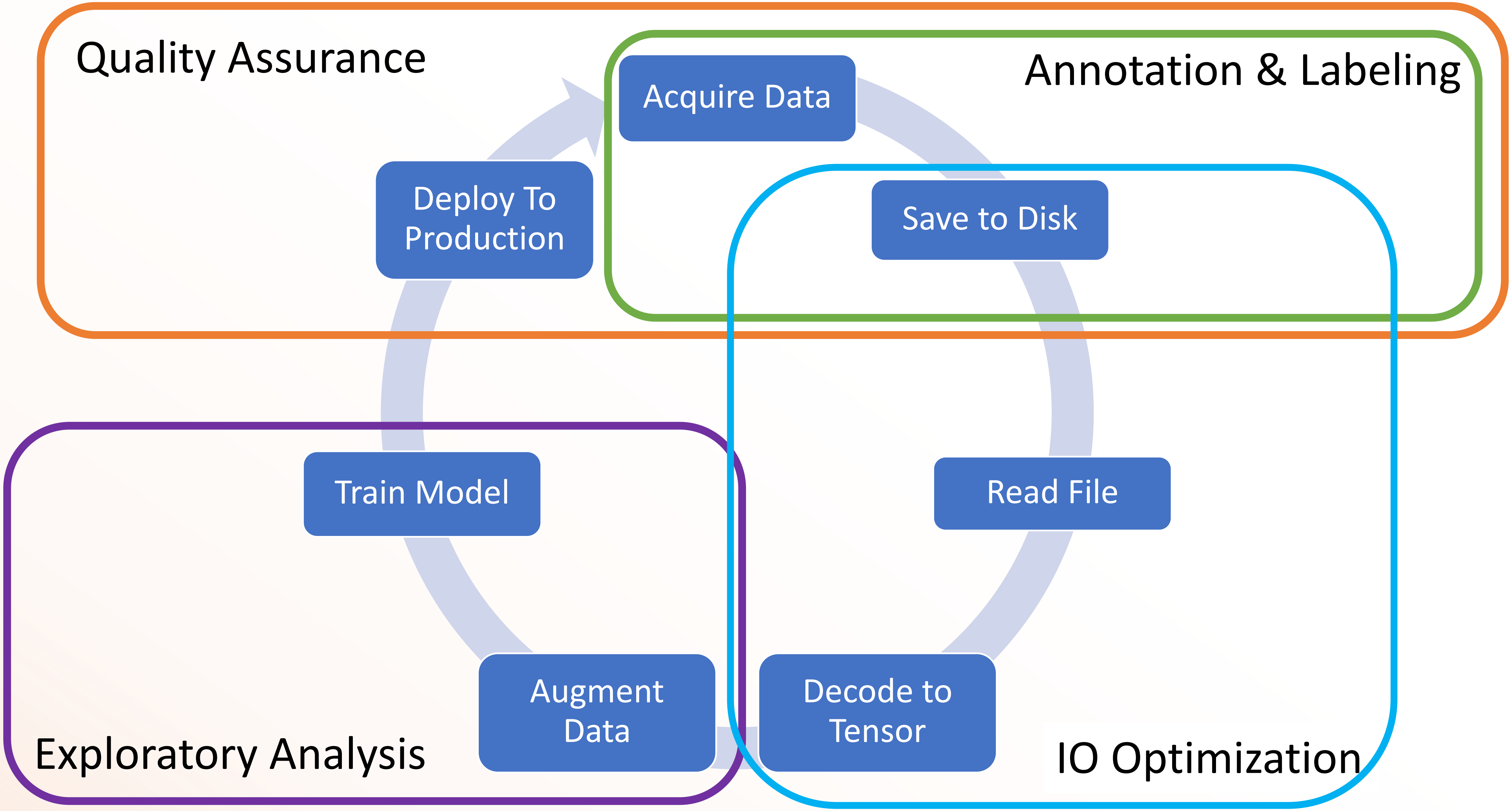
Hangar

Git For Your Data

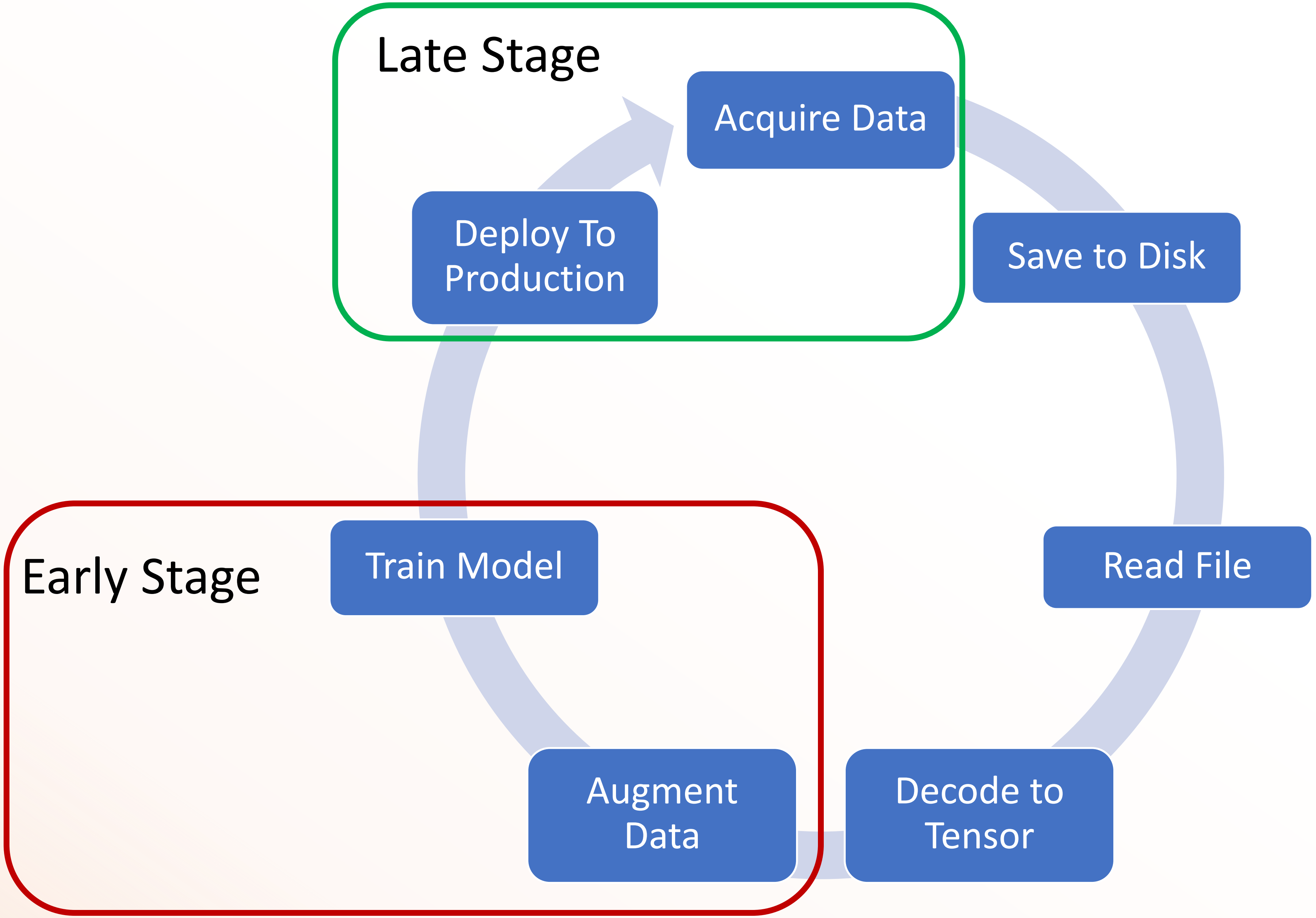
Why hasn’t the open-source ethos translated to datasets?

What is holding back open-source dataset curation?

Data Workflow



Data Workflow



Late Stage

Acquire Data

Save to Disk

Read File

Decode to Tensor

Augment Data

Train Model

Deploy To Production

Early Stage

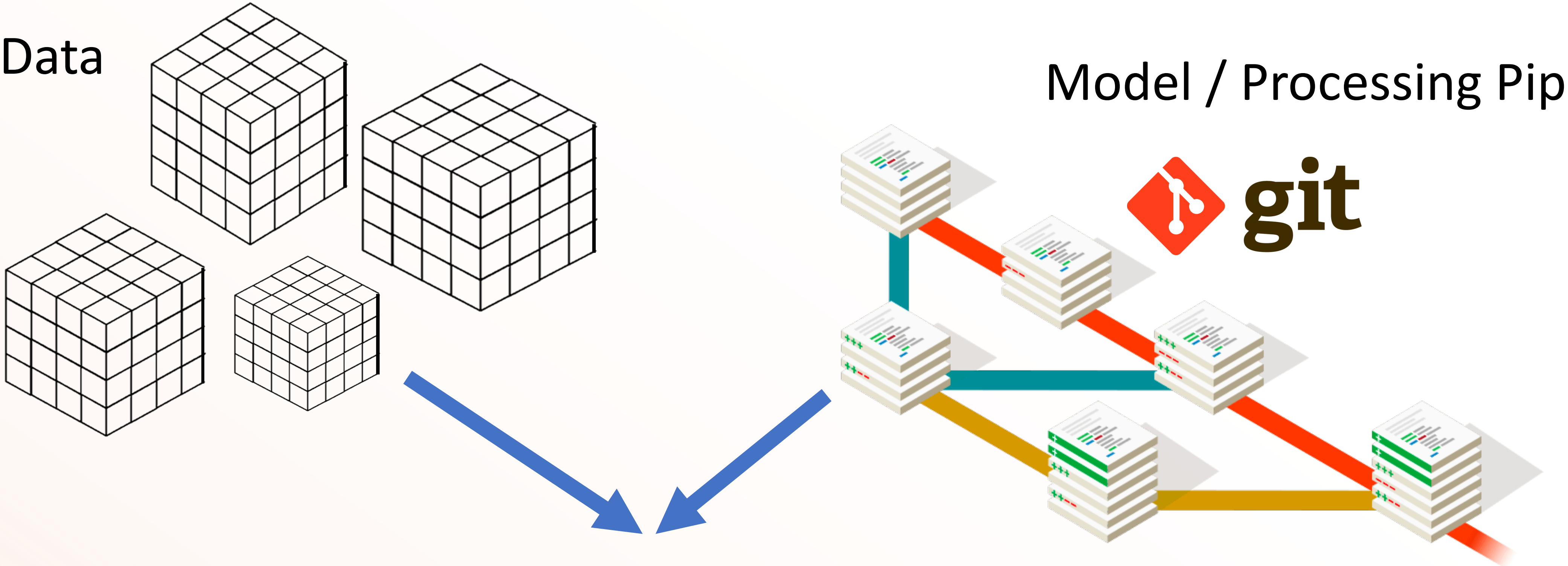
Managing Data on Disk is a Pain

- Each are designed to solve a particular problem.
- Each has limitations
- Each can be very complex to setup effectively

ZIP
HDF
[tile]DB
netCDF
Zarr.
SciDB
python
TFRecord
NumPy
PyTables
CSV
DICOM
Digital Imaging and Communications in Medicine
protobuf
Protocol Buffers

Creating a system to track data history is a chore…

It’s not a priority for most people who “use” data



Data

Model / Processing Pipelines

Your Product

How would we build a version control system…

designed for numeric data…

if we started from the ground up?

Design Goals

- Efficiently store n-dimensional arrays.
- Time travel through the history, checkout from any point.
- Ensure integrity of data and history.
- Zero cost branching & merging.
- Built for distribution & collaboration.
- Partially clone / fetch small parts of data from massive dataset.
- Ability to saturate requests from reasonable sized compute clusters.
- Simple to use

Problems We Will Face

Domain Specific Needs

- Large-scale, n-dimensional, dense & sparse arrays

Storage Size Requirements

- No good if size of any dataset repository exceeds individual developer capabilities (mid-grade laptop / workstation)

Data Integrity & Provenance

- How to ensure that Data in == Data out for all of history? No Exceptions.
- How to verify historical record?

Performance Scaling

- Scale from individual laptop to large cluster training DL models.

Collaboration (Distribution)

- How to branch / diff / merge array data?
- Transfer speed limitations (due to data size)

What do we need?

Some way to store data on disk -----------------------> Storage Backends

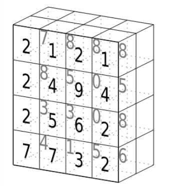
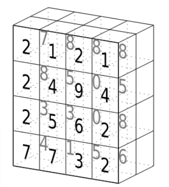
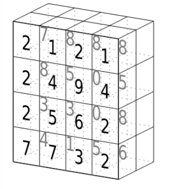
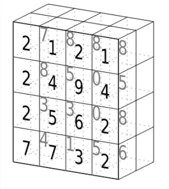
Some way to record records and history ------------> Book-Keeping

Some way to interact with repo -----------------------> API

Hangar Data Model

What is a dataset?

A collection of related sets of data pieces which act to describe some meaningful information

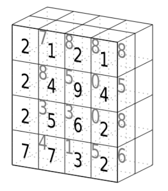
	Image	Bounding-box	Category / Annotation
Sample 1		[[1, 1], [1, 80]], [4, 41], [7, 100]]	[3]
Sample 2		[[2, 1], [9, 82]], [3, 55], [16, 122]]	[1]
Sample 3			
Sample 4		[[2, 4], [2, 85]], [1, 11], [6, 120]]	

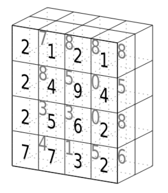
Hangar Data Model

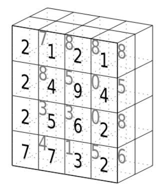
Column

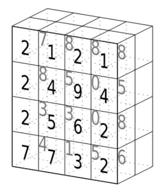
A grouping of samples each describing one component of a dataset

Image









Bounding-box

[[1, 1],   [1, 80]],
[4, 41], [7, 100]]

[[2, 1],   [9, 82]],
[3, 55], [16, 122]]

[[2, 4],   [2, 85]],
[1, 11], [6, 120]]

Category/Annotation

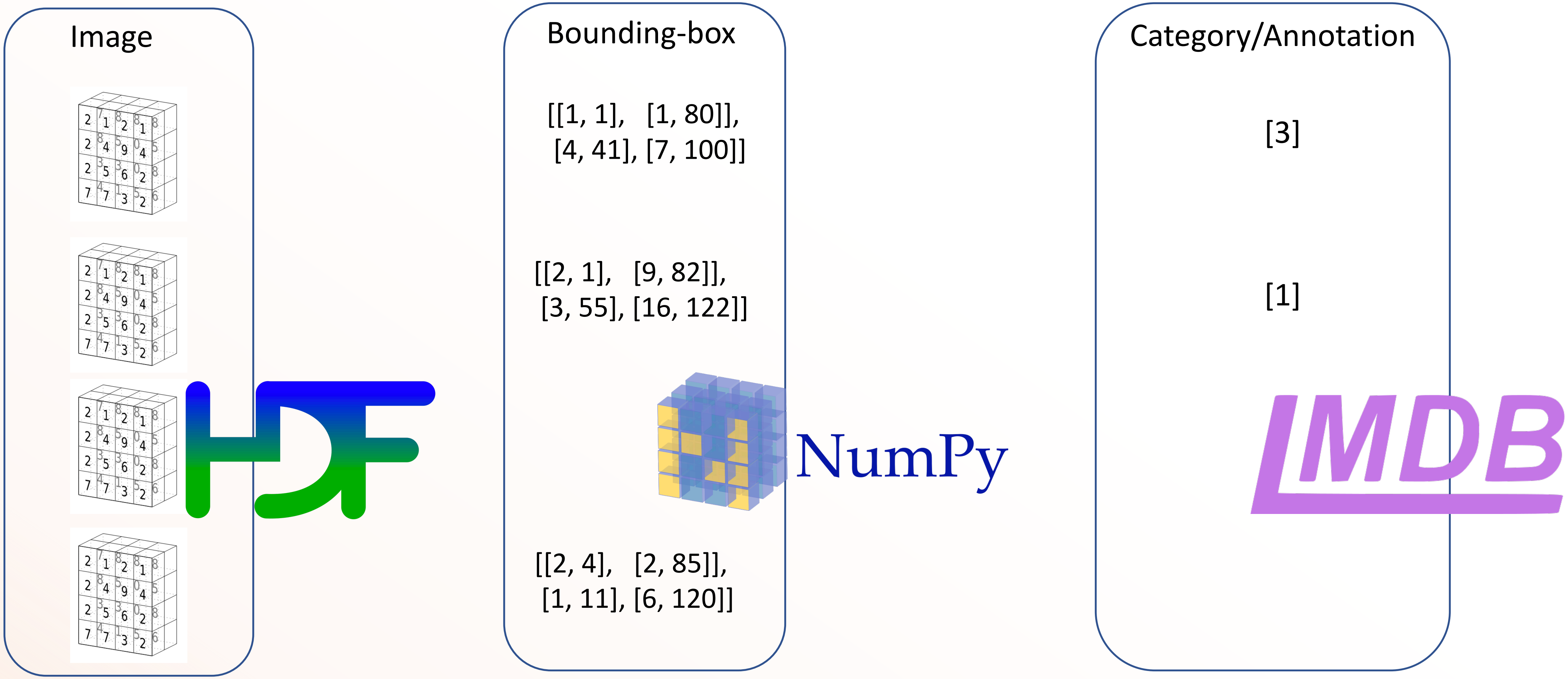
[3]

[1]

Hangar Data Model

Arbitrary Backend Selection

Arrayset stored in backend optimized for data of that particular shape / dtype / layout



Image

Bounding-box

[[1, 1],   [1, 80]],
 [4, 41], [7, 100]]

[[2, 1],   [9, 82]],
 [3, 55], [16, 122]]

[[2, 4],   [2, 85]],
 [1, 11], [6, 120]]

Category/Annotation

[3]

[1]

What do we need?

Some way to store data on disk -----------------------> Storage Backends ✓

Some way to record records and history ------------> Book-Keeping

Some way to interact with repo -----------------------> API

Book-Keeping Highlights

- Git tree like design
- Content Addressable Storage
- Operates orthogonal to data storage layer
- Enables branching / merging
- Cryptographic hashing algorithm ensures repo integrity

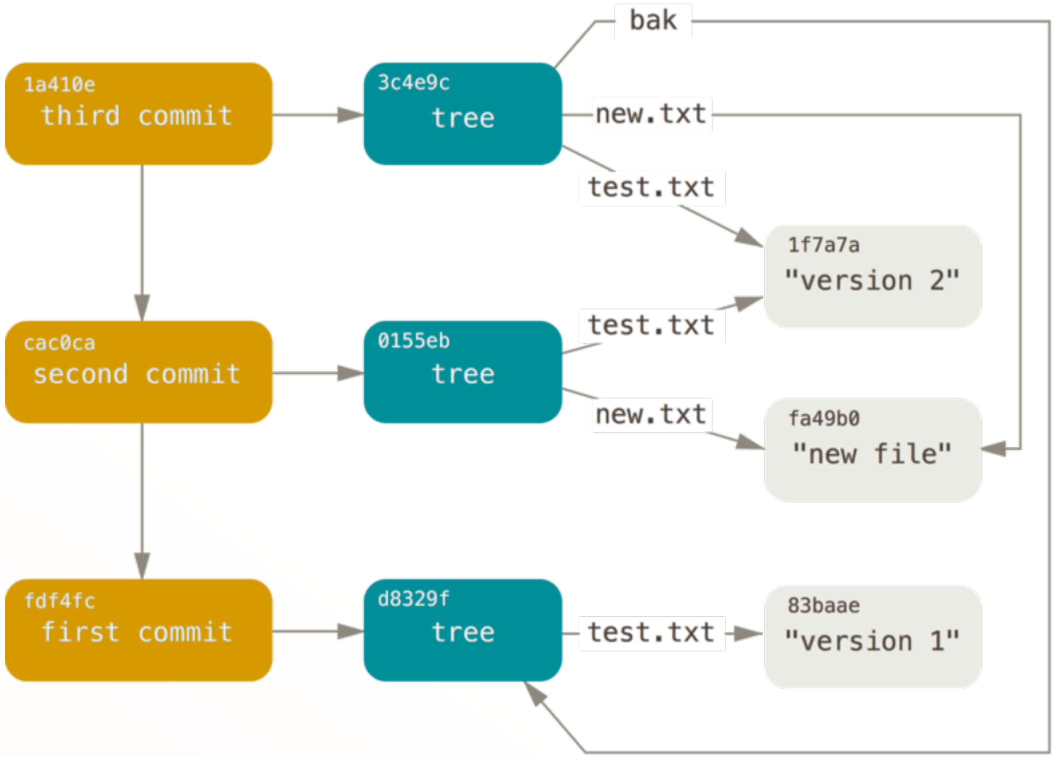


Image Reference: https://git-scm.com/book/en/v2/Git-Internals-Git-Objects

What do we need?

Some way to store data on disk -----------------------> Storage Backends ✓

Some way to record records and history ------------> Book-Keeping ✓

Some way to interact with repo -----------------------> API

Storage Backends

Role:

- Dump and retrieve arrays to disk
- Verify data integrity on read

Scaling Properties:

- Data in Backends is LARGE
- Stored in systems designed for massive data at scale.

Book-Keeping

Role:

- Record data present in each commit
- Track historical log of all commits and parents
- Store backend locating info locating info

Scaling Properties:

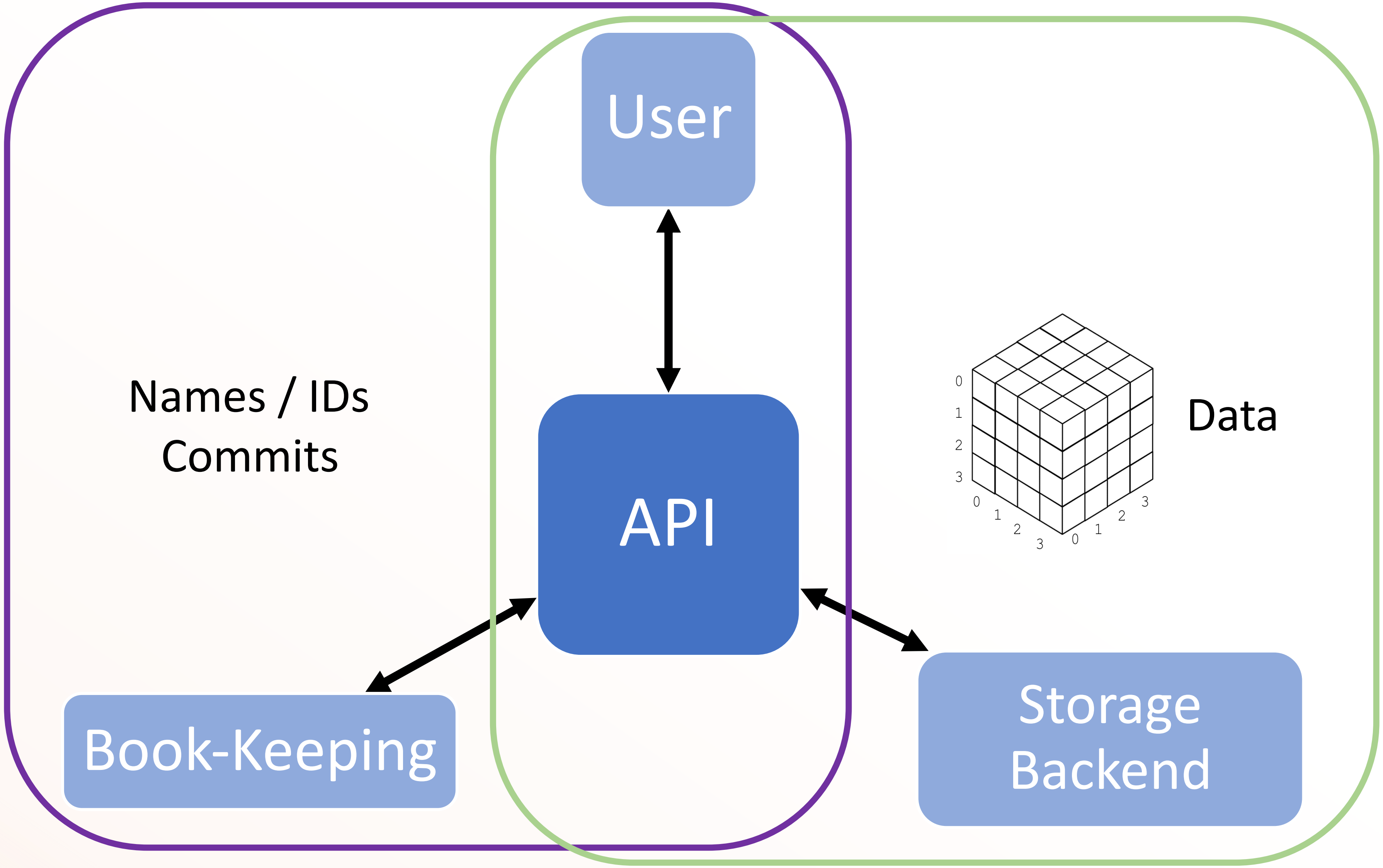
- Records are very small (tens or bytes each)
- LMDB Under the Hood - single level store, extremely fast and low profile lookups.

API

Inspired by Git, but fundamentally different under the hood:

- No working directory.
- Data does will not exist in same exact backend/location on clones of same repo.
- Any number of commits can be checked out in `read-only` mode simultaneously
- Multiple processes can simultaneously checkout and read from the same commit
- All processes (except retrieval) can be performed on content not present locally.
- A single `write-enabled` checkout allowed at a time; is indifferent to read checkouts.

Hangar Execution Model



What do we need?

Some way to store data on disk -----------------------> Storage Backends ✓

Some way to record records and history ------------> Book-Keeping ✓

Some way to interact with repo -----------------------> API ✓

Which Component Solves Which Problem?

	Backend	Book-Keeping	API
Domain Specific Needs - Large-scale, n-dimensional, dense & sparse arrays	✓		
Storage Size Requirements - No good if size of any dataset repository exceeds individual developer capabilities (mid-grade laptop / workstation)		✓	
Data Integrity & Provenance - How to ensure that Data in == Data out for all of history? No Exceptions. - How to verify historical record?	✓	✓	
Performance Scaling - Scale from individual laptop to large cluster training DL models.	✓	✓	✓
Collaboration (Distribution) - How to branch / diff / merge array data? - Transfer speed limitations (due to data size)		✓	✓

Hangar in a nutshell

- Add, branch, merge, time-travel
- Clone, fetch, push
- Scalable: store locally or on the cloud
- No need of materializing all data (partial fetch) Data loaders for major
- DL frameworks Extensible import / export / diff/ viz for data

Demo!

Machine Learning DataLoaders

TensorFlow

```
>>> from hangar import Repository
>>> from hangar import make_tf_dataset
>>> import tensorflow as tf

>>> tf.compat.v1.enable_eager_execution()
>>> repo = Repository('.’)
>>> co = repo.checkout()
>>> data = co.arraysets['mnist_data’]
>>> target = co.arraysets['mnist_target’]

>>> dset = make_tf_dataset([data, target])

>>> dset = dset.batch(512)
>>> for bdata, btarget in tf_dset:
...         print(bdata.shape, btarget.shape)
```

PyTorch

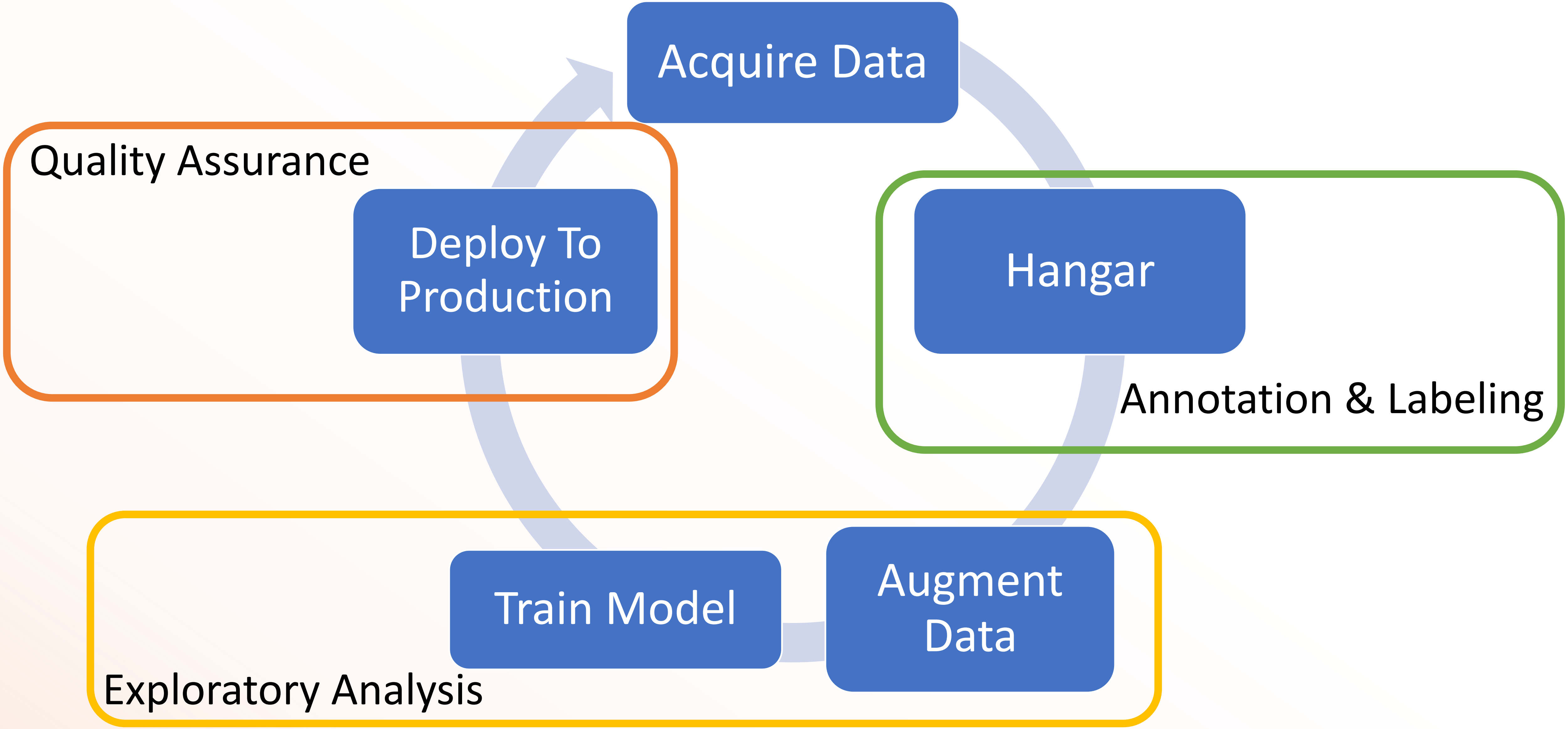
```
>>> from hangar import Repository
>>> from hangar import make_torch_dataset
>>> from torch.utils.data import DataLoader

>>> repo = Repository('.’)
>>> co = repo.checkout()
>>> aset = co.arraysets['dummy_aset’]

>>> dset = make_torch_dataset(aset, index_range=slice(1, 100))

>>> loader = DataLoader(dset, batch_size=16)
>>> for batch in loader:
...         train_model(batch)
```

Hangar Workflow



Acquire Data

Hangar

Augment Data

Train Model

Deploy To Production

Quality Assurance

Annotation & Labeling

Exploratory Analysis

Wrapping Up

State of project

- Core is very solid. Release Candidate Quality
- Comprehensive test suite.
- Growing user base.

Unanswered Questions

- Scaling Limits
    - Extremely fast + small records (10+ million samples tested with no issues)
    - Storage backends designed specifically for this task (data is distributed across multiple files containing collections of samples)
    - Unclear where upper limit is!

How To Contribute?

- Benchmark & Test Breaking Point
  - Have a lot of data?? Get in Touch!
- Increase backend support.
  - Easy to do (3 methods required)
  - Complete documentation already present
  - Automatically tested!
- Performance of remote operations
- CLI Improvements
- Visualization of Diffs
- Hosted example repos

Questions?

Get in touch!

Github: @rlizzo
www.github.com/tensorwerk/hangar-py
Email: hangar.info@tensorwerk.com