Data Science Survival Skills

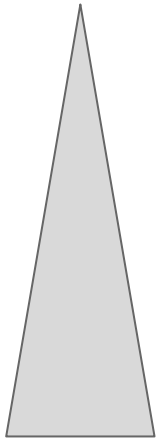
Code Deployment

Sharing code with others

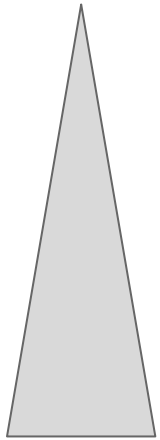
- Who is your audience?


- What is the purpose?

Sharing code



- Raw code
- Library (*.dll, *.so)
- “EXE”
- Installer



User friendliness

Effort by developer

Sharing raw code

Recap from earlier lectures:

- Code files
- Comments
- **Documentation**

⇒ Readthedocs and alike!

Code

What do I need to run your code?

- OS (Windows, Mac OS X, Linux, ...)
- Additional hardware? GPU, Network access, ...
- Drivers
- Environment
- Libraries/dependencies
- Library version
- Code editor
- ...

We have a REPRODUCE RESEARCH RESULTS (RRR) project seminar to tackle exact this!

Platform

- Windows 32 bit/64 bit (x86, x64)
- Mac OS X (10.5 runs 64-bit)
- Linux/Ubuntu
- ARM

Sharing Python code

- Setup.py
- Requirements.txt
- Documentation!
- pip installable?

ARM platform

ARM vs x86
Also 32/64 bit
RISC vs. CISC
(reduced instruction set computer vs. complex ...)

Advantages

- Lower TDP
- More efficient
- Low power requirements

Microsoft Surface with ARM
Macbook Air M
Android phones
iPhones
iPads...
Coffee machines

Use tools for “compiling” Python files

- PyInstaller / fbs (this lecture)
- Py2exe
- Py2app (Mac OS)
- Nuitka
- Cython

Examples

Netflix

- Server-side: Java and Python, ensuring compatibility across different platforms
- Using containers with Docker for consistent deployment

Microsoft Office Suite

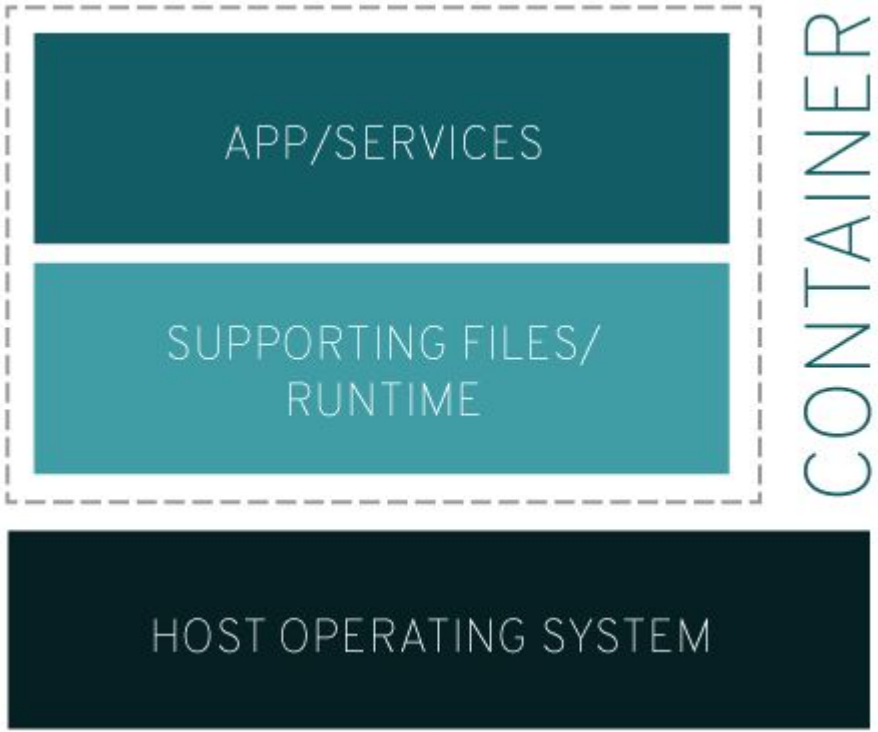
- Developed with unified codebase for Windows, macOS, iOS, Android
- Utilizes platform specific adaptations for optimal performance and user experience

Games

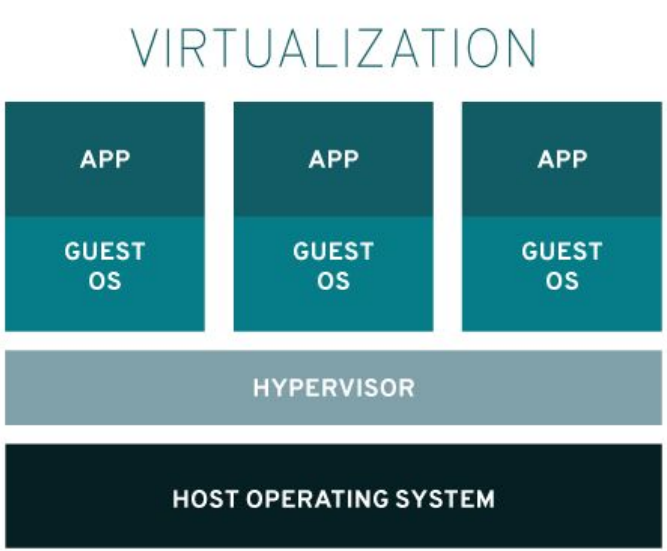
- Unity or Unreal Engine target multiple devices like PC, consoles, mobile devices
- Focus on consistent performance and user experience across hardware

Shipping the whole box - Containers

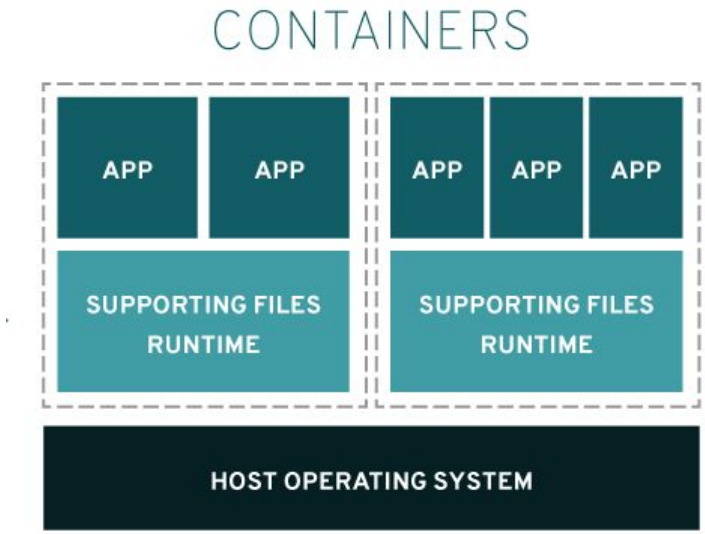
Running code out of the box - container!



Virtualization vs. container



Hypervisor emulates hardware, which allows multiple operating systems to run side by side

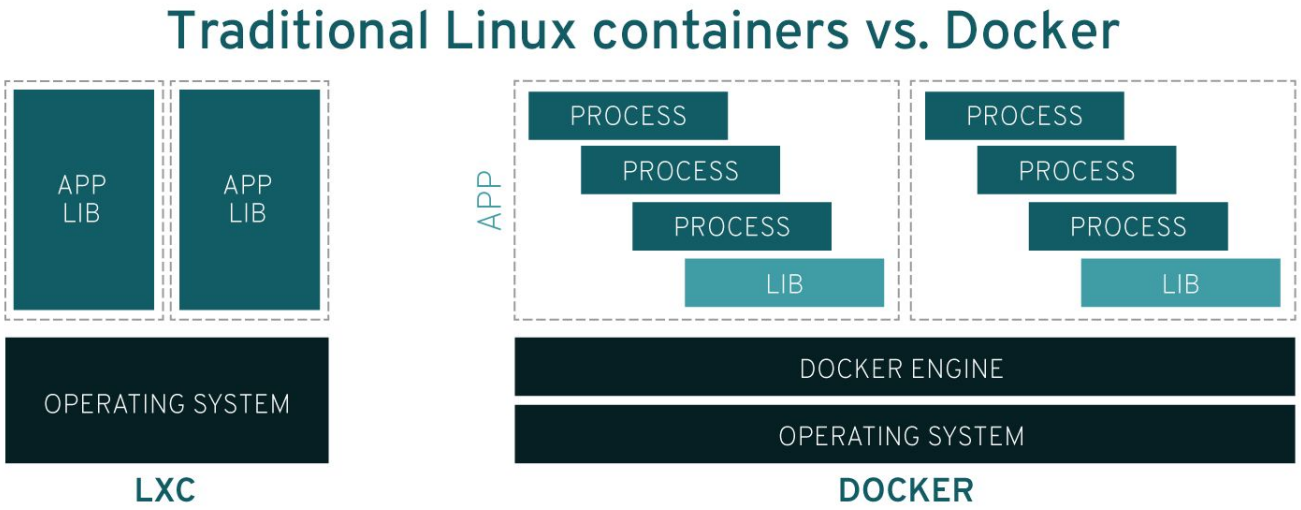


Containers are

- less resource-intensive
- Have a standard interface (start, stop, environment variables, etc.)
- retain application isolation
- are more easily managed as part of a larger application (multiple containers)

Docker vs LXC

LXD – Virtual Machine



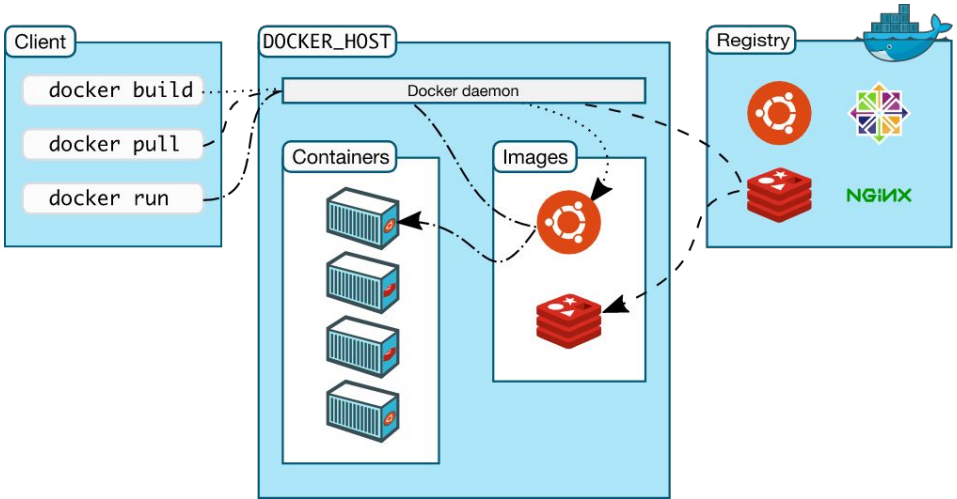
LXC is shipped with a full Linux environment: file system, networking and multiple applications. Docker relied on LXC, but now it has a Docker Engine being more lightweight.
Further, Docker promotes processes (broke down from Apps) instead of full Apps

Docker and Kubernetes

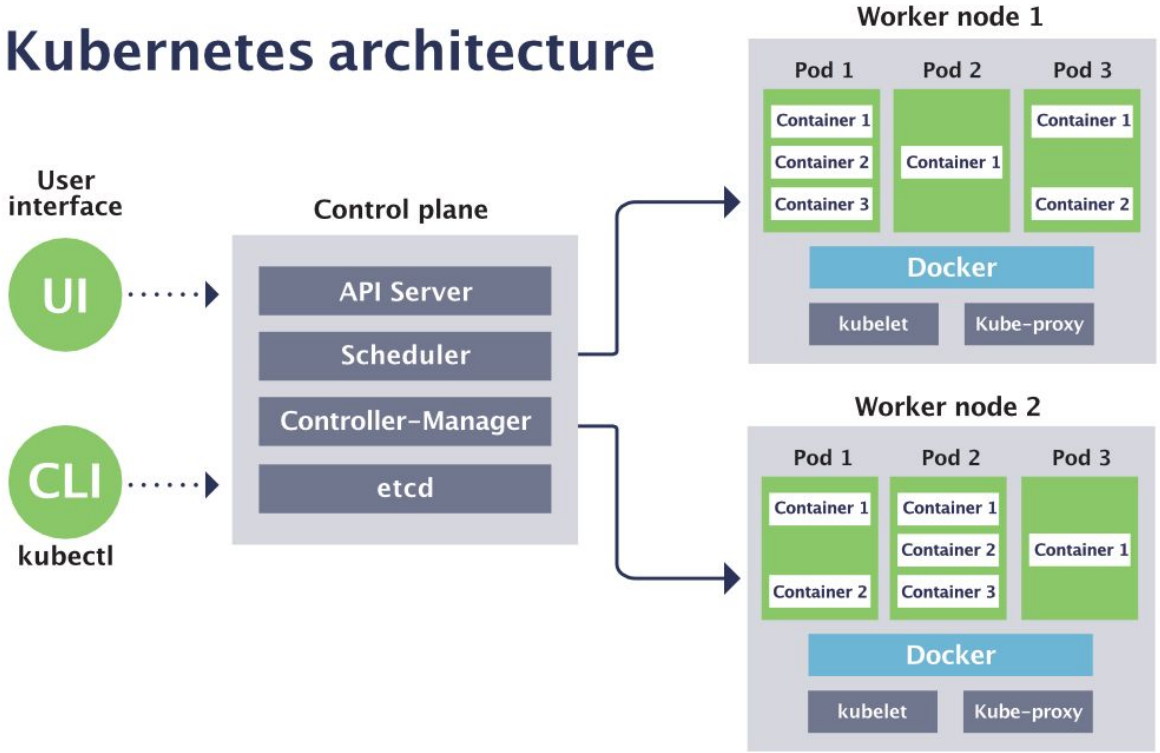
Docker uses CONTAINERS:

To summarize, a container:

- is a runnable instance of an image. You can create, start, stop, move, or delete a container using the DockerAPI or CLI.
- can be run on local machines, virtual machines or deployed to the cloud.
- is portable (can be run on any OS)
- Containers are isolated from each other and run their own software, binaries, and configurations.



Kubernetes

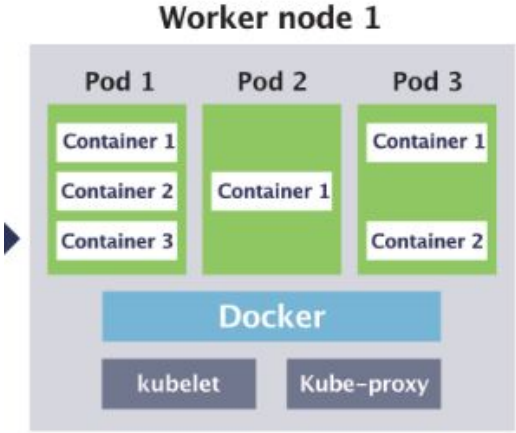


https://sensu.io/blog/how-kubernetes-works

Kubernetes Pods

A Kubernetes pod is a group of containers and is the smallest unit that Kubernetes administers.

- Pods have a single IP address
  → applied to every container within the pod
- Containers in a pod share the same resources such as memory and storage. → containers inside a pod are treated collectively as a single application, as if all the containerized processes were running together on the same host in more traditional workloads.
- A pod can have a single container and more. Only one container is common, when the application or service is a single process that needs to run.



https://sensu.io/blog/how-kubernetes-works

CI/CD

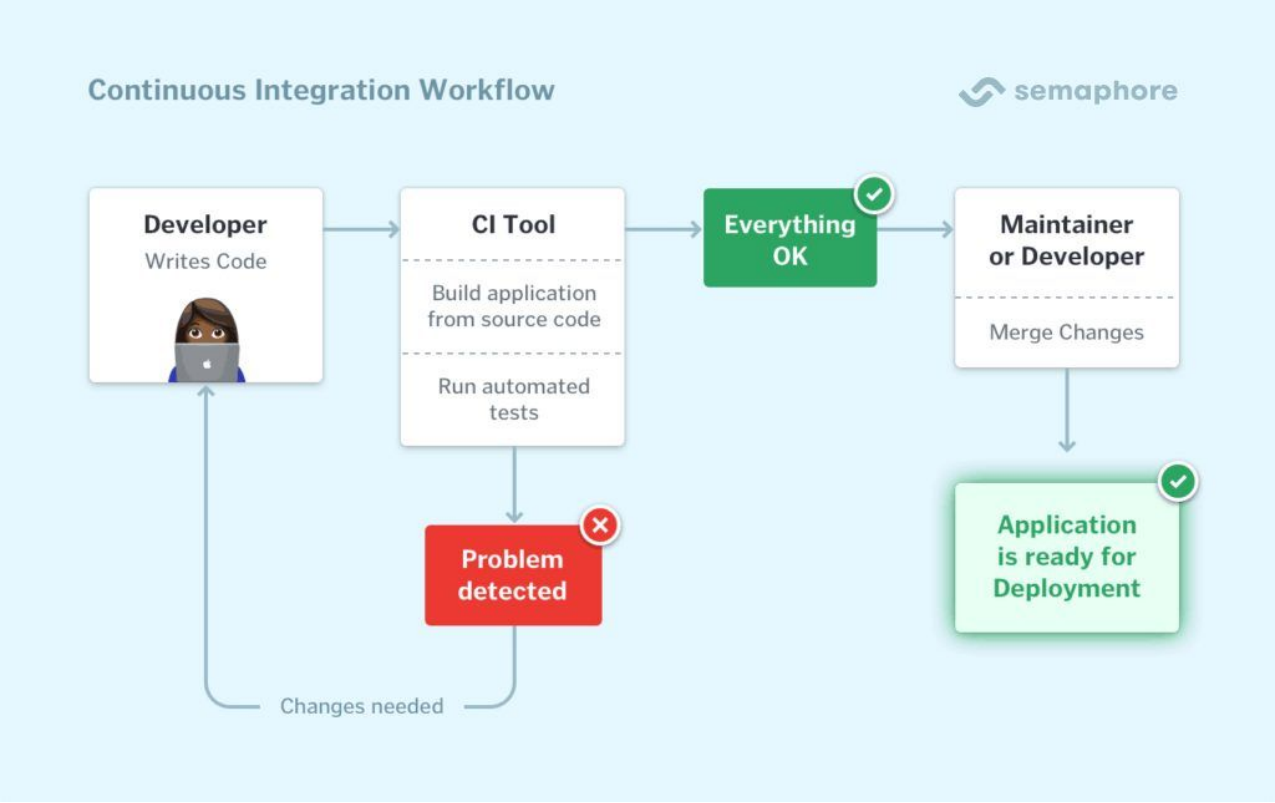
Understanding Continuous Integration (CI)

Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day.

Each integration is verified by an **automated build and automated tests** to detect integration errors as quickly as possible.

- Maintain a single source of truth for the codebase
- Facilitate early detection of integration issues and conflicts
- Ensure that the software is in a deployable state **at all times**

CI process



Tools:

- Jenkins
- Travis CI
- CircleCI
- GitLab CI

Travis CI

$64
per month

Starter
Ideal for hobby projects

CI Workflow

CI Workflow Steps:

- Code Commit: Developers regularly push code to a shared repository.
- Automated Build: The CI server automatically builds the system and runs unit and integration tests.
- Test Results: If tests pass, the build is considered stable. If they fail, the team is alerted to fix issues immediately.

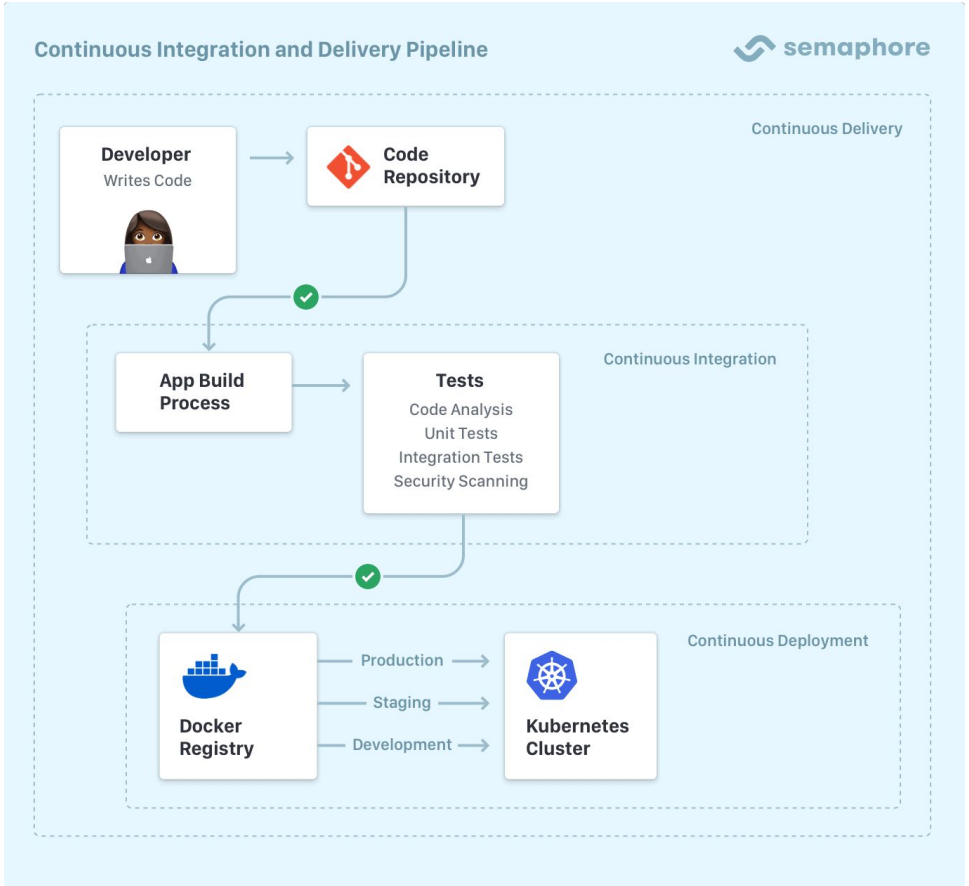
Automated Testing in CI:

- Integral part of CI to ensure code integrity after each commit.
- Types of tests: Unit, Integration, Functional, and Performance Tests.

CI Server Role:

- Monitors the repository and triggers a build upon new code commits.
- Manages the process of building, testing, and reporting.

Continuous Deployment



Continuous Deployment is an advanced practice where every change that passes all stages of the production pipeline is released to customers automatically, **without explicit approval**.

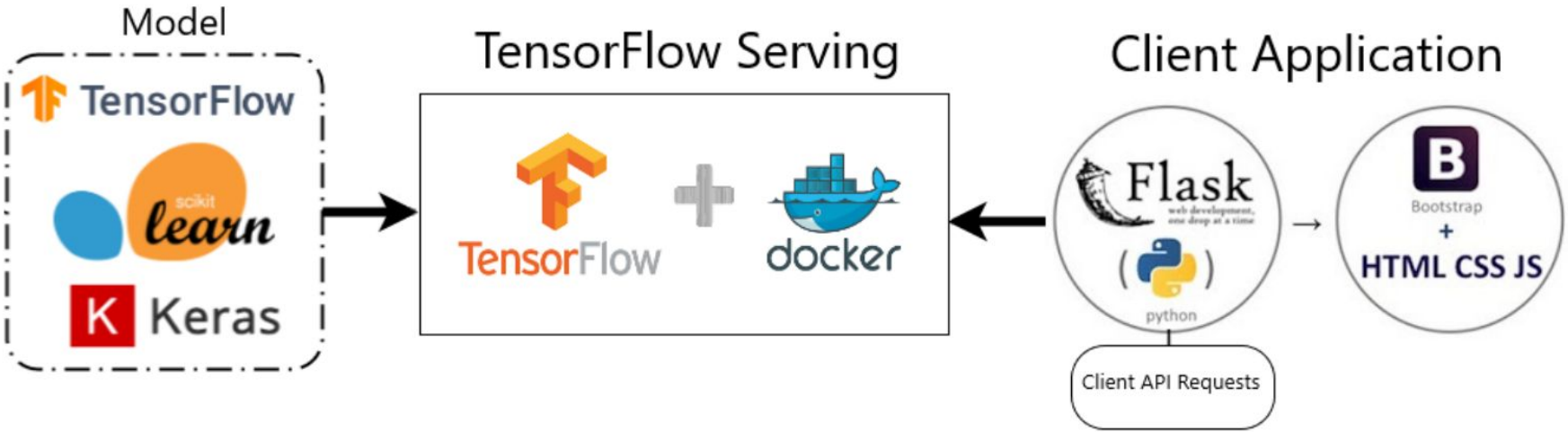
CI vs CD

Goals of CD: To automate the software release process and enable rapid, reliable, and frequent deployments and to reduce the time taken to bring new features, fixes, and updates to the end-users.

CD vs CI: While CI focuses on integrating and testing code changes, CD automates the next steps of **deploying changes to production environments. ⇒ CD ensures a fully automated pipeline from code commit to deployment.**

Deploy AI models

Deploying TensorFlow models



TFLITE

How it works

Pick a model

Pick a new model or retrain an existing one.

Read the developer guide →

Convert

Convert a TensorFlow model into a compressed flat buffer with the TensorFlow Lite Converter.

Read the developer guide →

Deploy

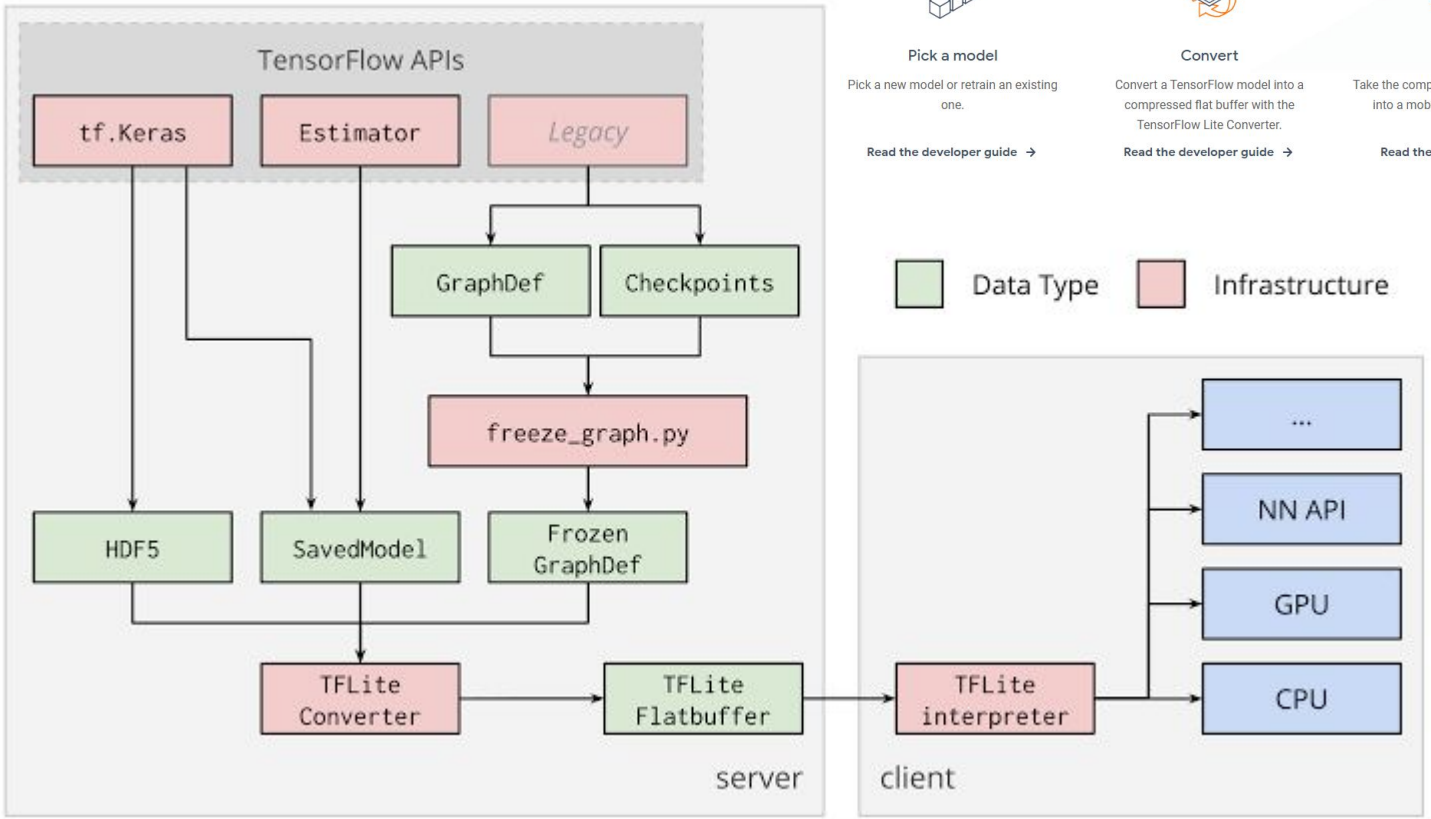
Take the compressed .tflite file and load it into a mobile or embedded device.

Read the developer guide →

Optimize

Quantize by converting 32-bit floats to more efficient 8-bit integers or run on GPU.

Read the developer guide →



TFLITE on µCs

TensorFlow Lite for Microcontrollers

TensorFlow Lite for Microcontrollers is designed to run machine learning models on microcontrollers and other devices with only few kilobytes of memory. The core runtime just fits in 16 KB on an Arm Cortex M3 and can run many basic models. It doesn't require operating system support, any standard C or C++ libraries, or dynamic memory allocation.

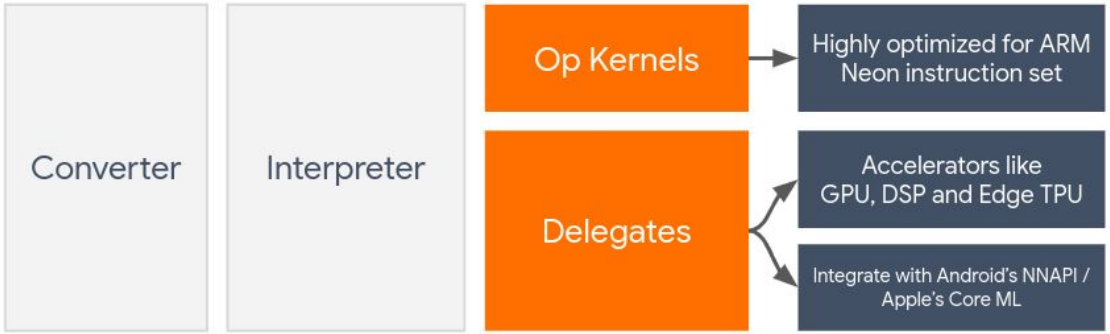
The following development boards are supported:

- Arduino Nano 33 BLE Sense
- SparkFun Edge
- STM32F746 Discovery kit
- Adafruit EdgeBadge
- Adafruit TensorFlow Lite for Microcontrollers Kit
- Adafruit Circuit Playground Bluefruit
- Espressif ESP32-DevKitC
- Espressif ESP-EYE
- Wio Terminal: ATSAMD51
- Himax WE-I Plus EVB Endpoint AI Development Board
- Synopsys DesignWare ARC EM Software Development Platform
- Sony Spresense

- Very limited memory
- Very limited throughput
- Limited operations
- Not easy to get to work

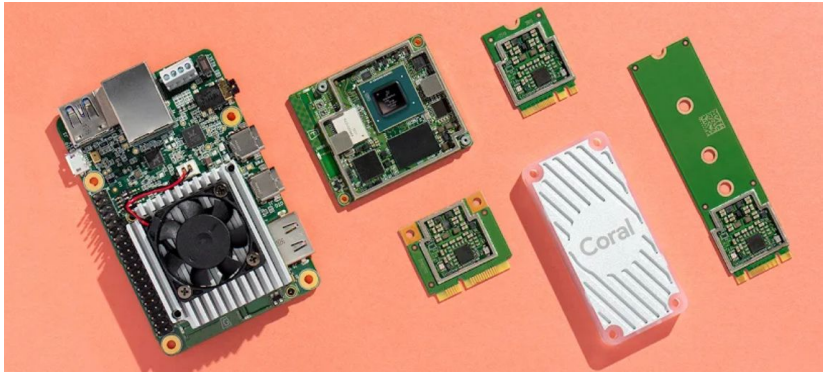
Fun stuff: https://experiments.withgoogle.com/collection/tfliteformicrocontrollers

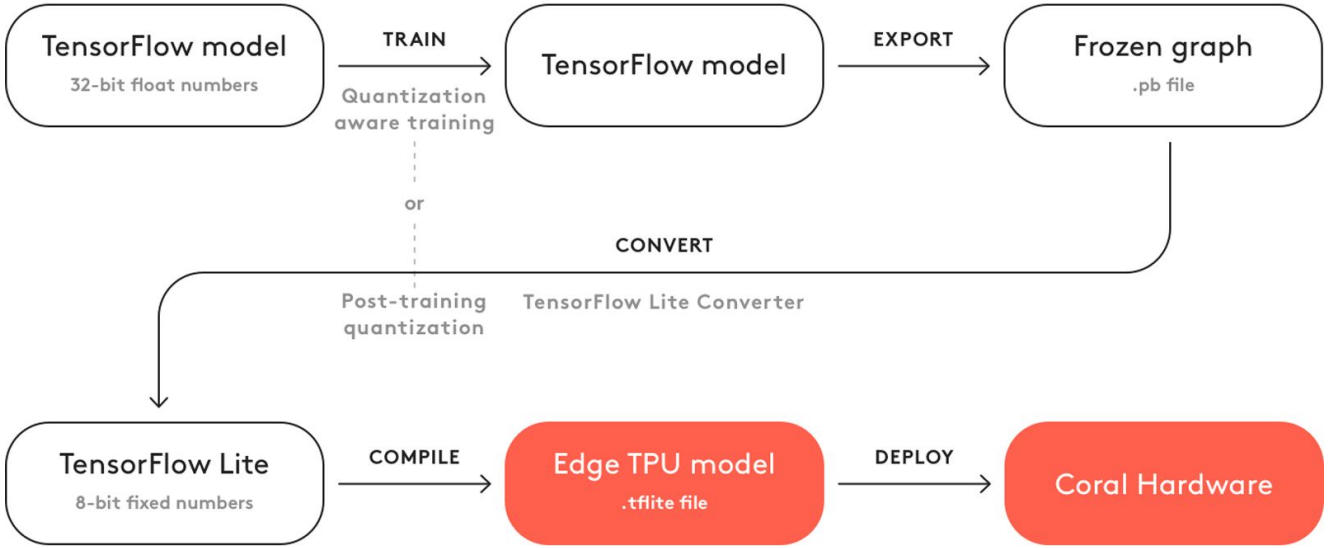
TFLITE on Delegates (e.g. EdgeTPU)



Model Type	GPU	NNAPI	Hexagon	CoreML
		Android		iOS
Floating-point (32 bit)	Yes	Yes	No	Yes
Post-training float16 quantization	Yes	No	No	Yes
Post-training dynamic range quantization	Yes	Yes	No	No
Post-training integer quantization	Yes	Yes	Yes	No
Quantization-aware training	Yes	Yes	Yes	No

Edge TPU


Coral


TensorFlow model
32-bit float numbers
TRAIN
Quantization aware training
TensorFlow model
EXPORT
Frozen graph
.pb file
or
CONVERT
Post-training quantization
TensorFlow Lite Converter
TensorFlow Lite
8-bit fixed numbers
COMPILE
Edge TPU model
.tflite file
DEPLOY
Coral Hardware

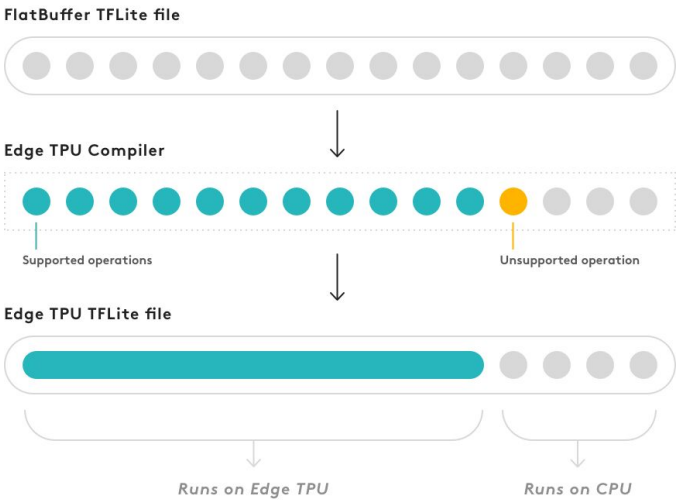
Edge TPU

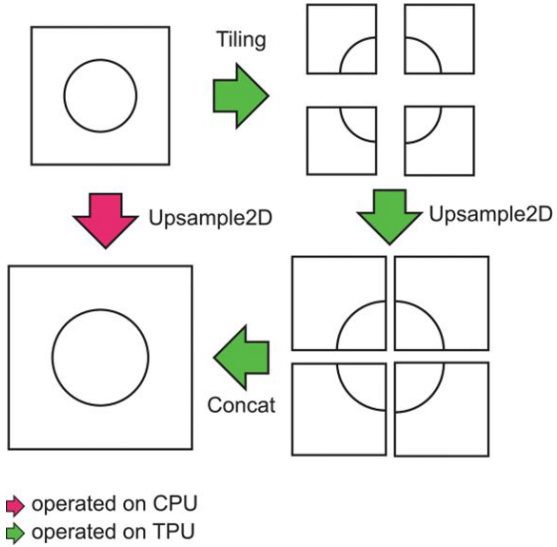
Model requirements

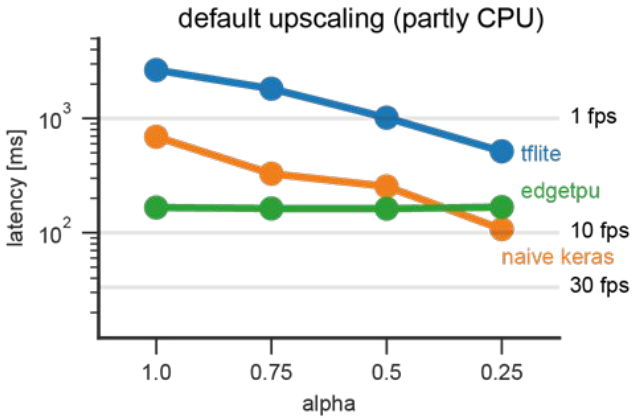
If you want to build a TensorFlow model that takes full advantage of the Edge TPU for accelerated inferencing, the model must meet these basic requirements:

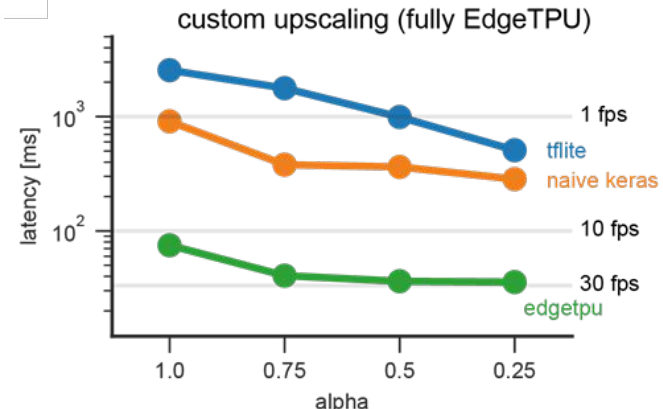
- **Tensor parameters are quantized** (8-bit fixed-point numbers; int8 or uint8).
- **Tensor sizes are constant at compile-time** (no dynamic sizes).
- **Model parameters (such as bias tensors) are constant at compile-time.**
- **Tensors are either 1-, 2-, or 3-dimensional.** If a tensor has more than 3 dimensions, then only the 3 innermost dimensions may have a size greater than 1.
- **The model uses only the operations supported by the Edge TPU** (see **table 1** below).

Edge TPU





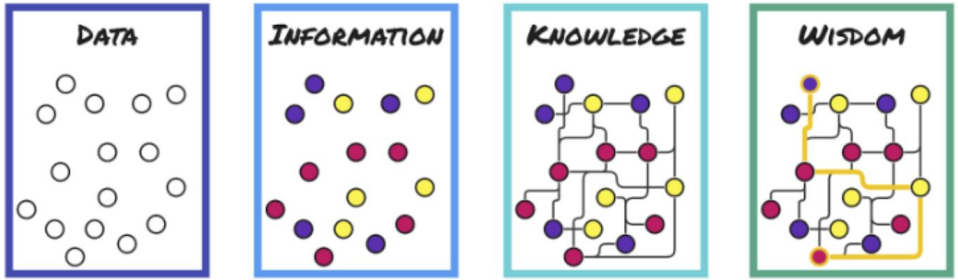




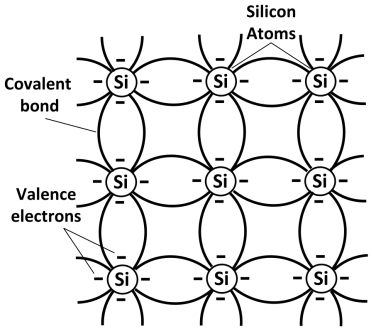
Kist and Döllinger, IEEE Access 2020

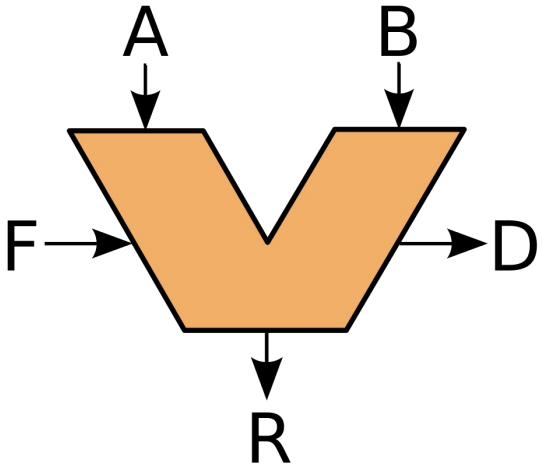
DSSS Summary

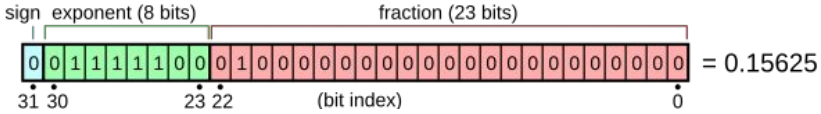
From data to knowledge

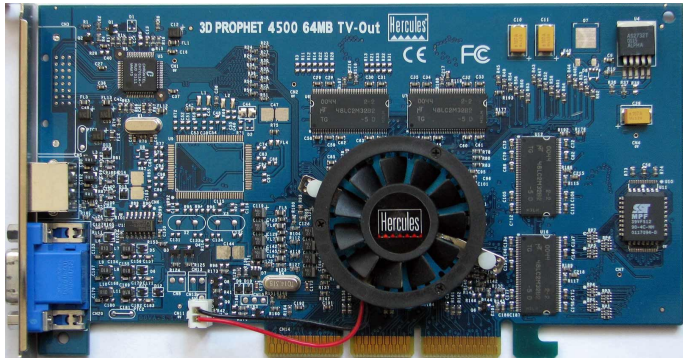


Hardware









NumPy

Versioning and project management

ALPHA

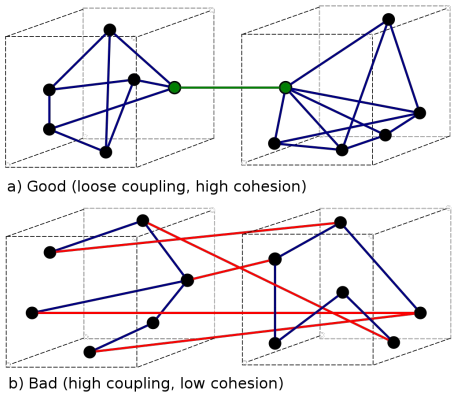
Early internal testing focused on core functionality

BETA

Wider testing for functionality and user feedback

RELEASE CANDIDATE

Final testing to confirm stability and readiness for release



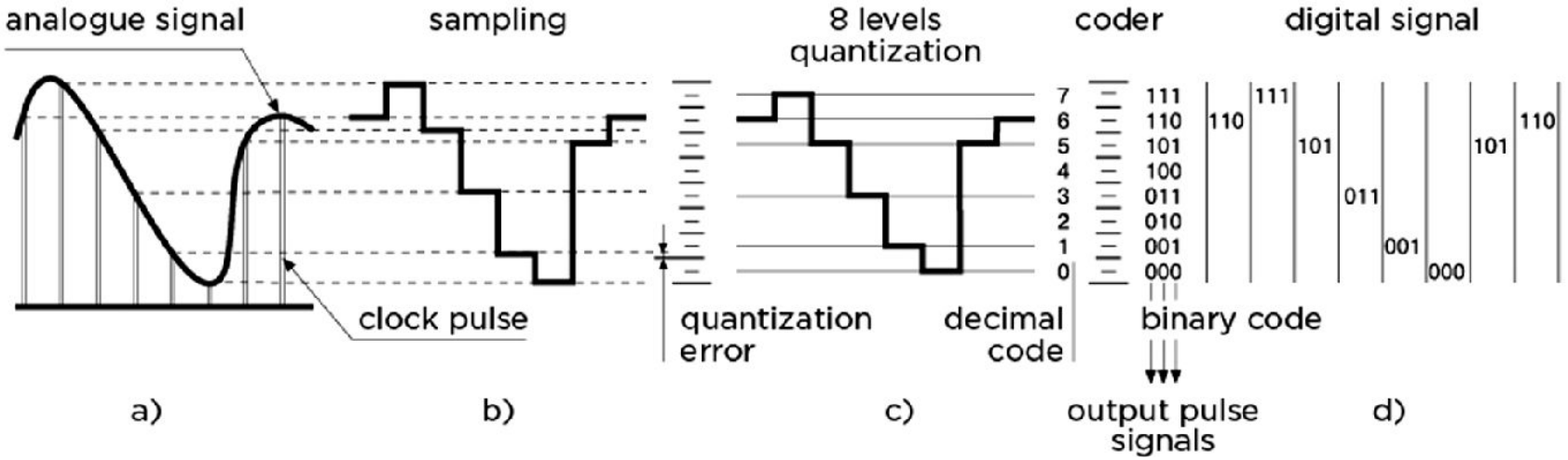
a) Good (loose coupling, high cohesion)

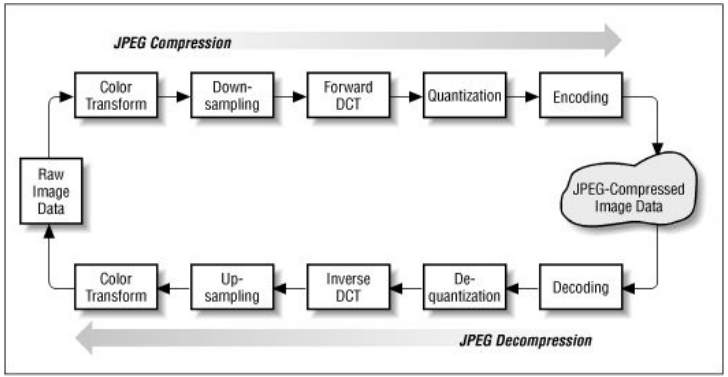
b) Bad (high coupling, low cohesion)

Version control, Docs, ...

```python
def factorial(n: int, double: bool = False) -> int:
    """
    Calculates the factorial of a given integer.

    Args:
        n (int): The integer to calculate the factorial of.
        double (bool, optional): If True, calculates the double factorial. Defaults to Fa

    Returns:
        int: The factorial or double factorial of `n`.

    Raises:
        ValueError: If `n` is negative.

    Examples:
        >>> factorial(5)
        120
        >>> factorial(5, double=True)
        15
    """
    # Function logic...
```

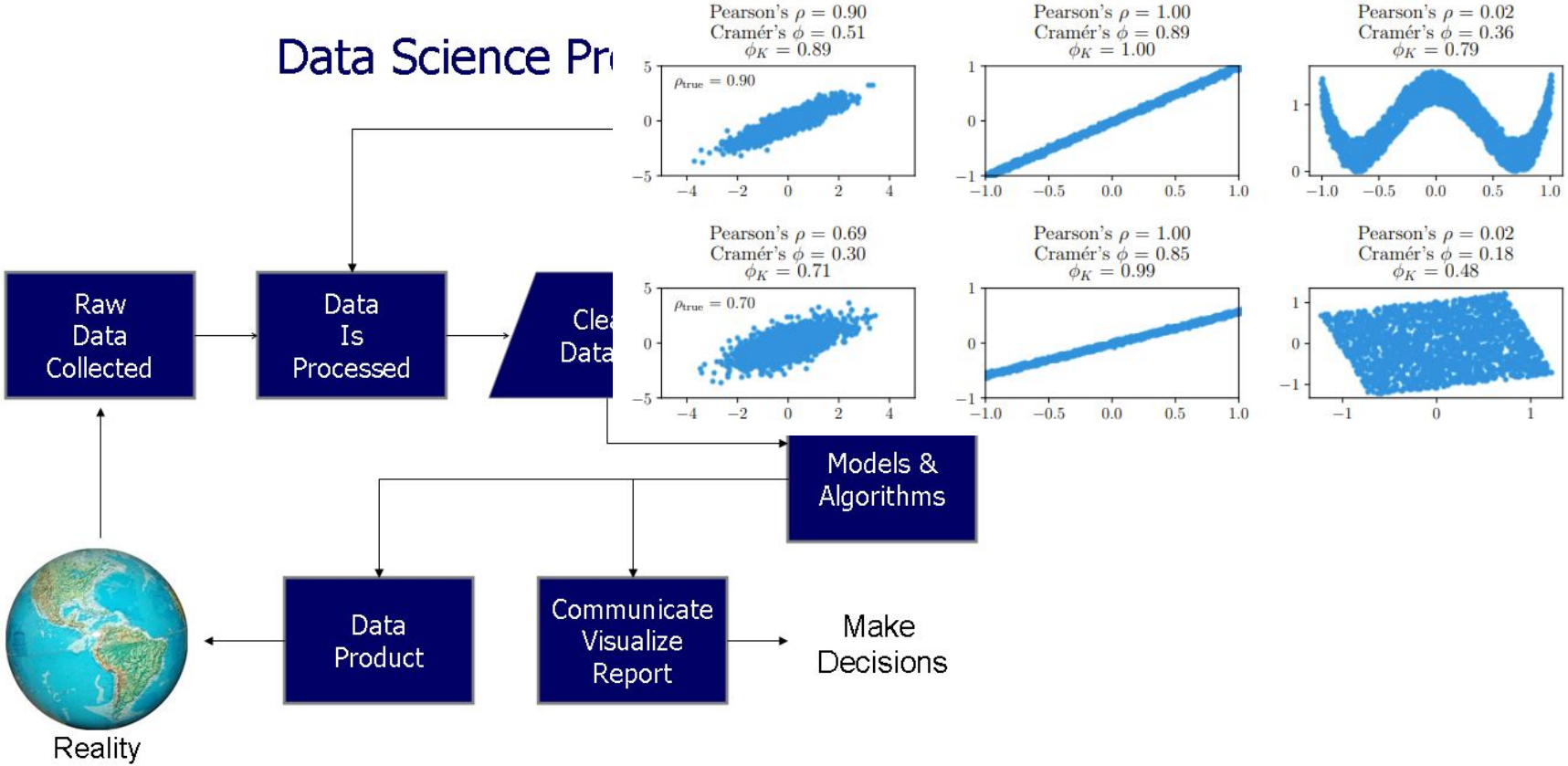
Data





Data types, zip files, ...

Data exploration



Data vis

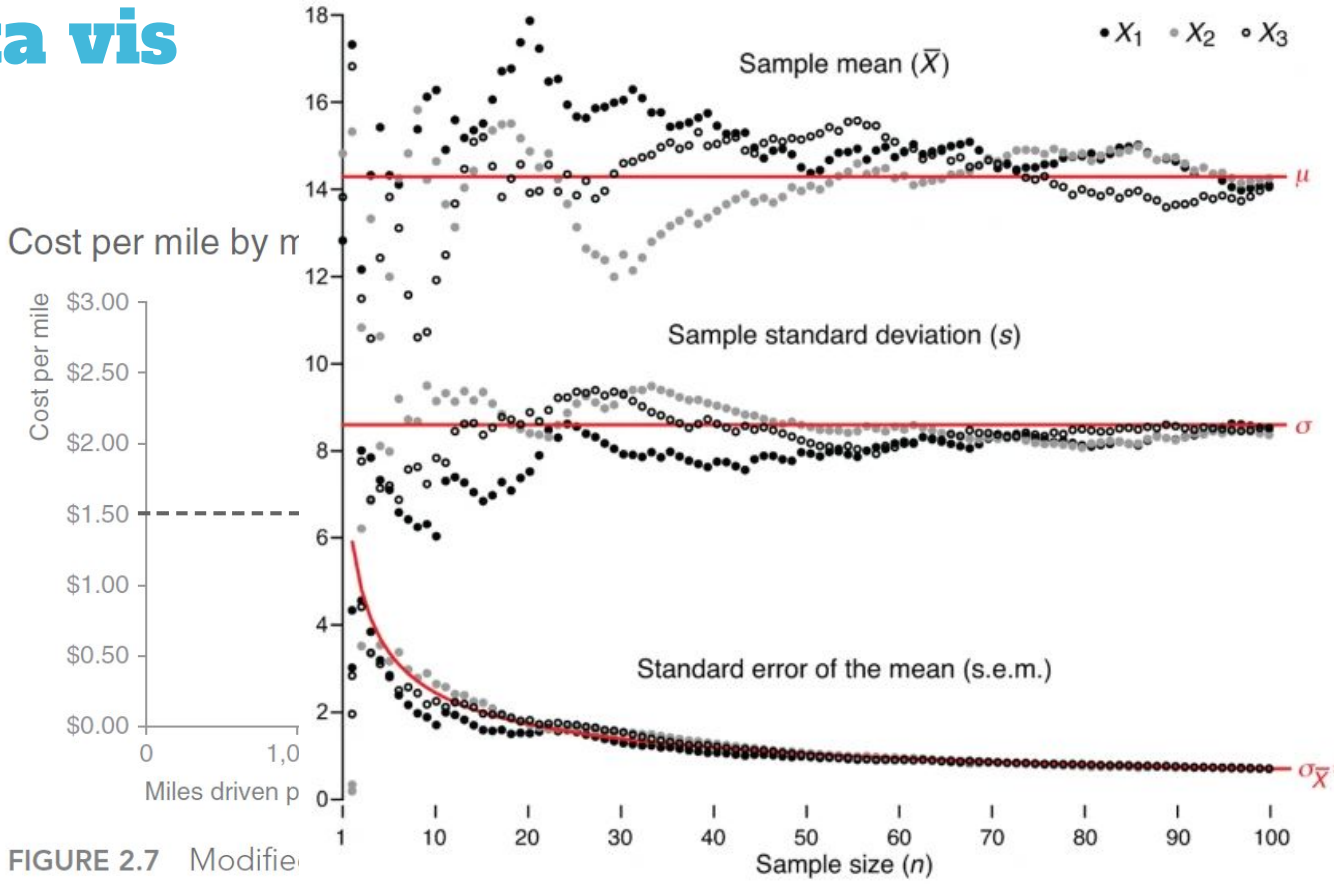
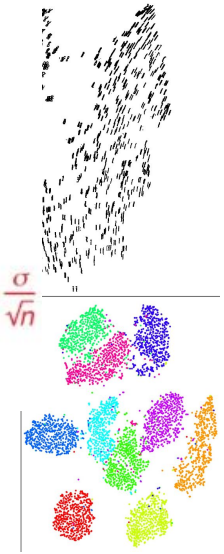
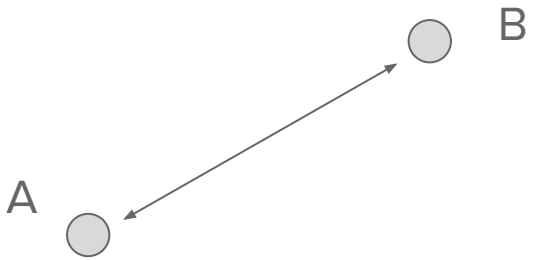


FIGURE 2.7 Modifie



Baselines

Computing a distance



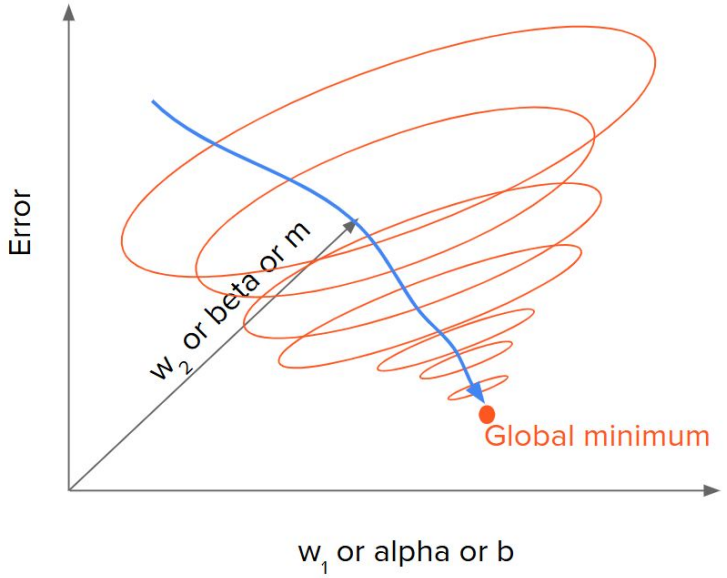
Baseline algorithm

Naive approach, e.g. computing odds

Classical math/analytic, exact solution

Established ML/DL solutions

ML



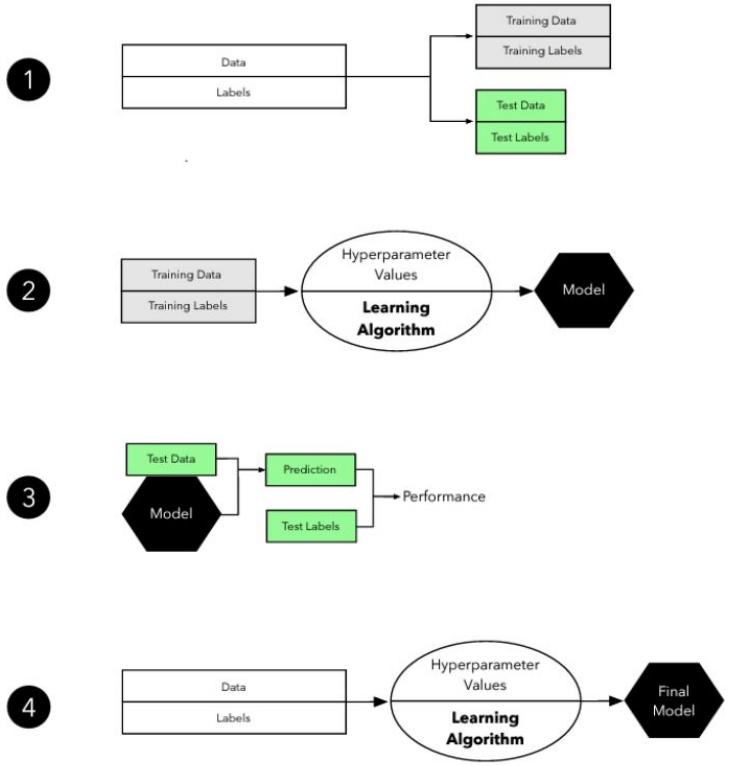
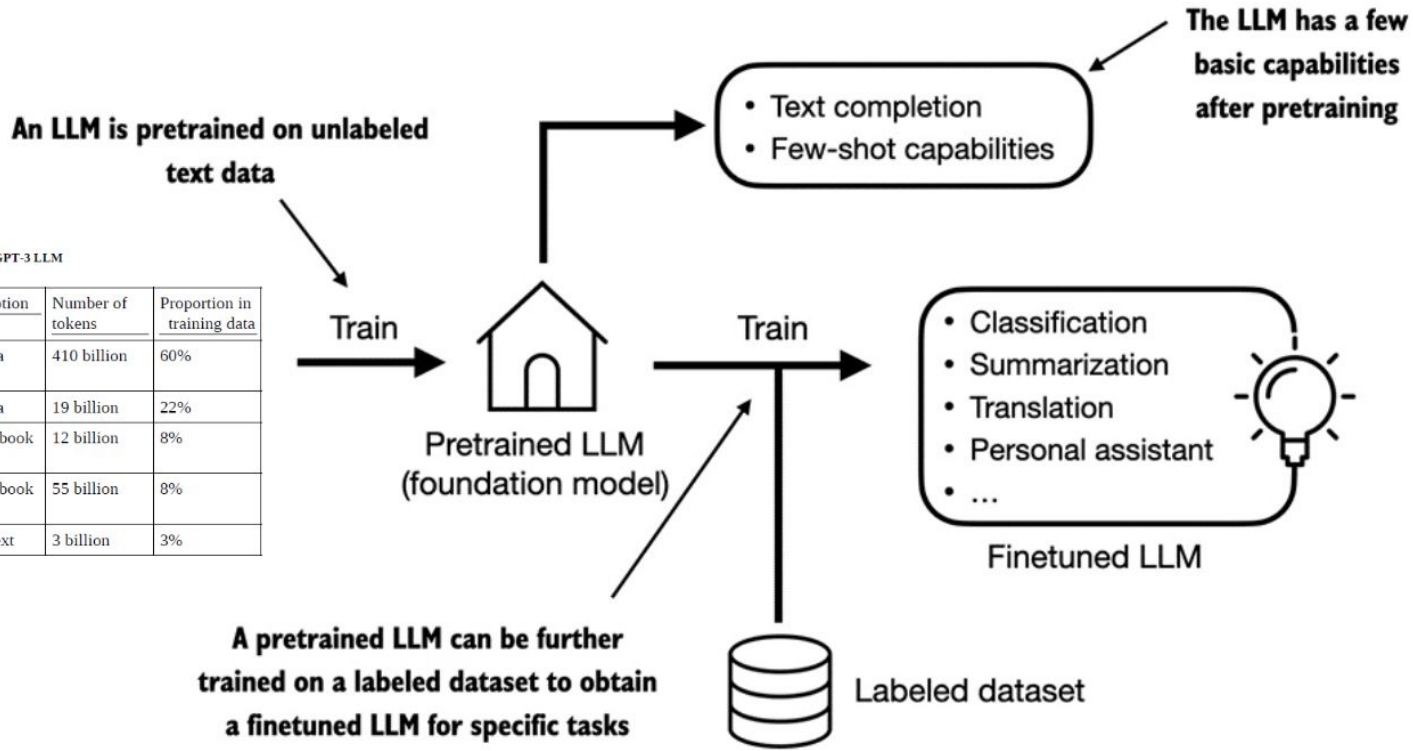


Figure 2: Visual summary of the holdout validation method.

NLP

Table 1.1 The pretraining dataset of the popular GPT-3 LLM

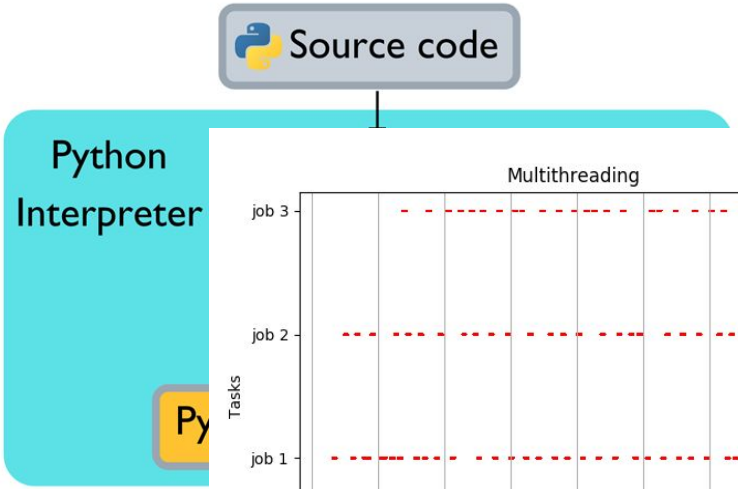
Dataset name	Dataset description	Number of tokens	Proportion in training data
CommonCrawl (filtered)	Web crawl data	410 billion	60%
WebText2	Web crawl data	19 billion	22%
Books1	Internet-based book corpus	12 billion	8%
Books2	Internet-based book corpus	55 billion	8%
Wikipedia	High-quality text	3 billion	3%

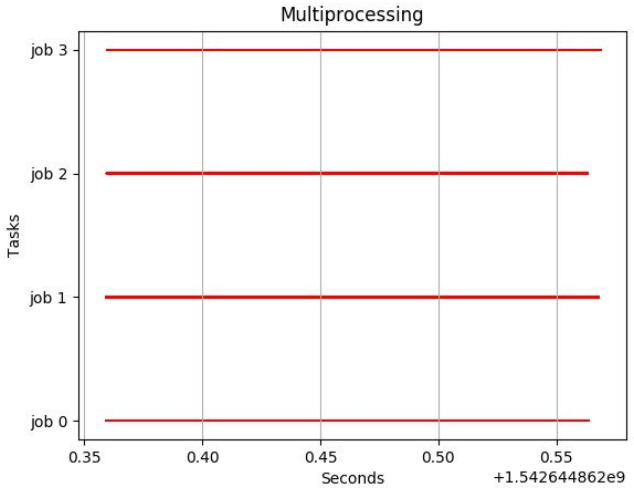


Multithreading/-processing

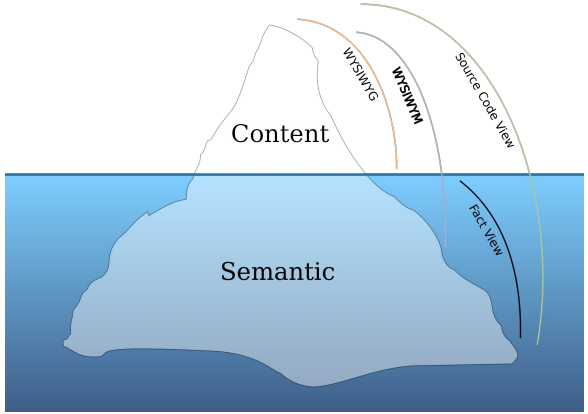
integrate_cy.pyx

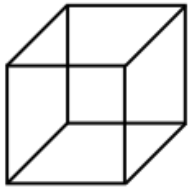
```
def f(double x):
    return x ** 2 - x


def integrate_f(double a, double b, int N):
    cdef int i
    cdef double s
    cdef double dx
```

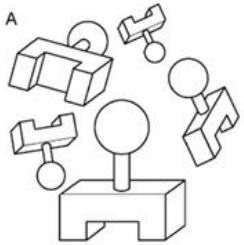


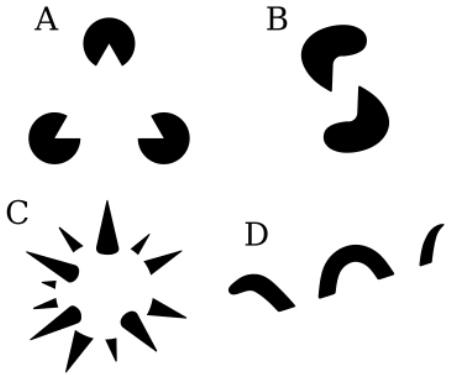


GUI


Content
WYSIWYG
WYSIWYM
Source Code View
Fact View
Semantic




A


A
B
C
D