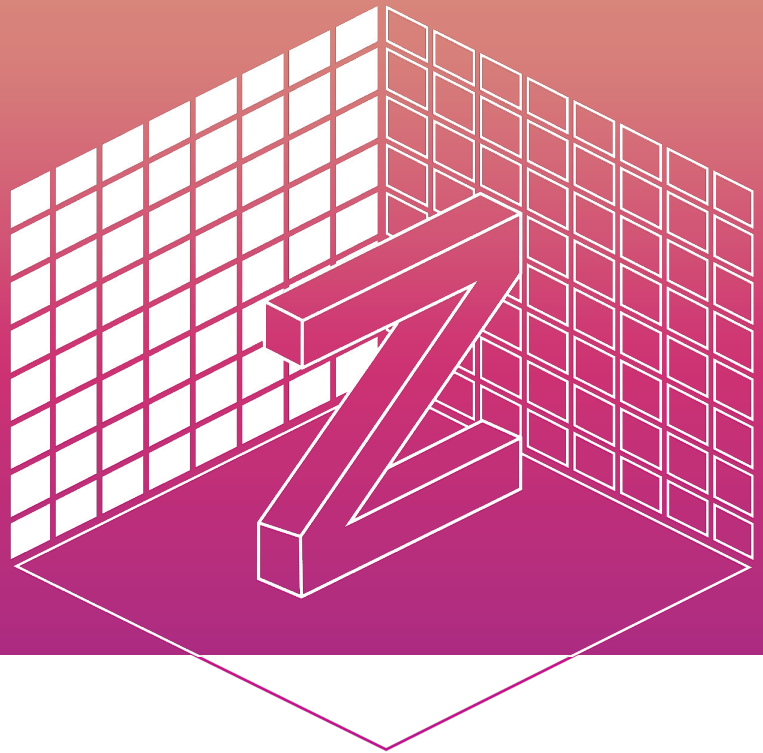


Chunked, Compressed, & Cloud-native
N-dimensional arrays

From Tensors to Clouds—Guide to Zarr-Python 3

Sanket Verma

Community and OSS @ Zarr

https://sanketverma.com/

Slides

https://bit.ly/zp3_pycon_de

Slides & Notebook

https://bit.ly/zp3_more

What I'll be talking about?

→ What is Zarr?

→ What's new in the Zarr-Python 3?

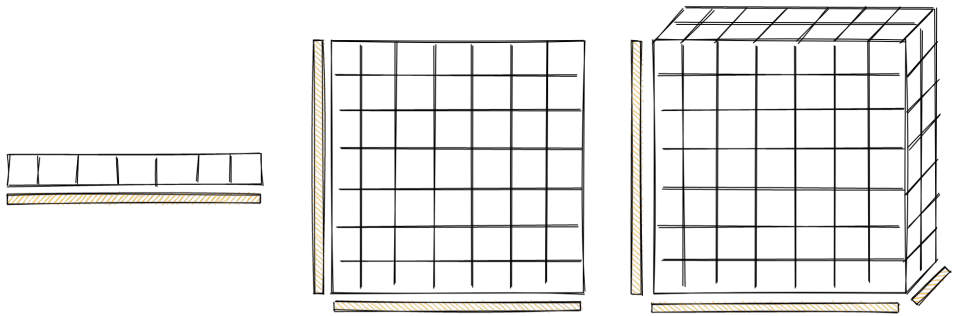
→ Demo

→ How to get involved?

How Zarr works?

How Zarr works?

Arrays are container of items of the same data-type & size (in bits). The number of dimensions and items in container are described by the shape.



shape	(7,)	(7,7)	(2,7,7)
# dimensions	1D	2D	3D
# items	7	7 * 7	7 * 7 * 2

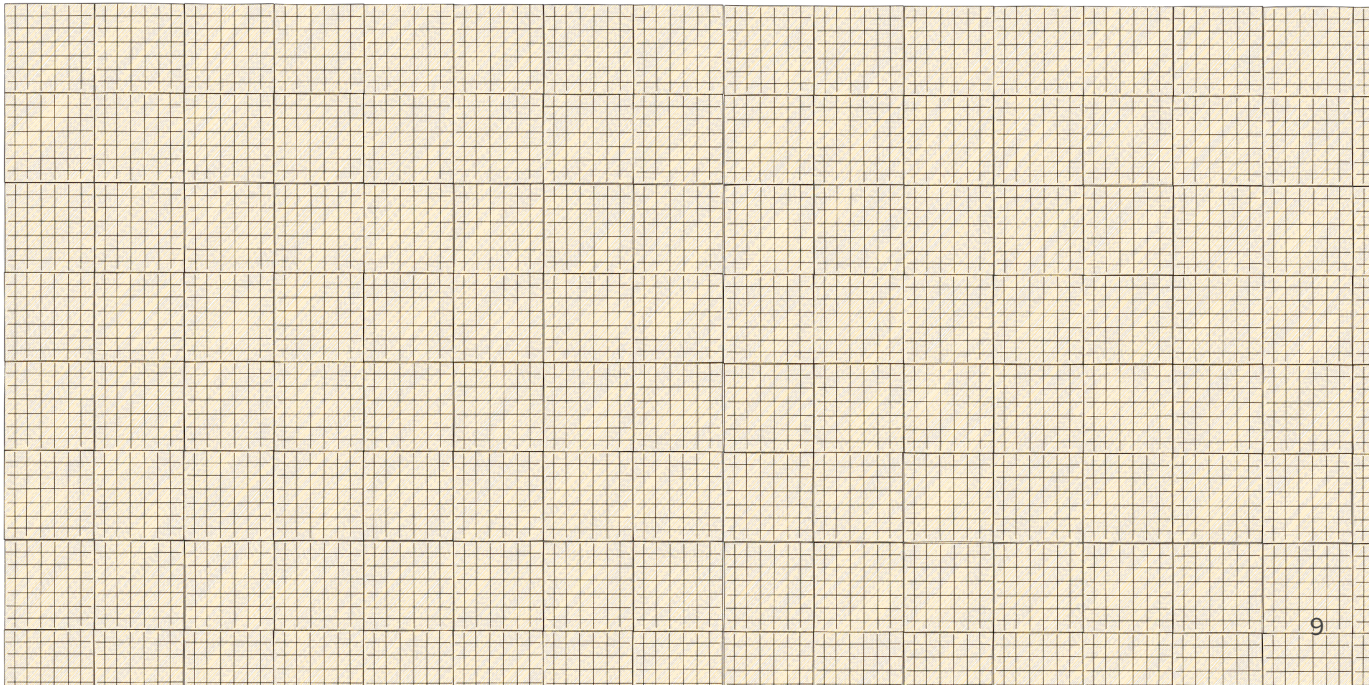
Credit: @trevmanz

How Zarr works?

Credit: @trevmanz

How Zarr works?

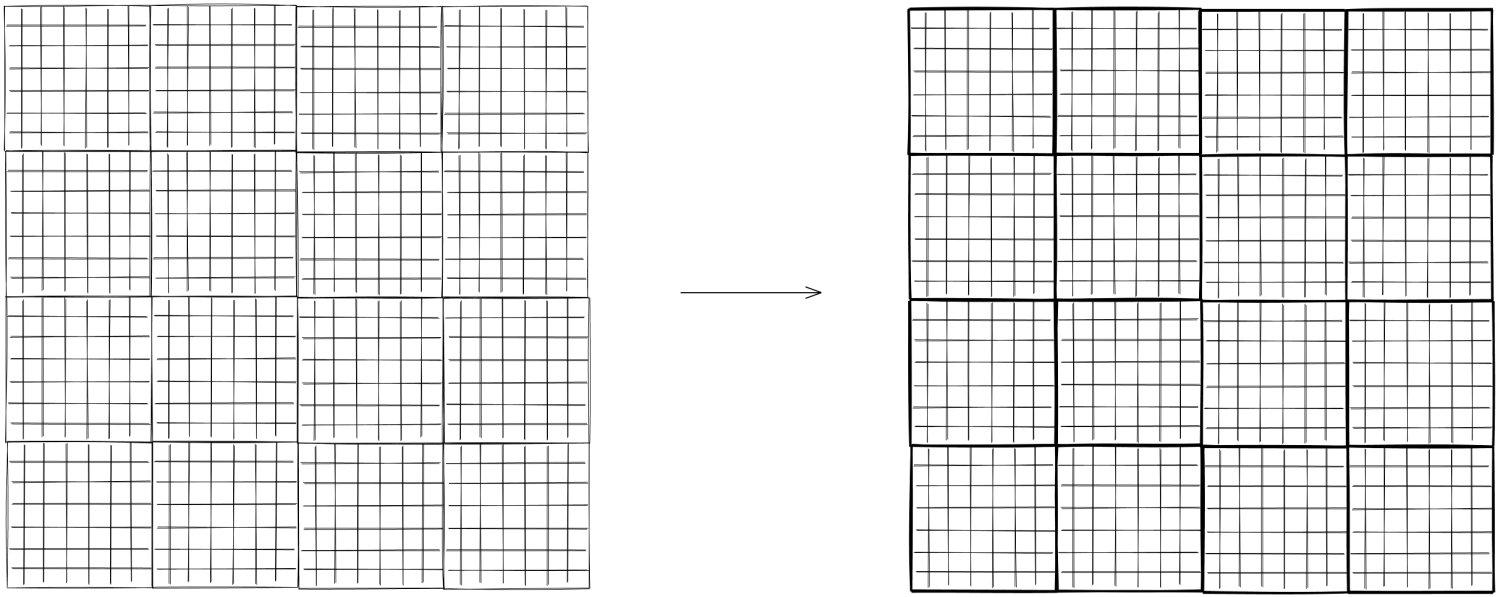
What if the data is too big to fit in memory?



Credit: @trevmanz

How Zarr works?

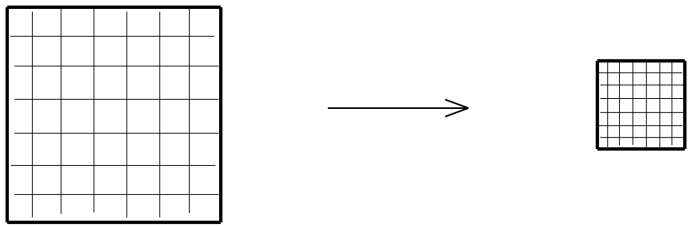
Divide array into chunks (Chunking)



Credit: @trevmanz

How Zarr works?

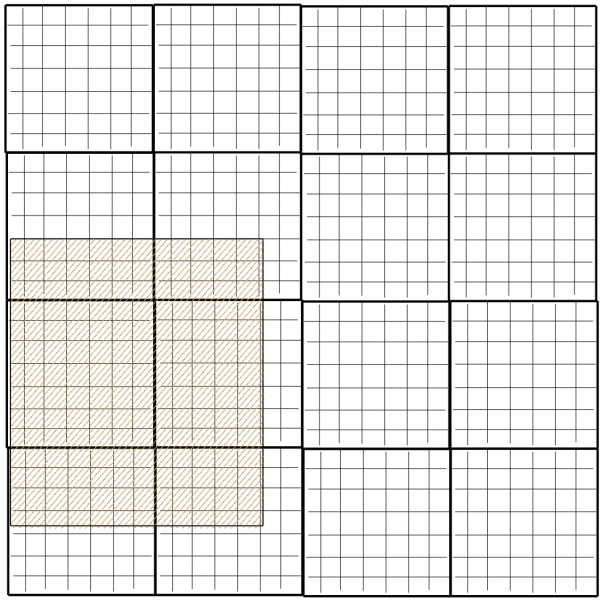
Compress each chunk

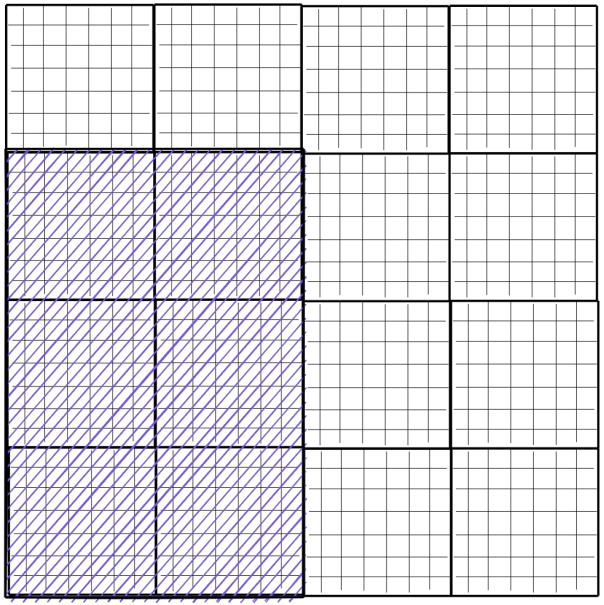


Credit: @trevmanz

How Zarr works?

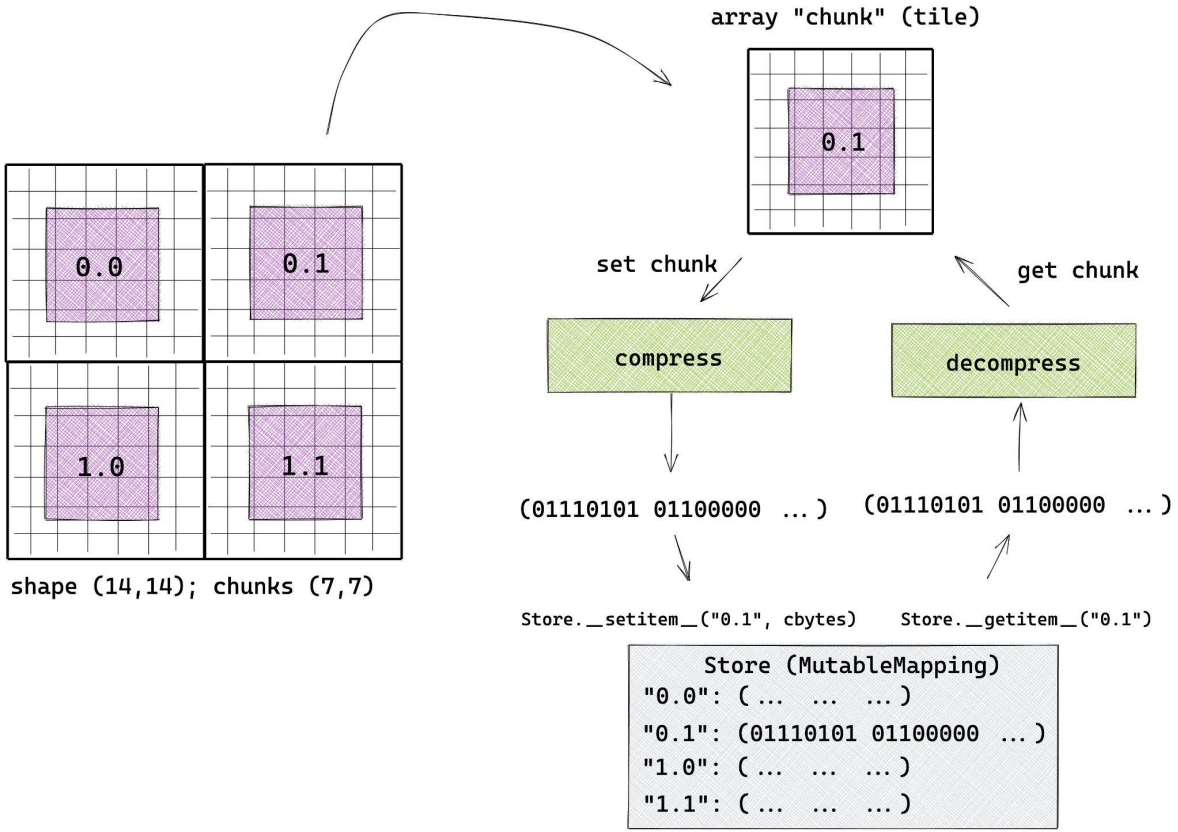
Retrieve chunks only when needed





Credit: @trevmanz

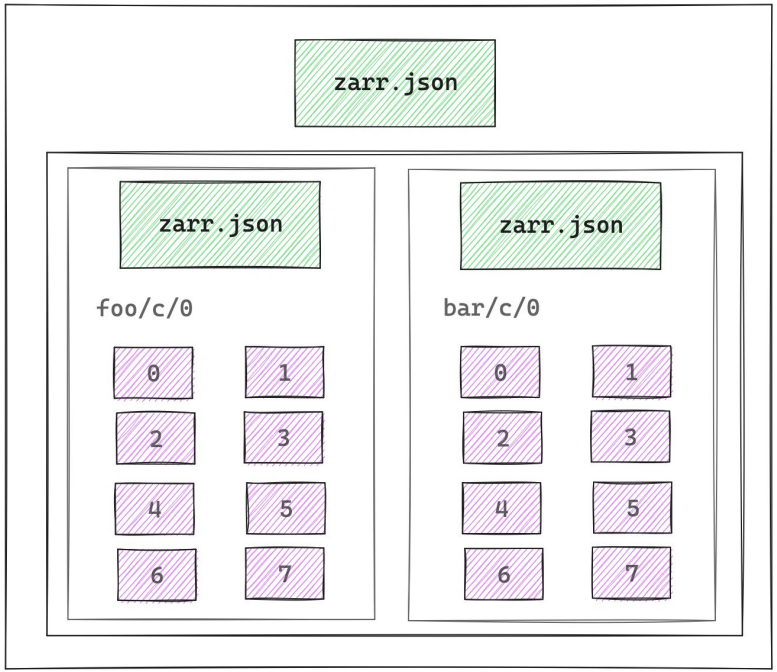
How Zarr works?



Credit: @trevmanz

How Zarr works?

Multiple arrays can be organised in hierarchies of groups

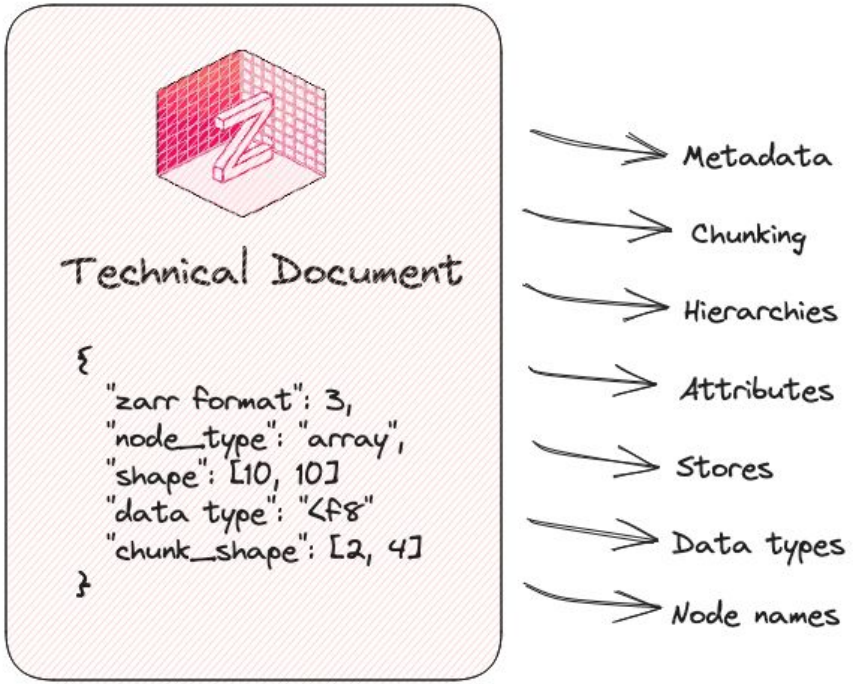


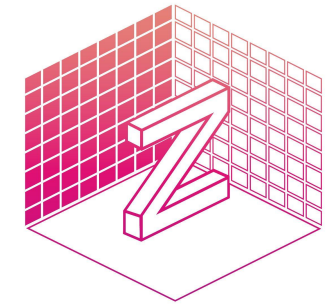
Zarr Specification

V1 → V2 → V3

https://zarr-specs.readthedocs.io/

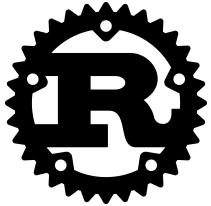
Zarr Specification





Zarr

↔

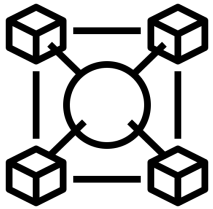


julia

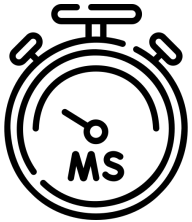
JS

What's new in Zarr-Python 3?

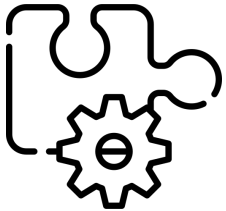
The Motivation



Interoperability



High-latency storage



Extensibility

Installation

```
$ pip install --upgrade zarr
```

or

```
$ conda install --channel conda-forge zarr
```

Installation

```
$ pip install “zarr[<extra>]”
```

where <extra> is:

gpu—support for GPUs

remote—reading/writing to remote data stores

Major design updates

New ABC Store (Abstract Base Class)

ABC Store

→ Replaces MutableMapping base class

→ Allows extensibility to add new custom stores

→ Implements LocalStore, ZipStore and FSStore

→ Removed various exotic stores like SQLite, MongoDB, Redis, DBMS, LMDBS Stores

Custom Codecs via Entrypoint Mechanism

Custom Codecs

→ Enable flexible data compression and encoding strategies

→ Zarr includes Blosc, Zstandard and Gzip compressors only

→ Numcodecs has been adapted to use Zarr’s Codec entrypoint system

GPU support for Zarr arrays

Using GPUs with Zarr

→ zarr.config.enable_gpu() configures Zarr to use GPU memory

→ Zarr currently enable reading ndarray into GPU memory. Encoding and decoding still uses CPU memory

Changes and deprecations

Changes and deprecations

→ zarr.create_array() instead of zarr.array()

→ zarr.create_group() instead of zarr.group()

→ from zarr.storage import LocalStore instead of

  from zarr import DirectoryStore

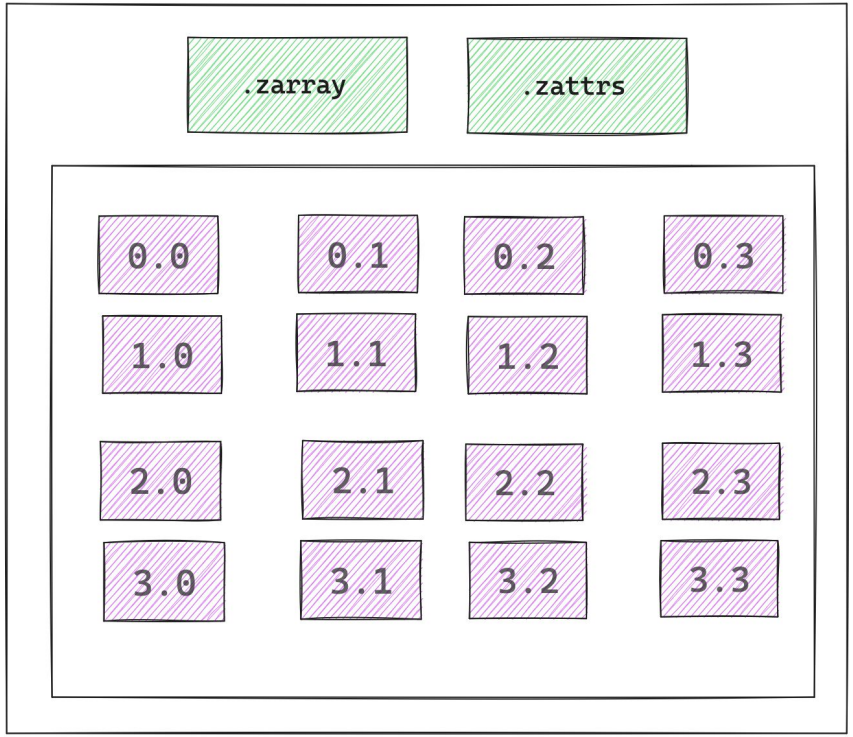
Changes and deprecations

- `zarr.attrs` has gone, with no replacement
- `zarr.codecs` has gone, use `numcodecs` instead
- `zarr.context` has gone, with no replacement
- `zarr.core` remains but should be considered private API
- `zarr.hierarchy` has gone, with no replacement (use `zarr.Group` inplace of `zarr.hierarchy.Group`)
- `zarr.indexing` has gone, with no replacement
- `zarr.meta` has gone, with no replacement
- `zarr.meta_v1` has gone, with no replacement
- `zarr.sync` has gone, with no replacement
- `zarr.types` has gone, with no replacement
- `zarr.util` has gone, with no replacement
- `zarr.n5` has gone, see below for an alternative N5 options

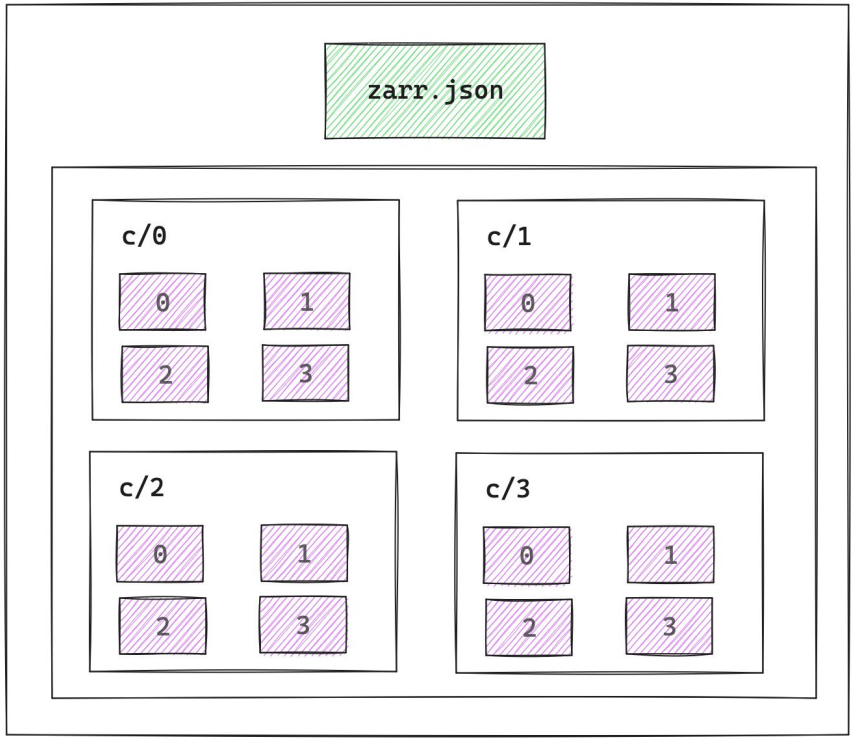
Migration Guide

https://zarr.readthedocs.io/en/stable/user-guide/v3_migration.html

Zarr Arrays → V2 vs. V3

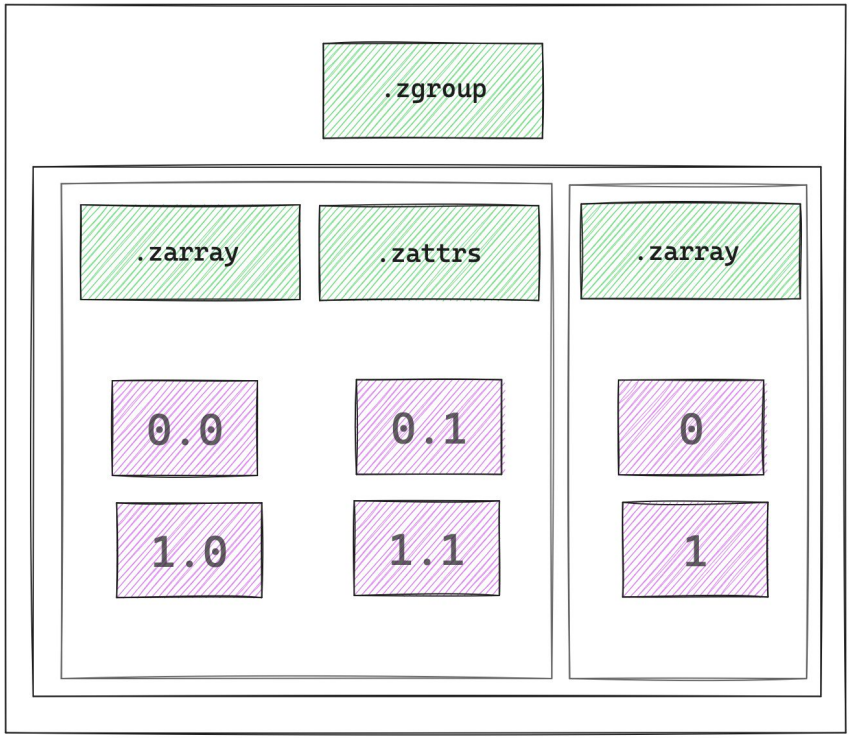


V2

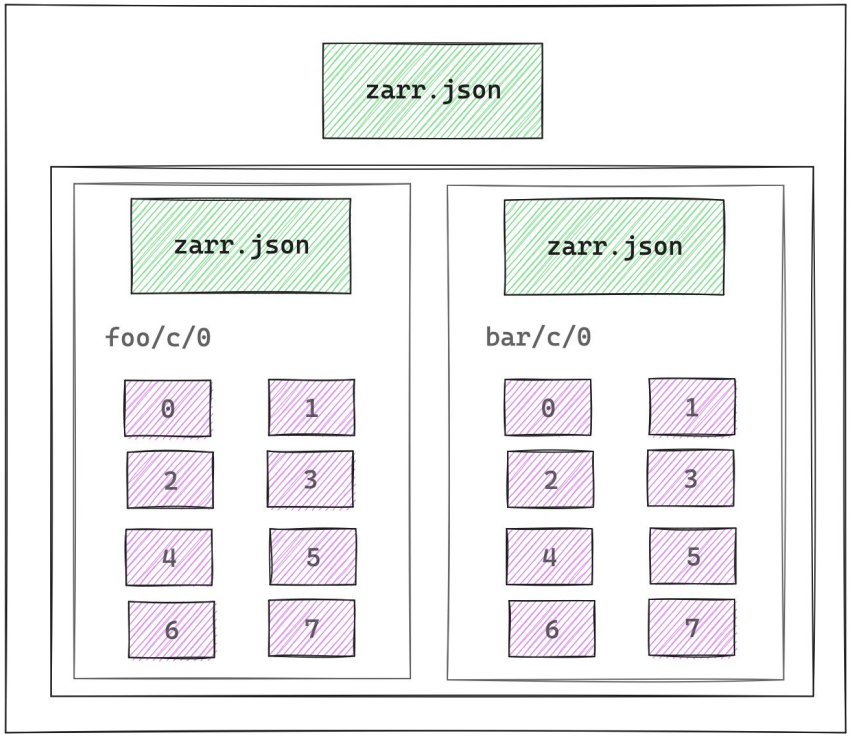


V3

Zarr Groups → V2 vs. V3



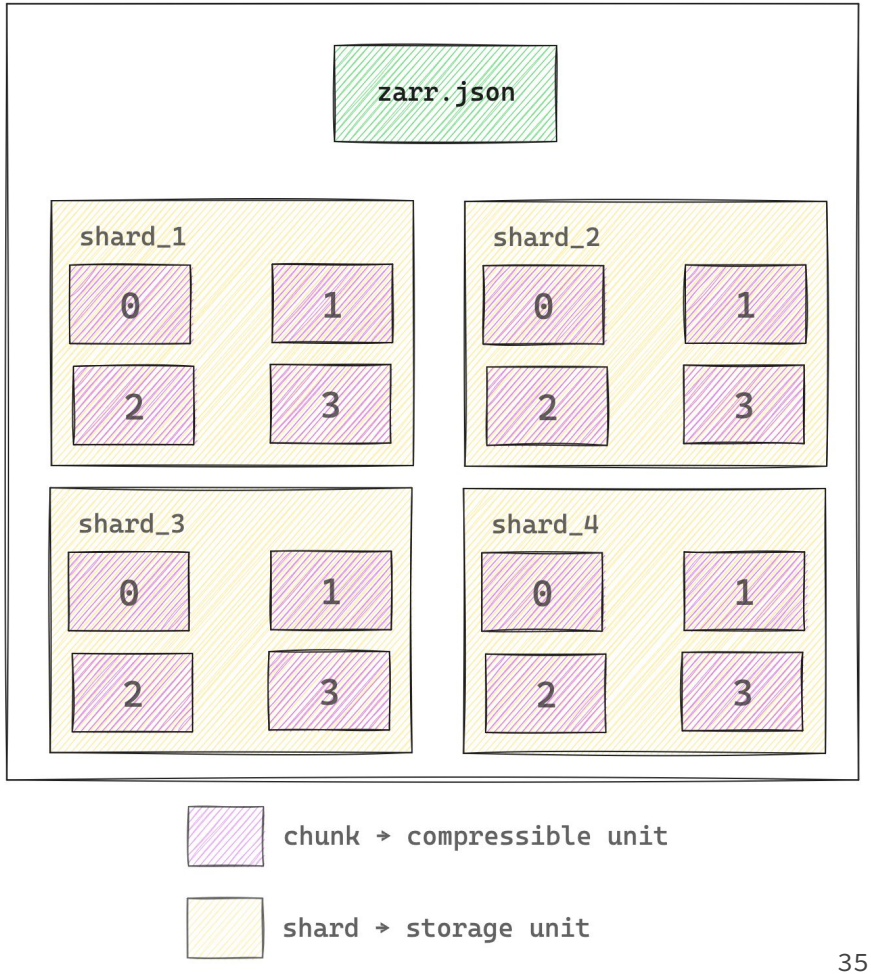
V2



V3

Sharding Codec

Via Extension Mechanism



Extensions

zarrs

A *Rust* library for the Zarr storage format

GitHub | Rust API | Python API | The zarrs Book | Benchmarks | CLI Tools

zarrs-python

pypi v0.1.3 | downloads/month 2k | downloads 22k | stars 34 | ci passing | cd passing

This project serves as a bridge between **`zarrs`** (Rust) and **`zarr`** (`zarr-python`) via **`PyO3`**. The main goal of the project is to speed up i/o (see **`zarr_benchmarks`**).

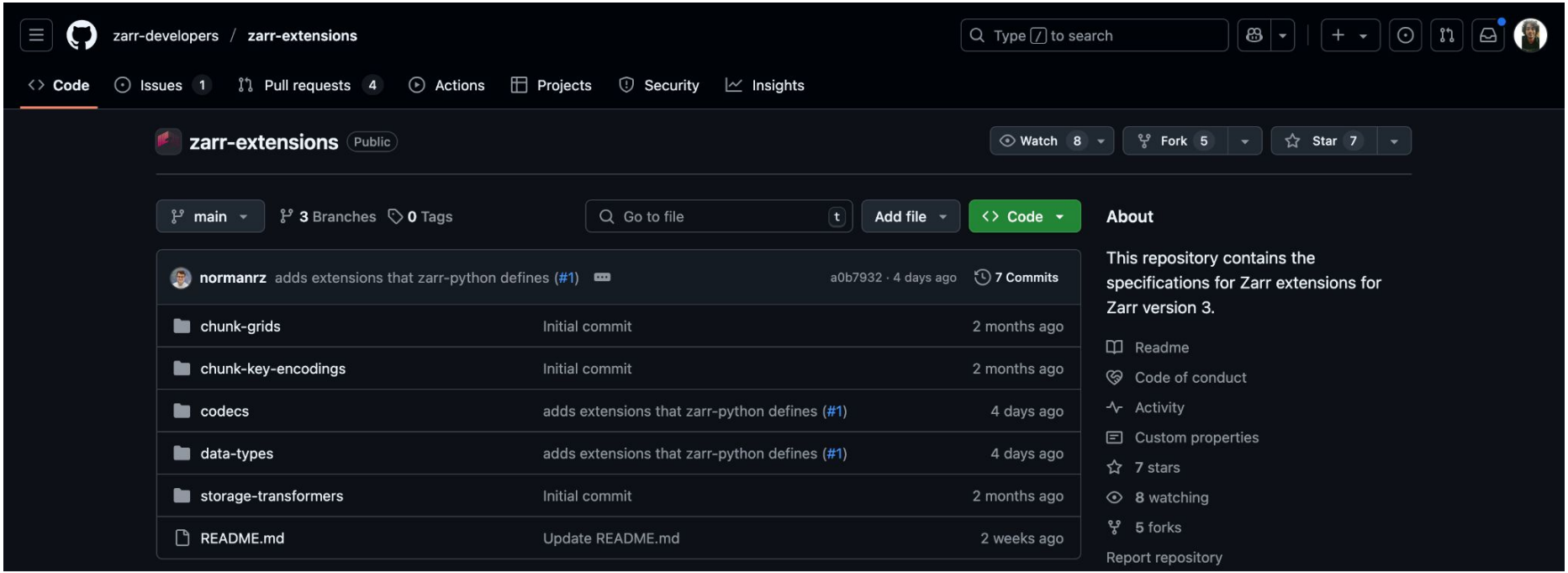
To use the project, simply install our package (which depends on `zarr-python>=3.0.0`), and run:

```
import zarr
import zarrs
zarr.config.set({"codec_pipeline.path": "zarrs.ZarrsCodecPipeline"})
```

You can then use your `zarr` as normal (with some caveats)!

Credit: @joshmoore

Register your extensions 👇



Demo

How does V3 data looks like?

WEBKNOSSOS

Citation:

Motta et al., Science 2019

Maintained by: /normanrz

(Norman Rzepka)

https://webknossos.org/links/NxMfC4B65CmTVGhr

/neuroglancer

https://github.com/google/neuroglancer

Maintained by: /jbms

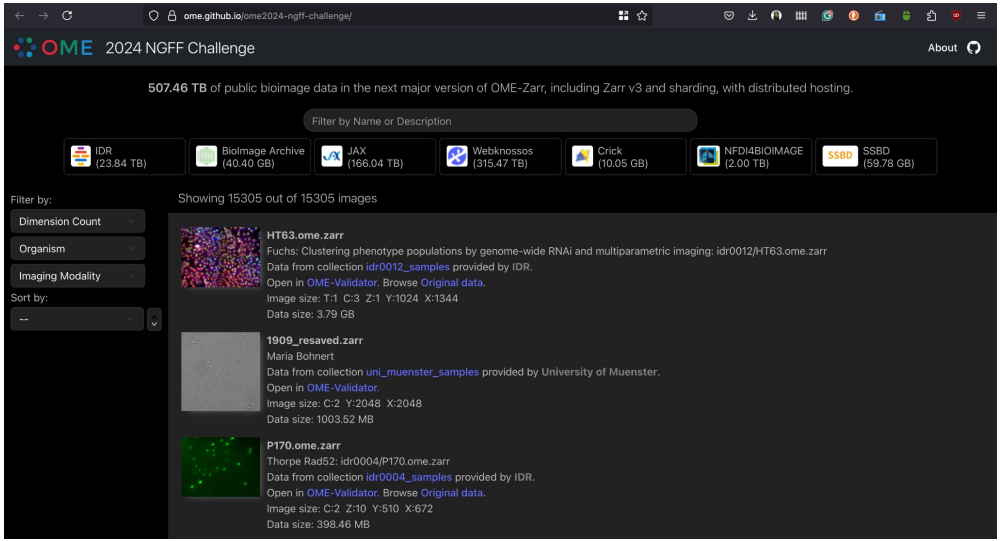
(Jeremy Maitin-Shephard)

Neuroglancer visualiser

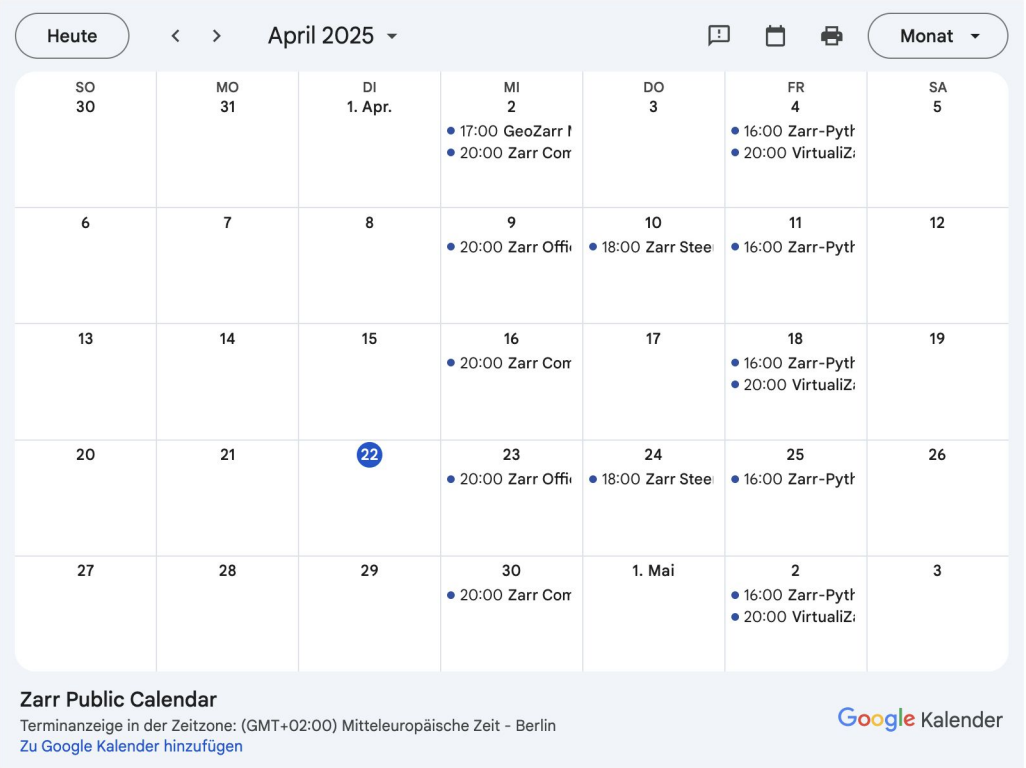
https://ome.github.io/ome2024-ngff-challenge/

Maintained by: /joshmoore

(Josh Moore)



Join us at community, ZEP, core-dev meetings or office hours!



https://zarr.dev/community-calls/

Thank you! 🙏

https://zarr.dev/