverage 57.00% from 2 jobs 

▼ Collapse

Artifact	Job
test result and coverage	test: [\$BASF2_PYTHON311_RELEASE]
sphinx documentation	docs
test result and coverage	test: [\$BASF2_PYTHON38_RELEASE]



How does this look in practice?

✓ passed Job #1210489 in pipeline #301107 for ed90e634 from refs/merge-requests/163/head by  Vidya Sagar Vobbilisetti 2 weeks ago

Artifacts

[Download artifacts archive](#)

Name	Size
📁 docs-build	



How does this look in practice?

passed Job #1210489 in pipeline #301107 for ed90e634 from refs/merge-requests/163/head by Vidya Sagar Vobbilisetti 2 weeks ago

Belle II / Software / b2luigi / Jobs / #1210489 / Artifacts

development.html	27.8 KiB	
examples.html	21 KiB	
faq.html	45.9 KiB	
features.html	24.3 KiB	
genindex.html	65.9 KiB	
index.html	36.2 KiB	
installation.html	22.4 KiB	



How does this look in practice?

passed Job #1210489 in pipeline #301107 for ed90e634 from refs/merge-requests/163/head by Vidya Sagar Vobbilisetti 2 weeks ago

Belle II / Software / b2luigi / Jobs / #1210489 / Artifacts

development.html 27.8 KiB

docs-build/index.html 21 KiB

passed Job #1210489 in pipeline #301107 for ed90e634 from refs/merge-requests/163/head by Vidya Sagar Vobbilisetti 2 weeks ago

Artifacts

You are being redirected away from GitLab

This page is hosted on GitLab pages but contains user-generated content and may contain malicious code. Do not accept unless you trust the author and source.

<http://belle2.pages.desy.de/-/software/b2luigi/-/jobs/1210489/artifacts/docs-build/index.html>

installation.html 22.4 KiB



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

How does this look in practice?



Merge request pipeline #301107 passed



Merge request pipeline passed for `ed90e634` 2 weeks ago [?](#)

Test coverage 57.00% from 2 jobs [?](#)



Collapse

Artifact	Job
test result and coverage	test: [\$BASF2_PYTHON311_RELEASE]
sphinx documentation	docs
test result and coverage	test: [\$BASF2_PYTHON38_RELEASE]

How does this look in practice?

#307301

✓ Passed Created 1 hour ago by **Vidya Sagar Vobbiliseti**, finished 35 minutes ago

For commit `8425f0a3`  Remove B2WARNING when no CDC event T0 found to avoid flooding the log

Related merge request `!5530` to merge `bugfix/fix-svd-shadow-declaration`

latest merge request  3 jobs  28 minutes 37 seconds, queued for 3 seconds

Pipeline

Jobs 3

Tests 1432

Group jobs by

Stage

Job dependencies

prepare-pipeline



prepare-pipeline



build-gcc



build-gcc



run-tests



run-tests






LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

How does this look in practice?

Pipeline **Jobs** 3 Tests 1432

Status	Job	Stage	Coverage
✓ Passed ⌚ 00:24:17 📅 37 minutes ago	#1231883: run-tests 🔗 refs/merge-requests/553... -o- 8425f0a3 gitlab-runner2	run-tests	  
✓ Passed ⌚ 00:02:21 📅 1 hour ago	#1231882: build-gcc 🔗 refs/merge-requests/553... -o- 8425f0a3 gitlab-runner2	build-gcc	
✓ Passed ⌚ 00:01:57 📅 1 hour ago	#1231881: prepare-pipeline 🔗 refs/merge-requests/553... -o- 8425f0a3 gitlab-runner2	prepare-pipeline	

How does this look in practice?

Belle II / Software / basf2 / Jobs / #1231881

Search visible log output

^
v
📄
↔
↑
↓
↶

↻

```

26 $ if [ "${CI_PIPELINE_SOURCE}" == "merge_request_event" ]; then # collapsed multi-line command
27 $ unset DISPLAY
28 $ if [ ! -f .release ]; then echo "head" > .release; fi
29 $ rm -rf .git/hooks && ln -sf $(dirname ${GITLAB_BELLE2_TOOLS_B2SETUP})/hooks .git/hooks
30 $ ln -fs site_scons/SConstruct
31 $ source ${GITLAB_BELLE2_TOOLS_B2SETUP}
32 Belle II software tools set up at: /cvmfs/belle.cern.ch/tools
33 Local development directory : /builds/t3_ooTokP/0/belle2/software/basf2
34 $ echo "stage start time -> $(date +%H:%M:%S)"
35 stage start time -> 15:58:16
36 $ if [ "${CI_PIPELINE_SOURCE}" == "merge_request_event" ]; then # collapsed multi-line command
37 The current HEAD of the branch main is 55f4b995e670a799839d887862c4d270fb9c1d37
38 $ echo "stage stop time -> $(date +%H:%M:%S)"
39 stage stop time -> 15:58:16
40 Saving cache for successful job 00:30
41 Creating cache bugfix-fix-svd-shadow-declaration-1-protected...
42 .sconf_temp/: found 14 matching artifact files and directories
43 .sconsign.dblite: found 1 matching artifact files and directories
44 bin/: found 380 matching artifact files and directories
45 build/: found 53076 matching artifact files and directories
46 config.log: found 1 matching artifact files and directories
47 data/: found 441 matching artifact files and directories
48 include/: found 4951 matching artifact files and directories
49 lib/: found 1523 matching artifact files and directories
50 modules/: found 955 matching artifact files and directories
51 target_head_hash.txt: found 1 matching artifact files and directories
52 No URL provided, cache will not be uploaded to shared cache server. Cache will be stored only locally.
53 Created cache
54 Cleaning up project directory and file based variables 00:01
55 Job succeeded

```

Duration: 1 minute 57 seconds
Finished: 1 hour ago
Queued: 2 seconds
Timeout: 1h (from project) ⓘ
Runner: #2478 (ooTokPDA) DESY b2-gitlab-runner2 for belle2/software/basf2
Source: Merge Request
Tags: gitlab-runner2

Commit 8425f0a3 🔗 in !5530
Remove B2WARNING when no CDC event T0 found to avoid flooding the log

Pipeline #307301 ✔ Passed for !5530
with bugfix/fix-svd-shadow-declaration 🔗

prepare-pipeline
v

Related jobs

→ ✔ prepare-pipeline

How does this look in practice?

Belle II / Software / basf2 / Jobs / #1230947

Showing last 499.92 KiB of log. [View raw or full log.](#) ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ ⏸

```

k)
10353 Skipped tests: b2bii/tests/b2bii_cppcheck.py (Test skipped due to the update of Cppcheck)
10354 Skipped tests: validation/tests/test_full_stack.py (The test is problematic when run during our C
I/CD pipelines)
10355 Skipped tests: display/tests/display_cppcheck.py (Test skipped due to the update of Cppcheck)
10356 Skipped tests: ir/tests/ir_cppcheck.py (Test skipped due to the update of Cppcheck)
10357 FAILED tests: analysis/tests/variablemanager_examples.py (exited with return code 1)
10358 FAILED tests: analysis/tests/memcheck.py (exited with return code 1)
10359 FAILED tests: analysis/tests/tutorials_B2A9XX.py (exited with return code 1)
10360 FAILED tests: analysis/tests/fei_examples.py (exited with return code 1)
10361 FAILED tests: analysis/tests/tutorials_B2A8XX.py (exited with return code 1)
10362 FAILED tests: analysis/tests/tutorials_B2A3XX.py (exited with return code 1)
10363 FAILED tests: analysis/tests/graFEI_examples.py (exited with return code 1)
10364 FAILED tests: analysis/tests/postmdstidentification_examples.py (exited with return code 1)
10365 FAILED tests: online_book/tests/steering_files.py (exited with return code 1)
10366 FAILED tests: skim/tests/test_skims.py (exited with return code 1)
10367 10 out of 312 tests failed, 32 were skipped
10368 Uploading artifacts for failed job 00:05
10369 Uploading artifacts...
10370 build_doc/html/: found 1069 matching artifact files and directories
10371 Uploading artifacts as "archive" to coordinator... 201 Created id=1230947 responseStatus=201 Created
token=glcvt-64
10372 Uploading artifacts...
10373 output_tests_reports/tests-google.xml: found 1 matching artifact files and directories
10374 output_tests_reports/tests-framework.xml: found 1 matching artifact files and directories
10375 output_tests_reports/tests-non-framework.xml: found 1 matching artifact files and directories
10376 Uploading artifacts as "junit" to coordinator... 201 Created id=1230947 responseStatus=201 Created to
ken=glcvt-64
10377 Cleaning up project directory and file based variables 00:01
10378 ERROR: Job failed: exit code 1

```

Duration: 24 minutes 31 seconds

Finished: 1 day ago

Queued: 8 seconds

Timeout: 1h (from project) [?](#)

Runner: #2469 (qtNm_pvp) DESY b2-
gitlab-runner1 for
belle2/software/basf2

Source: Push

Test summary: 10 of 1432 failed

Tags: gitlab-runner1

Job artifacts [?](#)

The artifacts will be removed in 4 weeks [?](#)

Download

Browse

Commit 8ecbd44c [?](#)

Merge branch 'bugfix/remove-ClassIm
p' into 'main'

Pipeline #306948 ✖ Failed for relea
se/11-00 [?](#)

run-tests ⏵

Related jobs



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

How does this look in practice?

Belle II / Software / basf2 / Pipelines / #307301

#307301

✓ Passed Created 1 hour ago by **Vidya Sagar Vobbilisetti**, finished 40 minutes ago

For commit `8425f0a3` [Remove B2WARNING when no CDC event T0 found to avoid flooding the log](#)

Related merge request [15530](#) to merge bugfix/fix-svd-shadow-declaration

latest merge request [3 jobs](#) ⌚ 28 minutes 37 seconds, queued for 3 seconds

Pipeline **Jobs** 3 **Tests** 1432

Summary

1432 tests 0 failures 0 errors 100% success rate 8m 10s

Jobs

Job	Execution time	Failed	Errors	Skipped	Passed	Total
run-tests	8m 10s	0	0	37	1395	1432













How does this look in practice?

Pipeline Jobs **3** Tests **1432**

[run-tests](#)

1432 tests 0 failures 0 errors 100% success rate 8m 10s

Tests

Suite	Name	Filename	Status	Duration	Details
tracking	tracking_doxygen_check.py			9m 48s	View details
framework	sphinx_full_release.py			5m 15s	View details
framework	sphinx_light_release.py			3m 24s	View details
hlt	test_phase3_hlt_beam.py			3m 15s	View details
daq/hbasf2	test_histogram.py			3m 7s	View details
analysis	code-quality-doxygen.py			3m 5s	View details

What is Software Development?





LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Why Documentation?

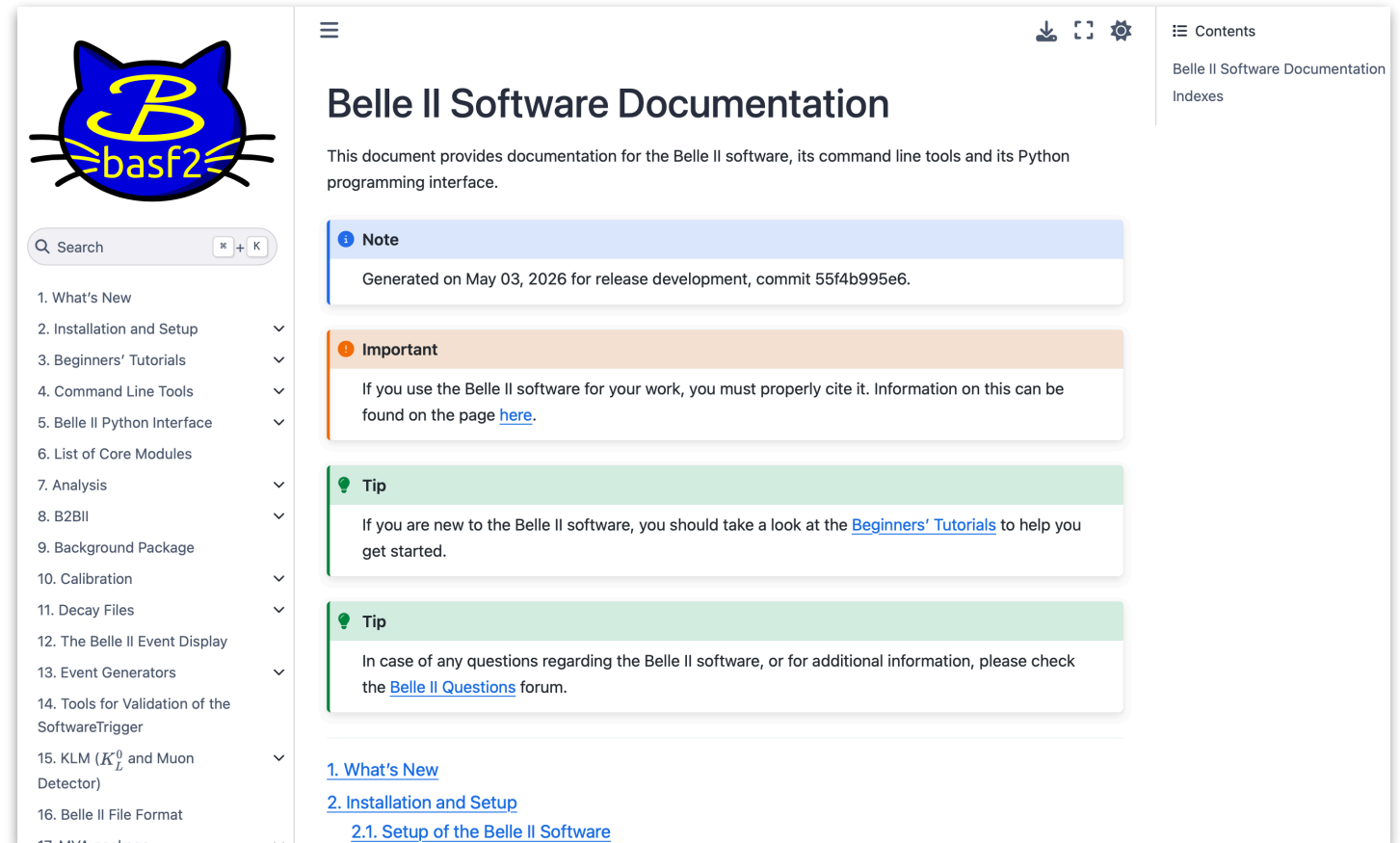


Why Documentation?

If it is not documented, it does not exist

Why Documentation?

If it is not documented, it does not exist



Belle II Software Documentation

This document provides documentation for the Belle II software, its command line tools and its Python programming interface.

Note
Generated on May 03, 2026 for release development, commit 55f4b995e6.

Important
If you use the Belle II software for your work, you must properly cite it. Information on this can be found on the page [here](#).

Tip
If you are new to the Belle II software, you should take a look at the [Beginners' Tutorials](#) to help you get started.

Tip
In case of any questions regarding the Belle II software, or for additional information, please check the [Belle II Questions](#) forum.

[1. What's New](#)
[2. Installation and Setup](#)
[2.1. Setup of the Belle II Software](#)

Contents
Belle II Software Documentation
Indexes

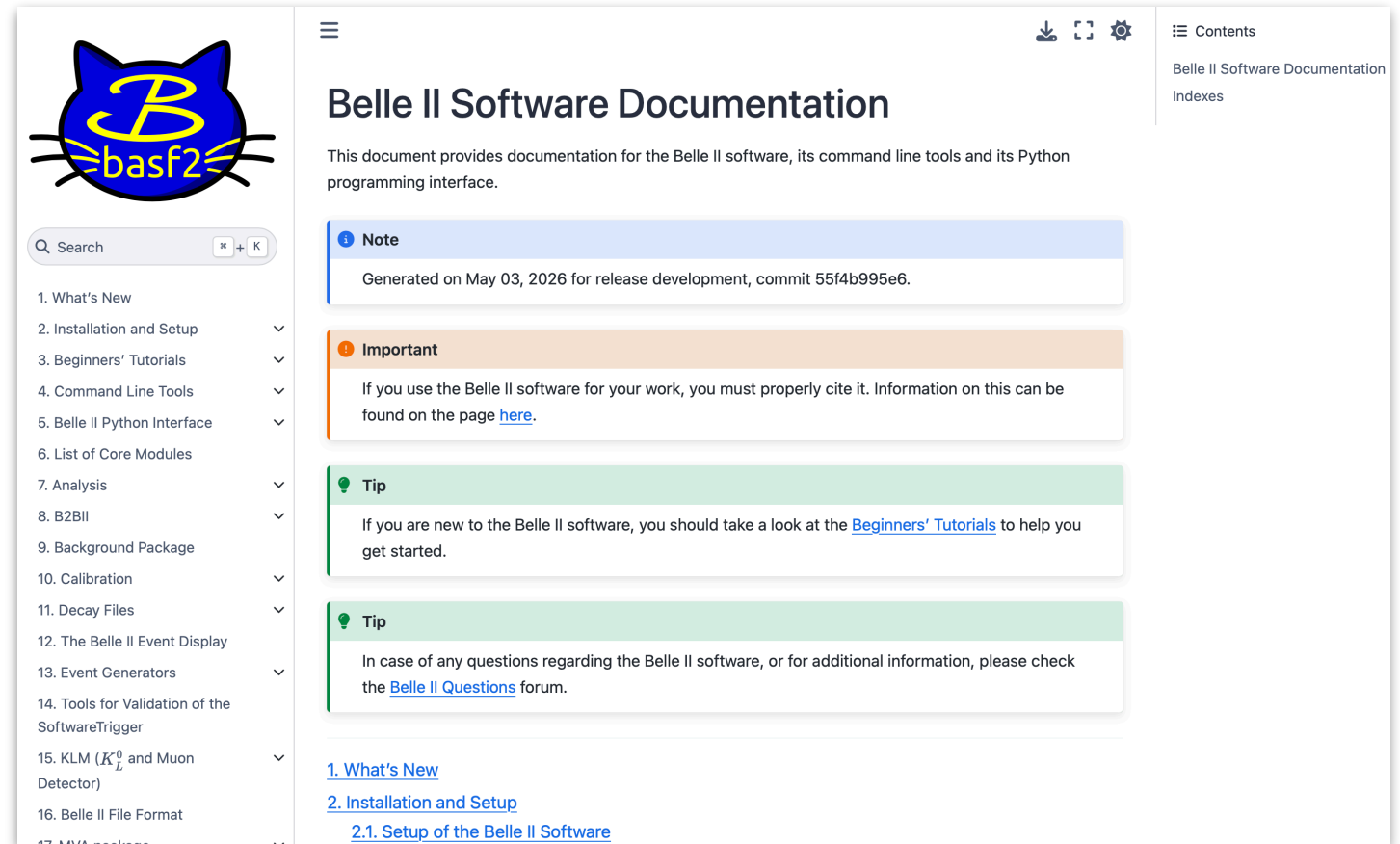
- 1. What's New
- 2. Installation and Setup
- 3. Beginners' Tutorials
- 4. Command Line Tools
- 5. Belle II Python Interface
- 6. List of Core Modules
- 7. Analysis
- 8. B2BI
- 9. Background Package
- 10. Calibration
- 11. Decay Files
- 12. The Belle II Event Display
- 13. Event Generators
- 14. Tools for Validation of the SoftwareTrigger
- 15. KLM (K_L^0 and Muon Detector)
- 16. Belle II File Format
- 17. MVA package

Why Documentation?

If it is not documented, it does not exist

What is documented?

- Code
- Examples
- Structures
- Educational Kits



Belle II Software Documentation

This document provides documentation for the Belle II software, its command line tools and its Python programming interface.

Note
Generated on May 03, 2026 for release development, commit 55f4b995e6.

Important
If you use the Belle II software for your work, you must properly cite it. Information on this can be found on the page [here](#).

Tip
If you are new to the Belle II software, you should take a look at the [Beginners' Tutorials](#) to help you get started.

Tip
In case of any questions regarding the Belle II software, or for additional information, please check the [Belle II Questions](#) forum.

[1. What's New](#)
[2. Installation and Setup](#)
[2.1. Setup of the Belle II Software](#)

Why Documentation?

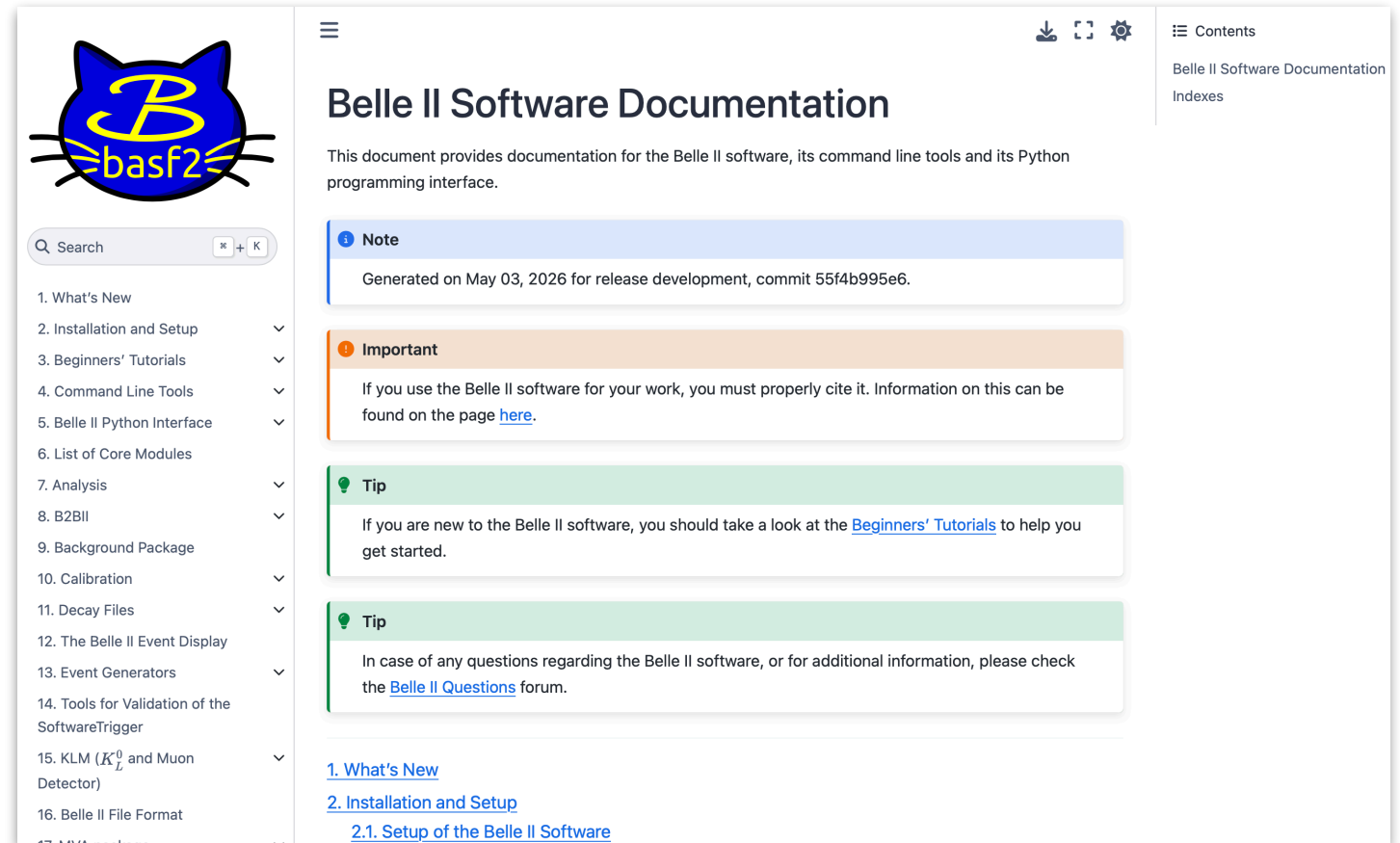
If it is not documented, it does not exist

What is documented?

- Code
- Examples
- Structures
- Educational Kits

What should be documented?

- What it does
- How to use it
- How it works



Belle II Software Documentation

This document provides documentation for the Belle II software, its command line tools and its Python programming interface.

Note
Generated on May 03, 2026 for release development, commit 55f4b995e6.

Important
If you use the Belle II software for your work, you must properly cite it. Information on this can be found on the page [here](#).

Tip
If you are new to the Belle II software, you should take a look at the [Beginners' Tutorials](#) to help you get started.

Tip
In case of any questions regarding the Belle II software, or for additional information, please check the [Belle II Questions](#) forum.

[1. What's New](#)
[2. Installation and Setup](#)
[2.1. Setup of the Belle II Software](#)

Contents
Belle II Software Documentation
Indexes

A tool that turns text + code into structured documentation websites

How does Sphinx work?

- Inline or separate files in reStructuredText or Markdown are collected and compiled into an output

Why Sphinx?

- Easy to learn (once setup is understood)
- Automatic API documentation
- Cross-referencing is easy
- Many different extensions



Sphinx

Create intelligent and beautiful documentation with ease

Rich Text Formatting

Author in [reStructuredText](#) or [MyST Markdown](#) to create highly structured technical documents, including tables, highlighted code blocks, mathematical notations, and more.

Powerful Cross-Referencing

Create [cross-references](#) within your project, and even across [different projects](#). Include references to sections, figures, tables, citations, glossaries, code objects, and more.

Versatile Documentation Formats

Generate documentation in the preferred formats of your audience, including HTML, LaTeX (for PDF), ePub, Texinfo, [and more](#).

Extensive Theme Support

Create visually appealing documentation, with a wide choice of [built-in](#) and [third-party](#) HTML themes and the ability to customize or [create new themes](#).

Fully Extensible

Add custom functionality, via robust [extension mechanisms](#) with numerous [built-in](#) and [third-party](#) extensions available for tasks like creating diagrams, testing code, and more.

Automatic API Documentation

Generate API documentation for Python, C++ and other [software domains](#), manually or [automatically from docstrings](#), ensuring your code documentation stays up-to-date with minimal effort.

Internationalization (i18n)

Add documentation [translations](#) multiple languages to reach a global audience.

Active Community and Support

Benefit from an [active community](#), with numerous resources, tutorials, forums, and examples.

RootInput

Reads objects/arrays from one or more .root files saved by the RootOutput module and makes them available through the DataStore. Files do not necessarily have to be local, <http://> and root:// (for files in xrootd) URLs are supported as well.

Package:

framework

Library:

librootio.so

Parameters:

- **branchNames** (*list(str), default=[]*)
Names of event durability branches to be read. Empty means all branches. (EventMetaData is always read)
- **branchNamesPersistent** (*list(str), default=[]*)
Names of persistent durability branches to be read. Empty means all branches. (FileMetaData is always read)
- **cacheSize** (*int, default=0*)
file cache size in Mbytes. If negative, use root default

RootInput

Reads objects/arrays from one or more .root files saved by the RootOutput module and makes them available through the DataStore. Files do not necessarily have to be local, <http://> and root:// (for files in xrootd) URLs are supported as well.

Package:

framework

Library:

librootio.so

Parameters:

- **branchNames** (*list(str), default=[]*)
Names of event durability branches to be read. Empty means all branches. (EventMetaData is always read)
- **branchNamesPersistent** (*list(str), default=[]*)
Names of persistent durability branches to be read. Empty means all branches. (FileMetaData is always read)
- **cacheSize** (*int, default=0*)
file cache size in Mbytes. If negative, use root default

```

39 RootInputModule::RootInputModule() : Module(), m_nextEntry(0), m_lastPersistentEntry(-1), m_tree(nullptr), m_persistent(nullptr)
40 {
41     //Set module properties
42     setDescription("Reads objects/arrays from one or more .root files saved by the RootOutput
43         module and makes them available through the DataStore. Files do not necessarily have
44         to be local, http:// and root:// (for files in xrootd) URLs are supported as well.");
45     setPropertyFlags(c_Input);
46
47     //Parameter definition
48     vector<string> emptyvector;
49     addParam("cacheSize", m_cacheSize,
50         "file cache size in Mbytes. If negative, use root default", 0);
51     addParam("inputFileName", m_inputFileName,
52         "Input file name. For multiple files, use inputFileNames or wildcards instead.
53         Can be overridden using the -i argument to basf2.",
54         string(""));

```

Belle II Software development

Main Page | Topics | Namespaces | Classes | Files | Search

EventLimiterModule
InteractiveModule
IoVDependentConditionMo
PartialSelectModule
PrescaleModule
PrintBeamParametersModu
PrintCollectionsModule
ProgressBarModule
ProgressModule
PruneDataStoreModule
RandomBarrierModule
TheKillerModule
GearboxModule
HistoManagerModule
ProfileModule
StatisticsSummaryModule
RootInputModule
RootOutputModule
SeqRootInputModule
SeqRootOutputModule
PyObjConvUtils
_RelationsInterfaceImpl
_StoreArrayImpl
RootIOUtilities
TangoPalette
HTML
IOIntercept
Stream
TestHelpers
Utils
CreateConsistencyInfoModul
DataFlowVisualization

RootInputModule Class Reference
Packages » framework | Modules » framework modules

Classes | Public Types | Public Member Functions | Static Public Member Functions | Protected Member Functions | Private Types | Private Member Functions | Private Attributes | List of all members

Module to read TTree data from file into the data store. More...

```
#include <RootInputModule.h>
```

Inheritance diagram for RootInputModule:

```

graph BT
    RootInputModule --> Module
    Module --> PathElement
  
```

Classes

struct ReadStats
for collecting statistics over multiple files. More...


Public Types

```
enum EModulePropFlags {
    c_Input = 1 ,
    c_Output = 2 ,
    c_ParallelProcessingCertified = 4 ,
    c_HistogramManager = 8 ,
    c_InternalSerializer = 16 ,
    c_TerminateInAllProcesses = 32 ,
    c_DontCollectStatistics = 64
}
```

Each module can be tagged with property flags, which indicate certain features of the module. More...

Belle2 | RootInputModule | Generated on Sun May 3 2026 03:36:29 for Belle II Software by doxygen 1.13.2

b2luigi - API documentation



and you will have all the functionality of `luigi` and `b2luigi` without the need to change anything:

```
class b2luigi.Task(*args, **kwargs) \[source\]
```

Bases: `Task`

Drop in replacement for `luigi.Task` which is 100% API compatible. It just adds some useful methods for handling output file name generation using the parameters of the task. See [Quick Start](#) on information on how to use the methods.

Also, we change the default value of `max_batch_size` to 1, so that you have to explicitly set it to a value greater than 1 to enable parameter batching.

Example

```
class MyAverageTask(b2luigi.Task):
    def requires(self):
        for i in range(100):
            yield self.clone(MyNumberTask, some_parameter=i)

    def output(self):
        yield self.add_to_output("average.txt")

    def run(self):
        # Build the mean
        summed_numbers = 0
        counter = 0
        for input_file in self.get_input_file_names("output_file.txt"):
            with open(input_file, "r") as f:
                summed_numbers += float(f.read())
                counter += 1

        average = summed_numbers / counter

        with self.get_output_file("average.txt").open("w") as f:
            f.write(f"{average}\n")
```

```

1 class Task(luigi.Task):
2     """
3     Drop in replacement for ``luigi.Task`` which is 100% API compatible.
4     It just adds some useful methods for handling output file name generation using
5     the parameters of the task.
6     See :ref:`quick-start-label` on information on how to use the methods.
7
8     Also, we change the default value of ``max_batch_size`` to 1, so that you have
9     to explicitly set it to a value greater than 1 to enable parameter batching.
10
11     Example:
12
13     .. code-block:: python
14
15         class MyAverageTask(b2luigi.Task):
16             def requires(self):
17                 for i in range(100):
18                     yield self.clone(MyNumberTask, some_parameter=i)
19
20             def output(self):
21                 yield self.add_to_output("average.txt")
22
23             def run(self):
24                 # Build the mean
25                 summed_numbers = 0
26                 counter = 0
27                 for input_file in self.get_input_file_names("output_file.txt"):
28                     with open(input_file, "r") as f:
29                         summed_numbers += float(f.read())
30                         counter += 1
31
32                 average = summed_numbers / counter
33
34                 with self.get_output_file("average.txt").open("w") as f:
35                     f.write(f"{average}\n")
36
37     """

```



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

basf2 - Online Book



Q Search + K

- 1. What's New
- 2. Installation and Setup
- 3. Beginners' Tutorials**
- 3.1. Welcome!
- 3.2. Fundamentals
- 3.3. Software Prerequisites
- 3.4. Working with Belle II software.
- 3.5. Offline analysis
- 3.6. Data model and computing
- 3.7. Workflow Management
- 3.8. Join us



3. Beginners' Tutorials

This online textbook aims to help new Belle II members to get started with the software by following through a series of hands-on lessons.

If you need to cite this online book, please use the following citation:

Belle II Collaboration, The Belle II Software Online Book. [Online]. Available: <your_url_for_this_page>.

Tip

Just as there are many versions of the Belle II software, there are many versions of this documentation to match it. After all, if a new feature is added in our software, we also want to have the documentation for it. You can change your version by clicking on [other versions](#).

If you are a new to all of this, we recommend you to select the *recommended release version* (`release-xx-xx-xx (recommended)` in the above list).

You can also take a sneak peek at the most recent version of the documentation by selecting the [development version](#). However not all of the code examples might work for you yet.

The earliest release version which contains this online book is `release-05-01-12`.

Warning



Belle II / Software / basf2 / Repository

Files main basf2 / online_book / index-01-online_book.rst

Search files (*.vue, *.rb...)

- online_book
 - analysis
 - awesome
 - basf2
 - computing
 - fundamentals
 - prerequisites
 - tests
 - welcome
 - collaborative_tools
 - collaborative_tools.rst
 - workflowmanagement
 - .librarians
 - analysis.rst
 - basf2.rst
 - cat_hat_angry_point.jpg
 - computing.rst
 - contribute.rst

index-01-online_book.rst 1 Open Find file Blame Edit

initial commit
Priyanka Cheema authored 18 Feb 2025 ab79987f History

index-01-online_book.rst 2.29 KIB Code Preview Copy Download

```

1 .. _onlinebook:
2
3 =====
4 Beginners' Tutorials
5 =====
6
7 This online textbook aims to help new Belle II members to get started with
8 the software by following through a series of hands-on lessons.
9
10 If you need to cite this online book, please use the following citation:
11 Belle II Collaboration, The Belle II Software Online Book. [Online].
12 Available: <your_url_for_this_page>.
13
14 .. tip::
15
16 Just as there are many versions of the Belle II software, there are many
17 versions of this documentation to match it. After all, if a new feature is
18 added in our software, we also want to have the documentation for it. You can
19 change your version by clicking on `other versions <https://software.belle2.org/>`.
20
21 If you are a new to all of this, we recommend you to select the
22 *recommended release version* (`release-xx-xx-xx (recommended)` in the
23 above list).
24

```

Provide feedback



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

basf2 - Packages

- 8. B2BII
- 9. Background Package
- 10. Calibration
- 11. Decay Files
- 12. The Belle II Event Display
- 13. Event Generators
- 14. Tools for Validation of the SoftwareTrigger
- 15. KLM (K^L and Muon Detector)
- 16. Belle II File Format
- 17. MVA package
- ONNX tips and tricks
- 18. PXD
- 19. Reconstruction
- 20. Simulation
- 21. Skims
- 22. SVD
- 23. Tracking
- 24. TRG
- 25. Tools for Physics Validation of the Software
- 26. Software development
- 27. How to document your code with Sphinx
- 28. Software publications

17. MVA package

Multivariate Analysis (MVA) generally makes use of statistical methods that deal with multiple, potentially correlated variables. In particle physics the term is mostly used interchangeably with machine learning (ML) techniques. One of the most popular applications are classifiers that distinguish a signal from some background. This is often implemented using Boosted Decision Trees (BDTs) or Neural Networks (NN).

The MVA package helps to integrate these methods with `basf2`.

Tip

For a hands-on introductory lesson with the MVA package, see [Continuum suppression using Boosted Decision Trees](#).

17.1. Overview

```

graph TD
    DB[Belle II Database] -- "loads and stores weightfiles" --> MVA[mva package]
    MVA -- "uses" --> B[Backends]
    MVA -- "provides" --> B
    subgraph Backends
        B1[FastBDT]
        B2[ONNX]
        B3[TMVA]
        B4[FANN]
        B5[Python]
        B6[Neurobayes]
    end
  
```

Belle II / Software / basf2 / Repository

Files

Search files (*.vue, *.rb...)

- > ir
- > klm
- > masterclass
- > mdst
- > mva
 - > dataobjects
 - > doc
 - > figs
 - > whatsnew-since
 - > index-01-mva.rst
 - > onnx-tips.rst
 - > examples
 - > interface
 - > methods
 - > modules
 - > scripts
 - > tests

main basf2 / mva / doc / index-01-mva.rst

index-01-mva.rst Find file Blame Edit

Remove any usage of/reference to Theano from basf2
Giacomo De Pietro authored 2 months ago 9b5e7974 History

index-01-mva.rst 25.11 KiB Code Preview

```

1 .. _mva:
2
3 MVA package
4 =====
5
6 Multivariate Analysis (MVA) generally makes use of statistical methods that deal
7 with multiple, potentially correlated variables. In particle physics the term is
8 mostly used interchangeably with machine learning (ML) techniques. One of the
9 most popular applications are classifiers that distinguish a signal from some
10 background. This is often implemented using Boosted Decision Trees (BDTs) or
11 Neural Networks (NN).
12
13 The MVA package helps to integrate these methods with `basf2`.
14
15 .. tip:: For a hands-on introductory lesson with the MVA package, see :ref:`onLine_book_cs_bdt`.
16
17 Overview
18 -----
19
20 .. _overview_mva_package:
21
22 .. figure:: figs/overview_mva_package.svg
23    :scale: 200 %
24
  
```



LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Bonus: b2luigi - Starter Kit

Quick Start

b2luigi Starter Kit

A Simple **b2luigi** Task

Building a Dependency Graph
with **b2luigi**

b2luigi.WrapperTask and
b2luigi Settings

Multiple Tasks and Introduction to
Schedulers

Introduction to **Basf2PathTask**

The **requires** decorator

Writing a **basf2** analysis
steering file

Scaling up your analysis

Submitting a task to the LSF
batch system

Submitting a task to the
HTCondor batch system

gbasf2 Analysis Task

Common Analysis Workflow

Common Analysis Workflow
Continued

b2luigi Features

Examples

API Documentation

FAQ



b2luigi Starter Kit

This tutorial was developed as a workshop for the October 2024 Belle II General Meeting. The idea of this tutorial is that each of the following examples is present in your current working directory and is executed by hand. To get the example files...

1. ... either copy the Python code directly from this page into a blank file or...
2. ... download the examples directly from their respective pages or...
3. ... clone the **b2luigi** repository and move to the **examples** directory, e.g.

```
git clone https://gitlab.desy.de/belle2/software/b2luigi.git
cd b2luigi/examples
```

Alternatively, you can also clone directly from [GitHub](#).

In any of these cases, it is recommended to run the tutorial in a virtual environment. With access to the Belle II software stack, you can use the following commands to set up a virtual environment and install the necessary packages:

```
source /cvmfs/belle.cern.ch/tools/b2setup
b2venv release-09-00-00
```

Hint

Use a full **basf2** release if you want to run the **basf2** examples with the reconstruction.

Belle II / Software / b2luigi

Files

main b2luigi / examples / tutorial / README.rst

Search files (*.vue, *.rb...)

README.rst

Find file Blame Edit

Minimal format change
Alexander Heidelberg authored 9 Apr 2025

472a0237 History

README.rst 1.96 KIB

Code Preview

```
1 .. _starterkit_label:
2
3 ``b2luigi`` Starter Kit
4 =====
5
6 This tutorial was developed as a workshop for the October 2024 Belle II General Meeting.
7 The idea of this tutorial is that each of the following examples is present in your current workin
8 To get the example files...
9
10 1. ... either copy the Python code directly from this page into a blank file or...
11 2. ... download the examples directly from their respective pages or...
12 3. ... clone the ``b2luigi`` repository and move to the ``examples`` directory, e.g.
13
14 .. code-block:: bash
15
16     git clone https://gitlab.desy.de/belle2/software/b2luigi.git
17     cd b2luigi/examples
18
19 Alternatively, you can also clone directly from `GitHub <https://github.com/belle2/b2luigi>`.
20
21 In any of these cases, it is recommended to run the tutorial in a virtual environment.
22 With access to the Belle II software stack, you can use the following commands to set up a virtual
23
24 .. code-block:: bash
```

Provide feedback

Multiple Tasks and Introduction to Schedulers

Hint

This example demonstrates how a single task can require multiple tasks to complete. This is useful if the output of multiple tasks is needed to compute the output of the following task. Additionally, we introduce the core concepts of schedulers.

`b2luigi WrapperTask` is not the only way to require multiple tasks. Also normal tasks can require multiple tasks. This is useful if the wrapper task uses the inputs of the required tasks to compute its output.

For this, we define the first and second task in the same way as in the previous example.

```
import b2luigi

class MyTask(b2luigi.Task):
    parameter = b2luigi.IntParameter()

    def run(self):
        with open(self.get_output_file_name("output.txt"), "w") as f:
            f.write(f"{self.parameter}")

    def output(self):
        yield self.add_to_output("output.txt")

class MyOtherTask(b2luigi.Task):
    parameter = b2luigi.IntParameter()

    def requires(self):
        return MyTask(parameter=self.parameter)
```

```
1 """
2 .. _exercise04_label:
3
4 Multiple Tasks and Introduction to Schedulers
5 =====
6
7 .. hint::
8     This example demonstrates how a single task can require multiple tasks to complete.
9     This is useful if the output of multiple tasks is needed to compute the output of the following task.
10    Additionally, we introduce the core concepts of schedulers.
11
12 :class:`b2luigi WrapperTask` is not the only way to require multiple tasks.
13 Also normal tasks can require multiple tasks.
14 This is useful if the wrapper task uses the inputs of the required tasks to compute its output.
15
16 For this, we define the first and second task in the same way as in the previous example.
17 """
18 import b2luigi
19
20
21 class MyTask(b2luigi.Task):
22     parameter = b2luigi.IntParameter()
23
24     def run(self):
25         with open(self.get_output_file_name("output.txt"), "w") as f:
26             f.write(f"{self.parameter}")
27
28     def output(self):
29         yield self.add_to_output("output.txt")
30
31
32 class MyOtherTask(b2luigi.Task):
33     parameter = b2luigi.IntParameter()
34
35     def requires(self):
36         return MyTask(parameter=self.parameter)
```



**HELP
WANTED**

NEEDED