Python for Accelerator Controls at CERN

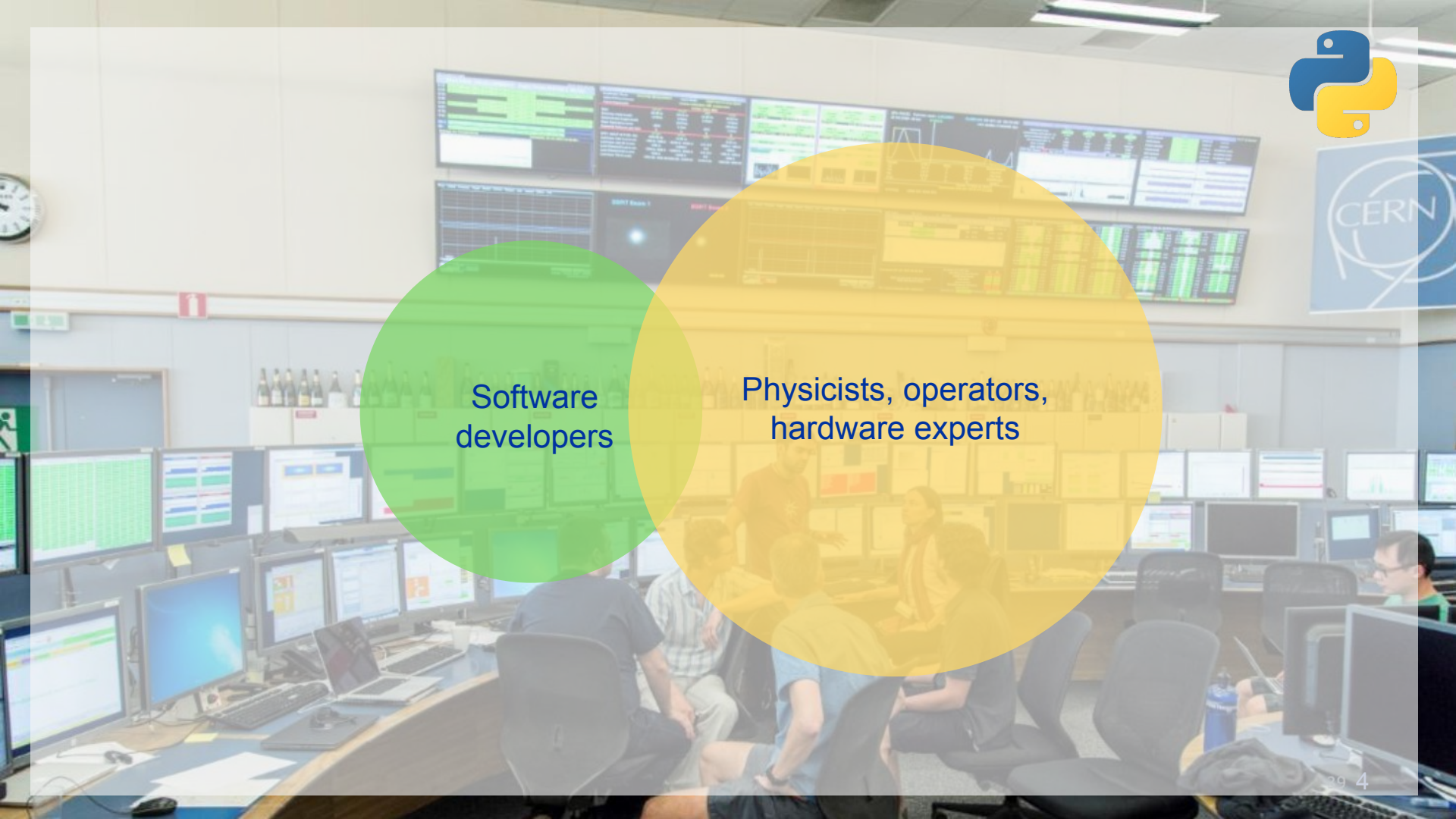
Ivan Sinkarenko (CERN, BE-CSS-SET)

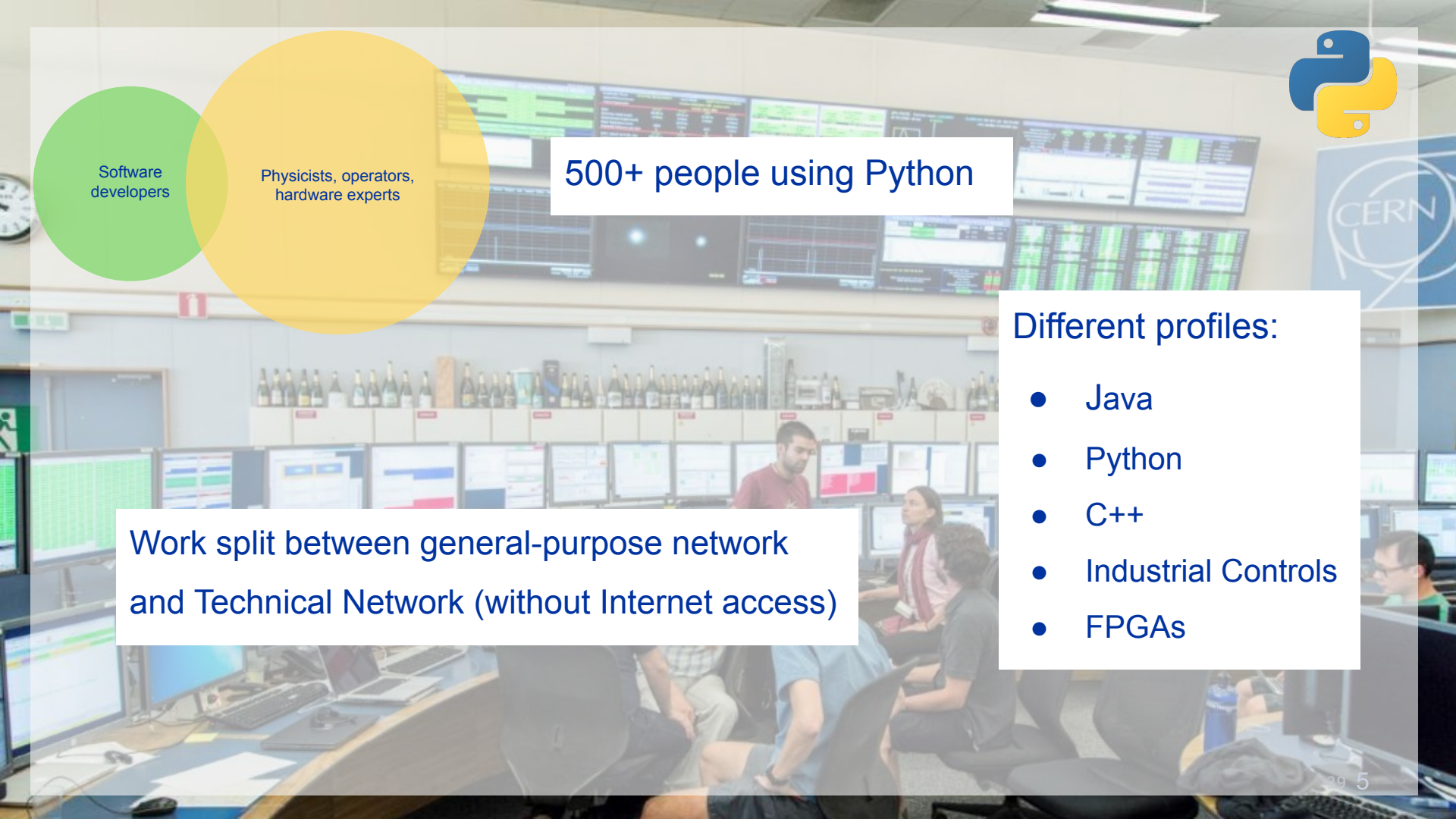
June 2024

Contents

- Accelerator & Technology Sector community at CERN
- Role of Python in 2024
- Zen of Acc-Py
- Acc-Py portfolio
  - Operational application deployment
  - Package Repository
  - DevOps
  - Monitoring
- Community, teaching & guidance

Accelerator & Technology Sector at CERN


Software developers
Physicists, operators, hardware experts


Software developers
Physicists, operators, hardware experts
500+ people using Python
Different profiles:
Java
Python
C++
Industrial Controls
FPGAs
Work split between general-purpose network and Technical Network (without Internet access)
CERN

Role of Python in 2024

Role of Python: Before…

- Python growth used to be uncoordinated, some tools developed for individual use gained popularity in an uncontrolled manner
  - We pay high cost for it even today
- Python officially embraced by the Controls Group ~5 years ago, having created a dedicated Python support team (Acc-Py)
- Initially many had reservations about allowing dynamic language into control rooms, because…
  - It's possible to run non-compiled code from personal directories
  - Broken code does not crash until that line is executed
  - Dynamic typing leaves room for bugs
- Early on, it was possible to get access to the most of control system functionality thanks to pre-existing Java libraries + Jpype, Py4J.

Role of Python: Today

- Today many new projects are started with Python
- Python occasionally chosen to write a new version of existing software using another technology
- Efforts to bring Machine Learning into controls fully rely on Python
- PyQt is appreciated for creating GUIs
- Being the most popular language helps recruitment + easy to get started
- Some find that fast development iterations with Python can produce better performing software due better tuning of the algorithms
- Python evolution is coordinated through Acc-Py team

Zen of Acc-Py

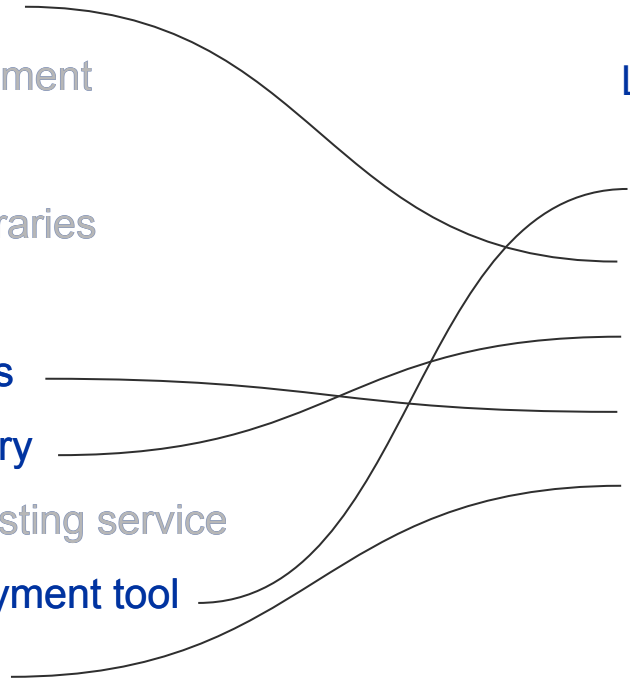
Zen of Acc-Py

- Ensure long-term stability of the code
  - Use of virtual environments in development (no single integrated environment)
  - Preference of packages over scripts
    - Explicitly declared dependencies
    - Code can be tested, released and easily maintained
  - For operational environments, ensure immutability of deployed code
  - Don’t follow the hype in tools and development practices
    - Use stdlib as much as possible, or well established tools, such as pip
    - Follow the trend only when it’s clear to become standard, e.g. pyproject.toml
  - No magical surprises
    - User-site-packages not allowed due to shared home directories
    - No setting of PYTHONPATH or LD_LIBRARY_PATH
- People are free to choose the tools they like, except those that ensure operational stability
- Python should be used for high-level software. Use in Front-End Computers* is discouraged.

Acc-Py portfolio

Acc-Py portfolio

- Python distribution
- Interactive environment
- Devtools
- Controls & GUI libraries
- Container images
- Gitlab CI templates
- Package Repository
- Documentation hosting service
- Operational deployment tool
- Monitoring

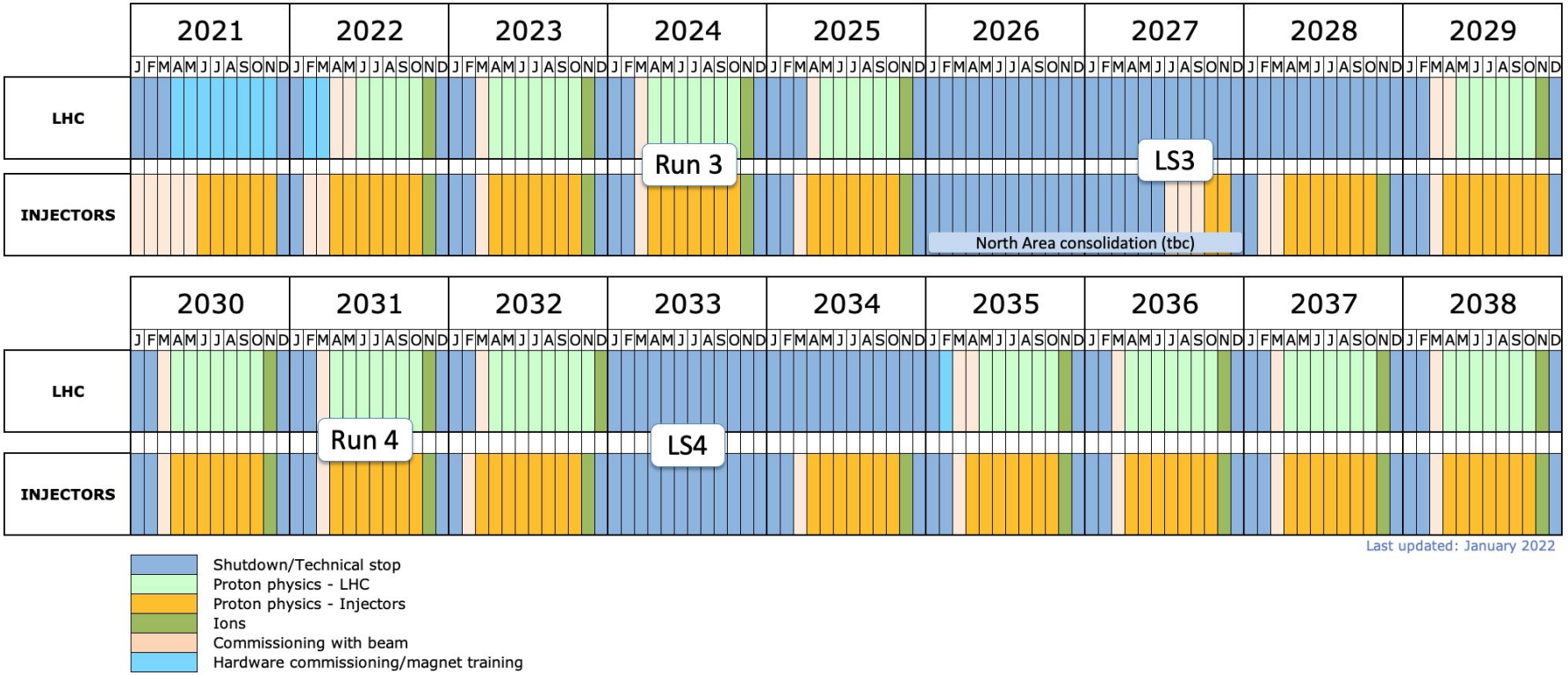


Let's focus on:

- **Operational deployment tool**
- Python distribution
- Package Repository
- Gitlab CI templates
- Monitoring

Operational deployment tool

CERN accelerator schedule

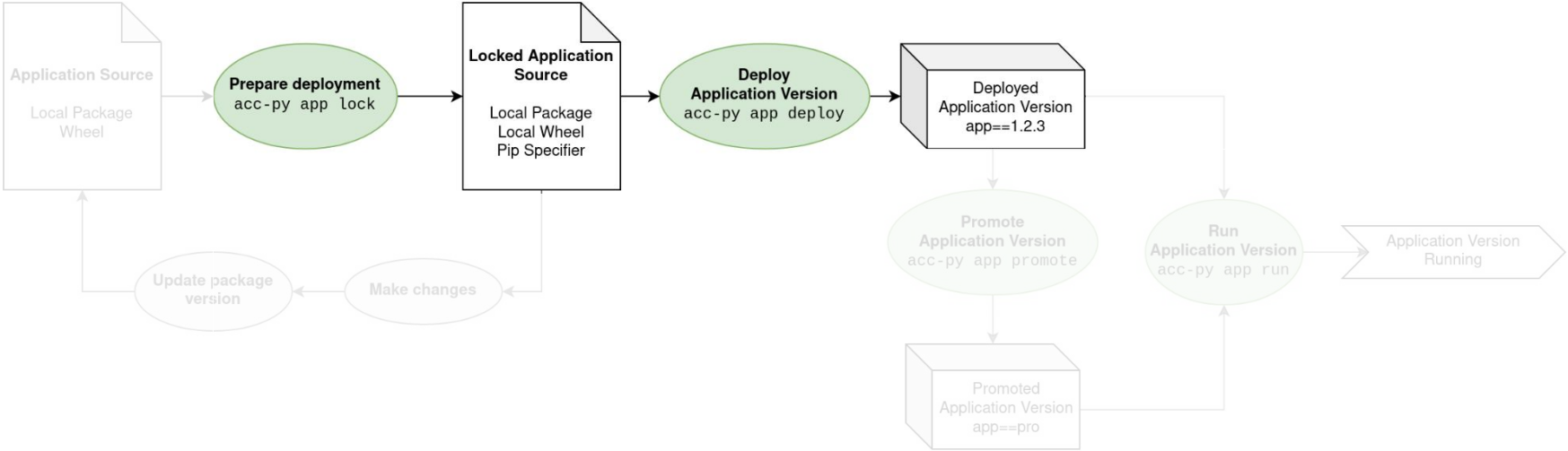


Operational deployment tool

- Traditionally, all control room applications are hosted on a shared NFS, accessible by any console in the room
- With Python, we must ensure that deployed applications are immutable for the duration of the Run*
  - Versioning of the deployed applications
  - Dedicated venv and dependencies per app version
  - Fixed Python interpreter per app version
  - No user write access
  - Deployment of new versions restricted to known authors
- Dependencies in development must match dependencies in production → lock files
  - Both Python and Java dependencies

Operational deployment tool

- We provide a Python CLI tool that developers run themselves



Operational deployment tool

- Traditionally, all control room applications are hosted on a shared NFS, accessible by any console in the room
- With Python, we must ensure that deployed application is immutable for the duration of the Run*
  - Versioning of the deployed applications
  - Dedicated venv and dependencies per app version
  - **Fixed Python interpreter per app version**
  - No user write access
  - Deployment of new versions restricted to known authors
- Dependencies in development must match dependencies in production → lock files
  - Both Python and Java dependencies

Python distribution

Python distribution

- We maintain a curated list of Python interpreter versions
  - Released ~every 2 years. Currently - Python 3.7, 3.9, 3.11. Future - 3.13
  - Kept for the duration of the Run*
  - No user write access
  - No pip available outside of a venv
  - Ensures binary compatibility with low-level libraries, e.g. libstdc++
  - Every new release is tested against major control system libraries
  - Automatic monitoring of Python invocations
- Same Python distribution can be used in development, CI & production

Python distribution

😕 Before Acc-Py, people would interchangeably use

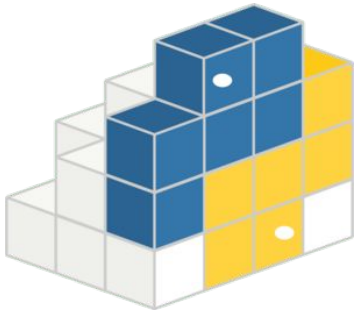
- System Python
- Anaconda
- Python distribution provided by another group with pre-installed dependencies
- Self-compiled Python
- …

Python distribution

☺ Nowadays everyone is using Acc-Py Python

Zen of Acc-Py

- Ensure long-term stability of the code
  - Use of virtual environments in development (no single integrated environment)
  - **Preference of packages over scripts**
    - **Explicitly declared dependencies**
    - **Code can be tested, released and easily maintained**
  - For operational environments, ensure immutability of deployed code
  - Don’t follow the hype in tools and development practices
    - Use stdlib as much as possible, or well established tools, such as pip
    - Follow the trend only when it’s clear to become standard, e.g. pyproject.toml
  - No magical surprises
    - User-site-packages not allowed due to shared home directories
    - No setting of PYTHONPATH or LD_LIBRARY_PATH
- People are free to choose the tools they like, except those that ensure operational stability
- Python should be used for high-level software. Use in Front-End Computers* is discouraged.

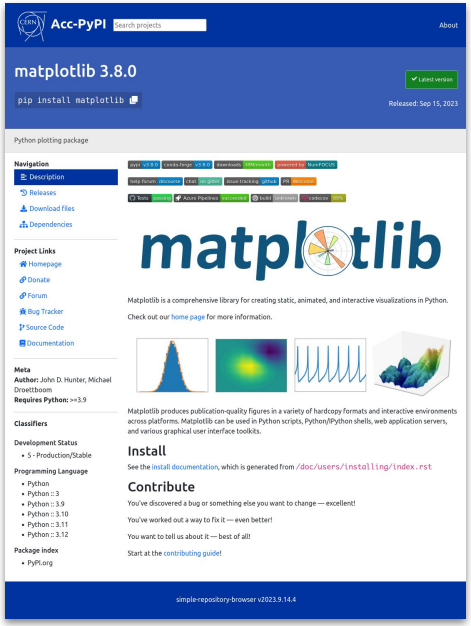


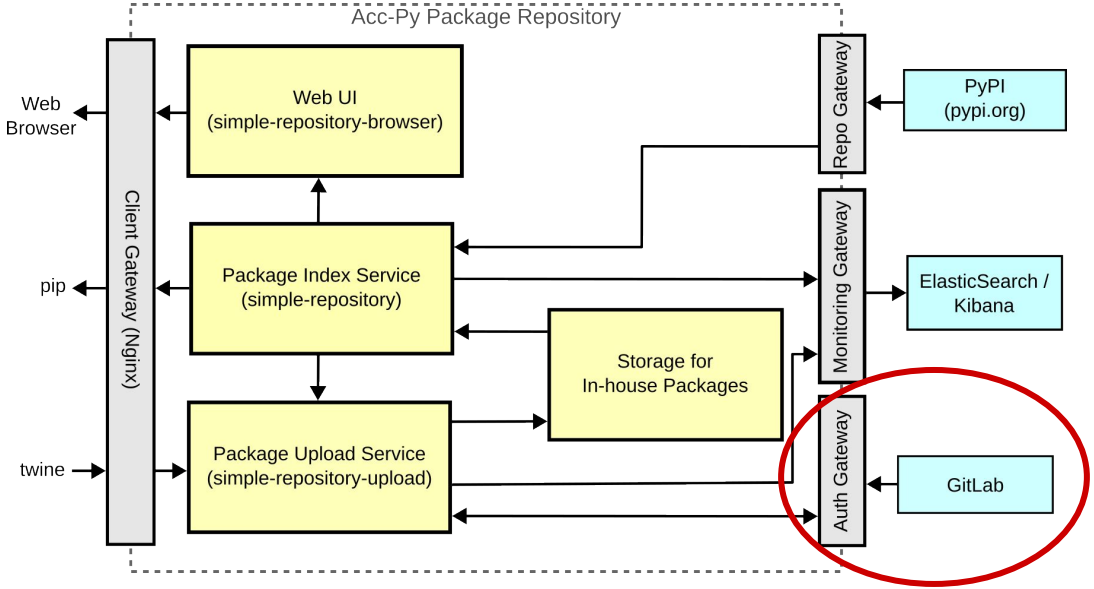
Package Repository

Package Repository

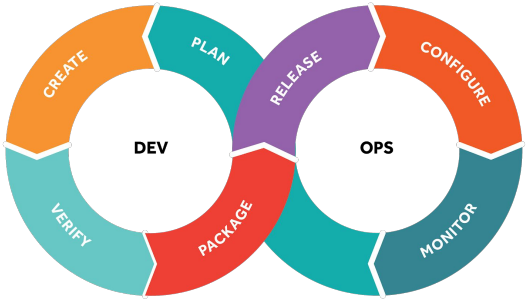
- You could release to PyPI.org, but…
  - Possibility for leaked secrets
  - Internal code rarely makes sense for the outside world
  - Creates a dependency on the 3rd-party service
  - Impossible to publish from technical network
- We run an internal package repository that…
  - Hosts in-house packages locally
  - Mirrors PyPI.org packages with security considerations
  - Uses CERN internal authentication mechanisms
  - Allows custom logic for managing package visibility
- pip is pre-configured to point to our internal repository

Package Repository





https://program.europython.eu/europython-2023/talk/LLHGKF/
https://accelconf.web.cern.ch/icalepcs2023/papers/thpdp067.pdf

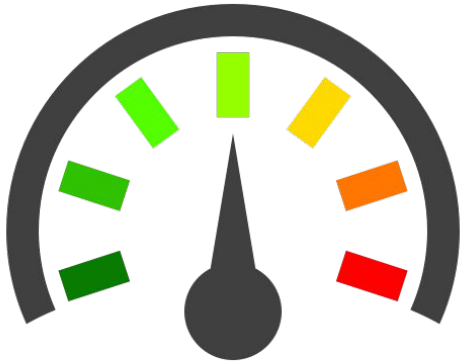

CREATE
PLAN
RELEASE
CONFIGURE
DEV
OPS
VERIFY
PACKAGE
MONITOR

DevOps

DevOps

- Gitlab CI is heavily encouraged for Python projects
- We provide CI templates to
  - lint / test Python code
  - build & publish Python packages
  - build & publish documentation
  - (future) deploy operational applications
  - (future) scan for vulnerabilities

- Templates use Container Images with built-in Acc-Py Python distribution, similar OS and the same binary as the one in deployment locations
- Possible to use the matrix of all supported Python versions

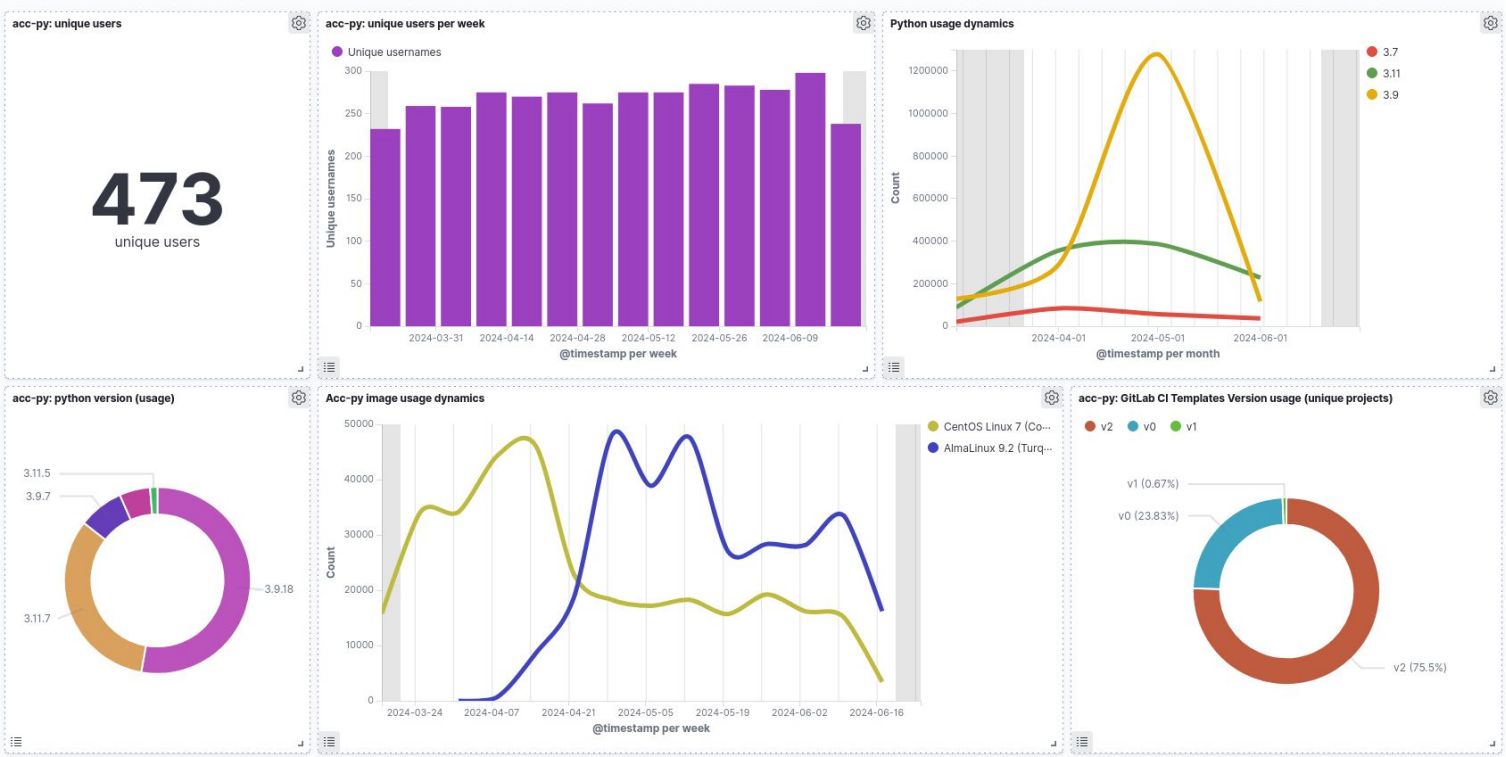


Monitoring

Monitoring

- We want to know our community
- In practice, ~30 out of 500 users actively engage with us
- We've tried running a user survey, but it proved to be high effort
  - Preparation of the survey
  - Spikes of responses only hours/a day after reminders
  - Classification of responses (often challenging, unless Yes/No question)
  - It is useful to prioritize ideas, but insufficient to know when is a good time to integrate them
- So far, continuous monitoring was the most useful tool for analytics

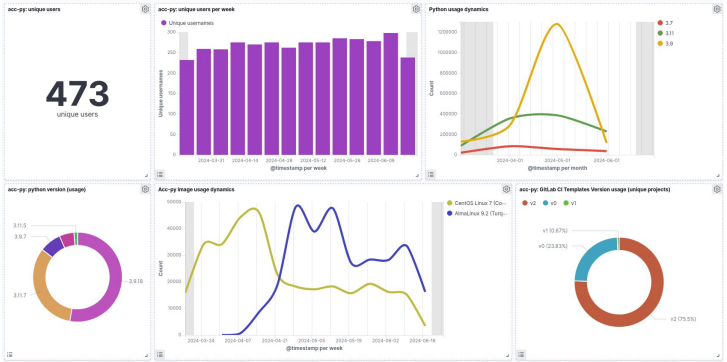
Monitoring: OpenSearch / Kibana



Monitoring: OpenSearch / Kibana

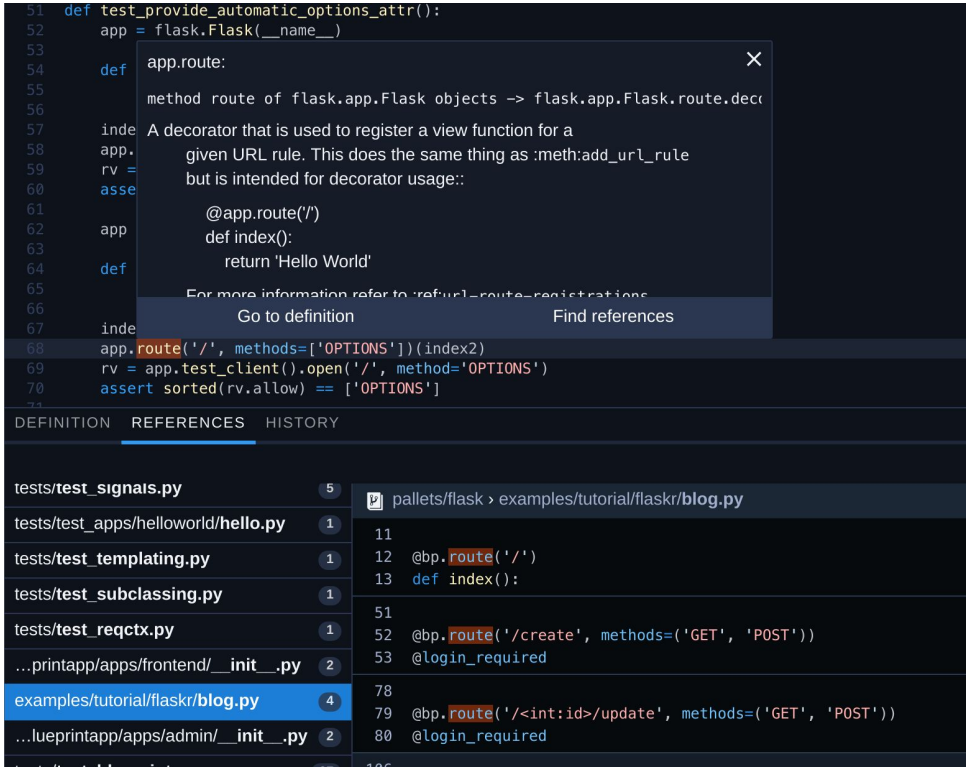
- All Python invocations traced
  - From operational consoles & development PCs
  - Helps to know installed package versions, Python version, etc
  - Distinguishes when run from CI templates

- Helps to know the adoption rate of features / products
- Helps to know what can be deprecated / removed
- Helps to study user preferences
- Last resort for the emergency forensics



Monitoring: SourceGraph

- Code search & navigation
- Scans Gitlab repositories
  - Some Gitlab groups are scanned automatically
  - Others require user opt-in



Community, teaching & guidance

Community, teaching & guidance

- All of community members join mattermost chat, where they can ask for help, or make suggestions
- Mailing list alongside the mattermost chat is used for important announcements
- We do regular presentations to give status updates and promote best practices
- We participate in shaping the Python learning resources
- Python team occasionally conducts code reviews for other teams within the group

- Teaching & guidance **is a never-ending process** - reality of CERN contract rotations, and participation of “second-job developers”. We also learn new things every day!


"That's all Folks!"

Useful links

- Introducing Python as a Supported Language for Accelerator Controls at CERN
  - https://accelconf.web.cern.ch/icalepcs2021/papers/mopv040.pdf
- The Python Package Repository Accelerating Software Development at CERN
  - https://program.europython.eu/europython-2023/talk/LLHGKF/
- Towards a Flexible and Secure Python Package Repository Service
  - https://accelconf.web.cern.ch/icalepcs2023/papers/thpdp067.pdf
- Protecting Your Controls Infrastructure Supply Chain
  - https://accelconf.web.cern.ch/icalepcs2023/papers/mo4bco03.pdf


THANKS FOR LISTENING
now dont STOP CLAPPING
meme-arsenal.ru