BCR

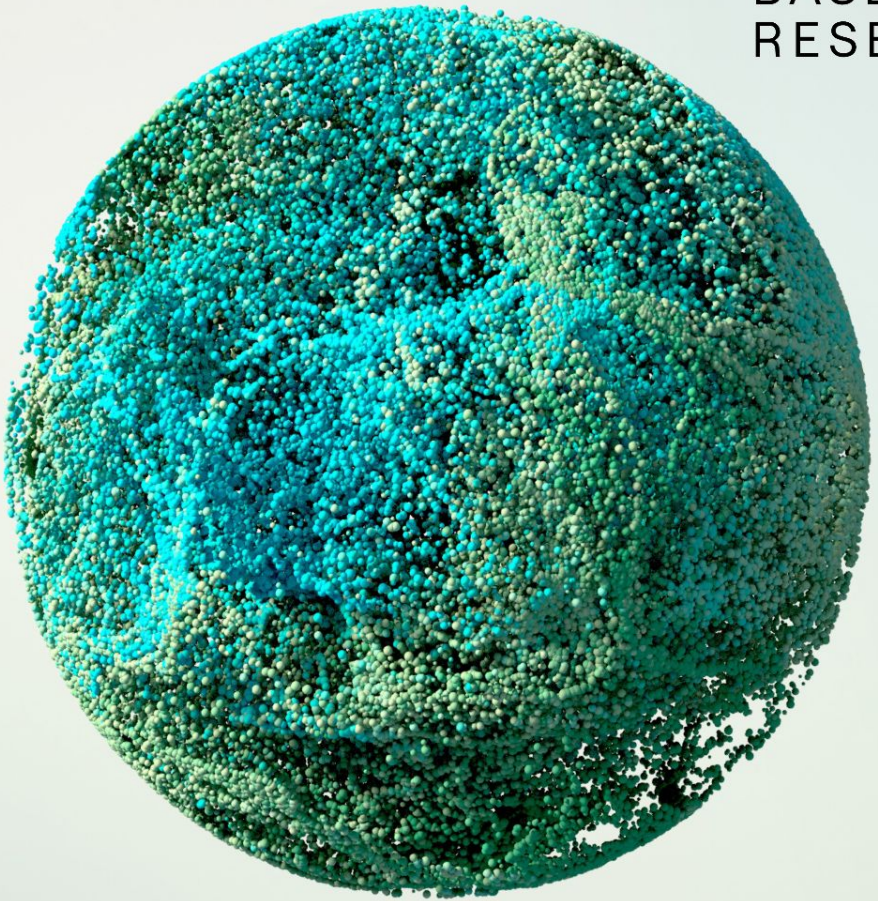
BASECAMP RESEARCH

Keith Kam
Data Product Manager - Engineering

Breaking Biology's Data Wall:

Scaling a Multi-disciplinary Data Platform at Basecamp Research

July 1st, 2025



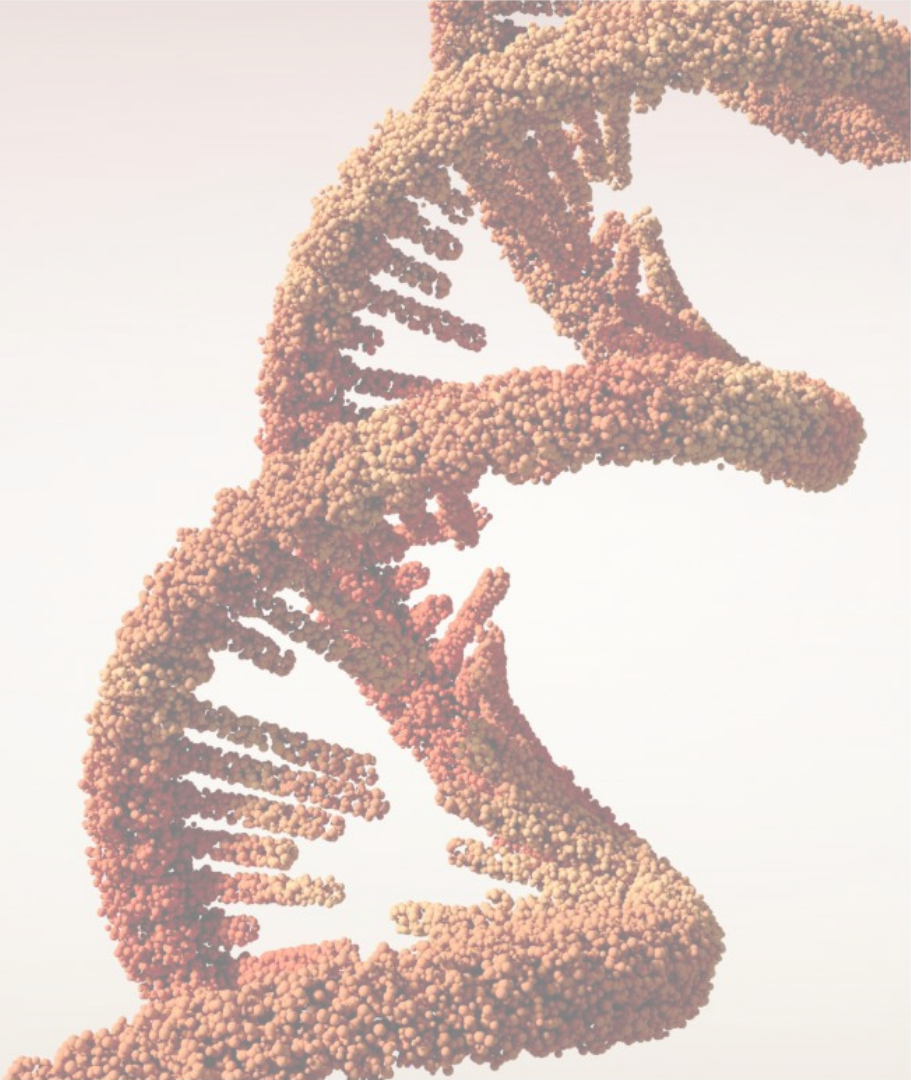
Agenda

01 **Introduction** to BioAI at BCR

02 **Challenges** of a Scaling Data Platform

03 **Strategies** for managing complexity

04 **Q&A**



Part 1/3

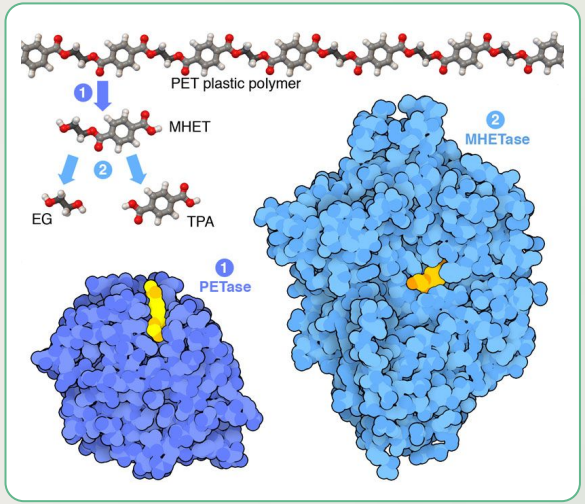
Introduction

Breaking Biology's Data Wall and Powering BioAI at Basecamp Research

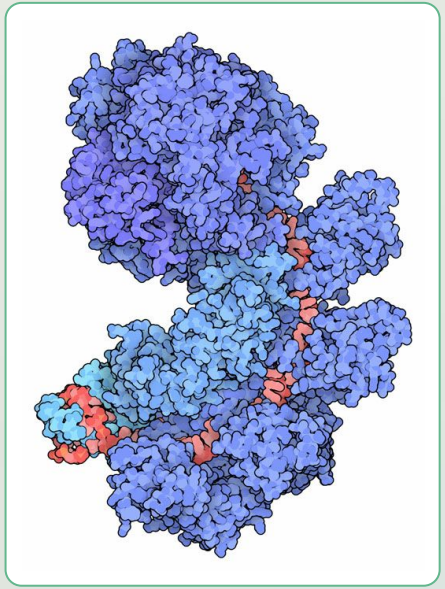
©2025 BASECAMP RESEARCH

Proteins are the building blocks of life

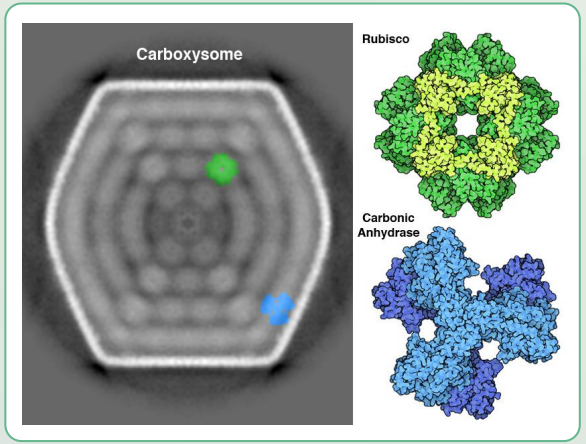
And these nanoscale molecular machines can



Break down plastics



Recognise and cut DNA



Capture carbon dioxide

https://pdb101.rcsb.org/

©2025 BASECAMP RESEARCH

AI is changing how we engineer biology

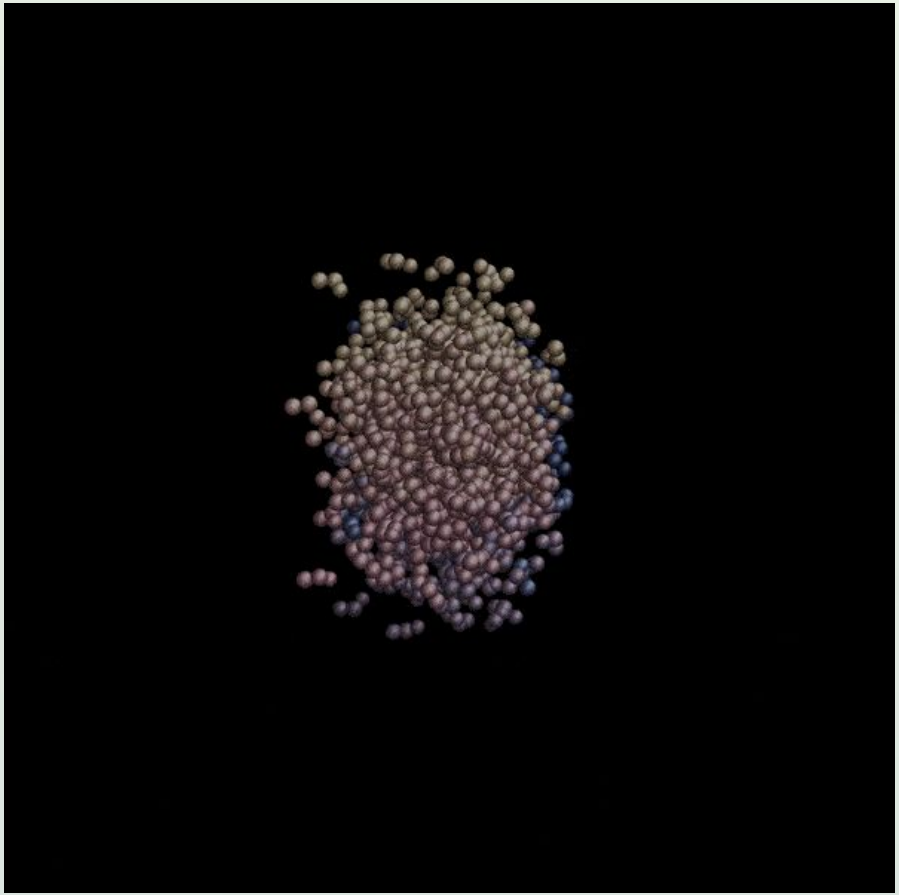
From individual proteins to pathways and therapeutics

💻 Compute

+ 🧮 Algorithms

+ 💽 Data

= ✨ Foundation models

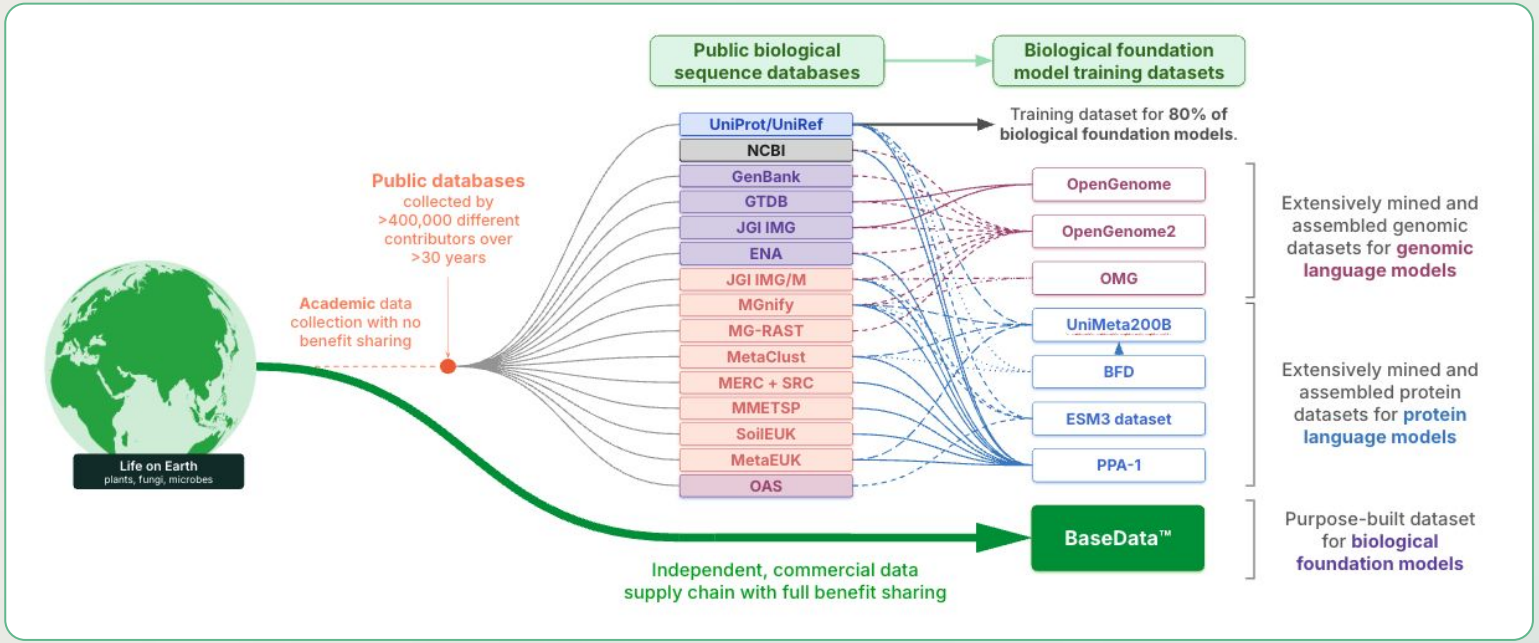


Institute for Protein Design https://www.ipd.uw.edu/software/

©2025 BASECAMP RESEARCH

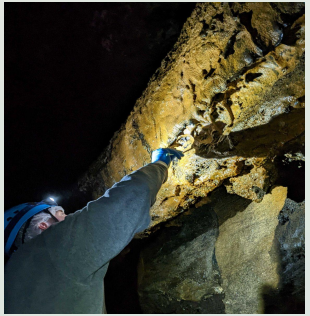
Biology's data wall is holding back BioAI

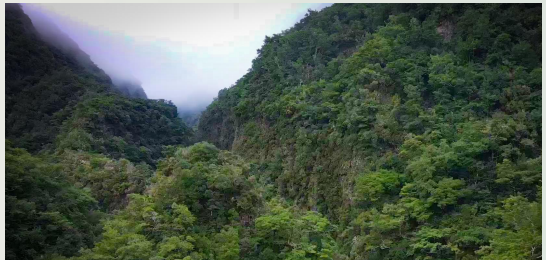
68% of data comes from just 5 species, 70% originates from only 10 countries, and the most-used AI training database is growing by less than 10% per year.



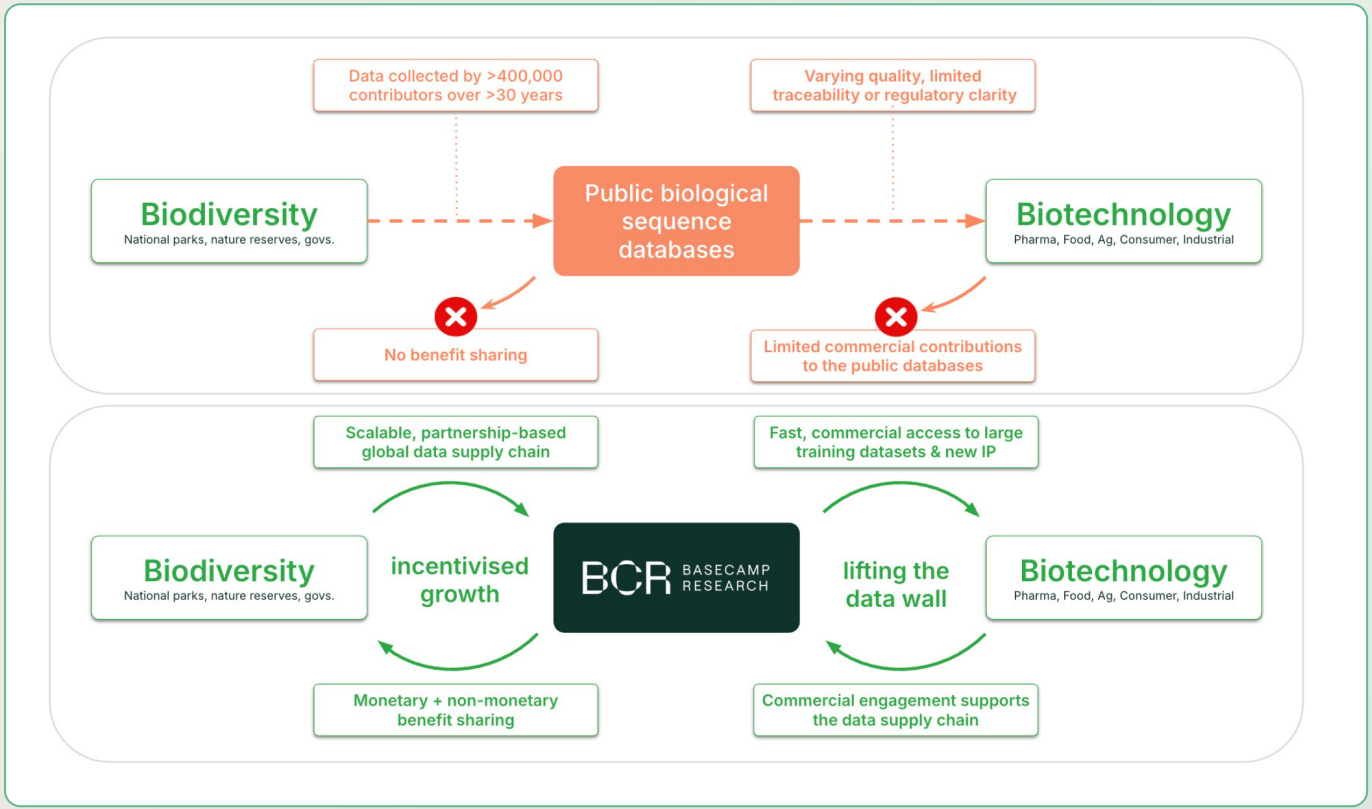
©2025 BASECAMP RESEARCH

Real exploration, real novelty





A data supply chain built upon partnership



©2025 BASECAMP RESEARCH

BaseData™

the world's largest and most diverse biological sequence dataset

400 terabases

of DNA sequencing – that's 400,000,000,000,000 bases

9.8B proteins

>10x larger than comparable protein databases

1M+ new species

Not represented in other genomic databases

Breaking Through Biology's Data Wall: Expanding the Known Tree of Life by Over 10x using a Global Biodiscovery Pipeline https://www.biorxiv.org/content/10.1101/2025.06.11.658620v1

BaseData™ in a nutshell

Global Data Supply Chain for Biological Sequences

A partnership-driven data supply chain across 26 countries and autonomous regions

High Quality, Diverse and Rapidly Growing Data

As of 2024, BaseData™ contains 1 million species not represented in other genomic databases

Fueling the Next Generation of BioAI Foundation Models

We are training biological foundation models in collaboration with NVIDIA AI

©2025 BASECAMP RESEARCH

Part 2/3

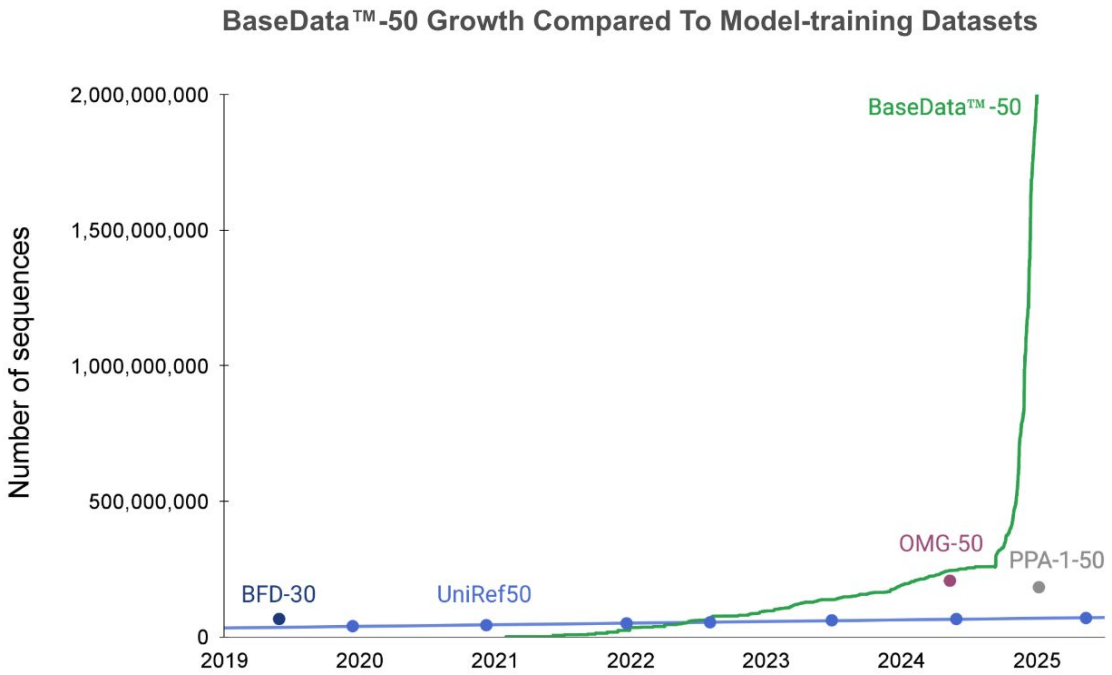
Challenges

Scaling a Heterogenous Data Platform

©2025 BASECAMP RESEARCH

Scaling infrastructure and data

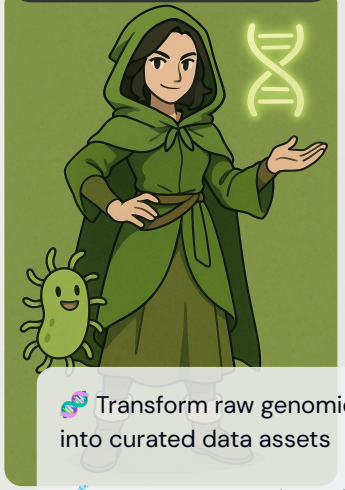
For the Fastest Growing Biological Sequence Database



©2025 BASECAMP RESEARCH

Diverse personas, use cases and tooling

Metagenomic Bioinformatician



🧬 Transform raw genomic data into curated data assets

🧪 Incorporate cutting edge tools into our annotation pipeline

🌟 Develop novel algorithms for specific biodiscovery campaigns

Software & Analytics Engineer

🔧 Engineer high performance, reusable tables

🔍 Create tools for biological information retrieval

🔄 Deploy workflows for large scale ad-hoc pipelines

Deep Learning & Computational Biologist

🗃️ Curate and design datasets for training ML models

💯 Automate internal model inference and evals

✨ Leverage public models for annotation and other workflows

©2025 BASECAMP RESEARCH

Heterogeneity

“

Data platforms are fundamentally heterogeneous: they need to accommodate a diversity of languages, storage systems and data processing and execution technologies under a single roof.

https://dagster.io/blog/what-dagster-believes-about-data-platforms

©2025 BASECAMP RESEARCH

Data Platform Context at Basecamp

Rapidly scaling data and infrastructure

In 2024 BaseData-50 grew by over 10x in a year, with a maximum monthly sequence growth rate of 2 billion novel, high-quality protein sequences.

Rapidly onboarding new tools and pipelines

As of June 2025, a year after migrating our existing primary use case to dagster, we have onboarded 12 additional use cases, covering 400+ assets.

Diverse personas, use cases and tooling

We are building a unified platform across bioinformatics, analytics and ML engineering.

©2025 BASECAMP RESEARCH

Part 3/3

Strategies

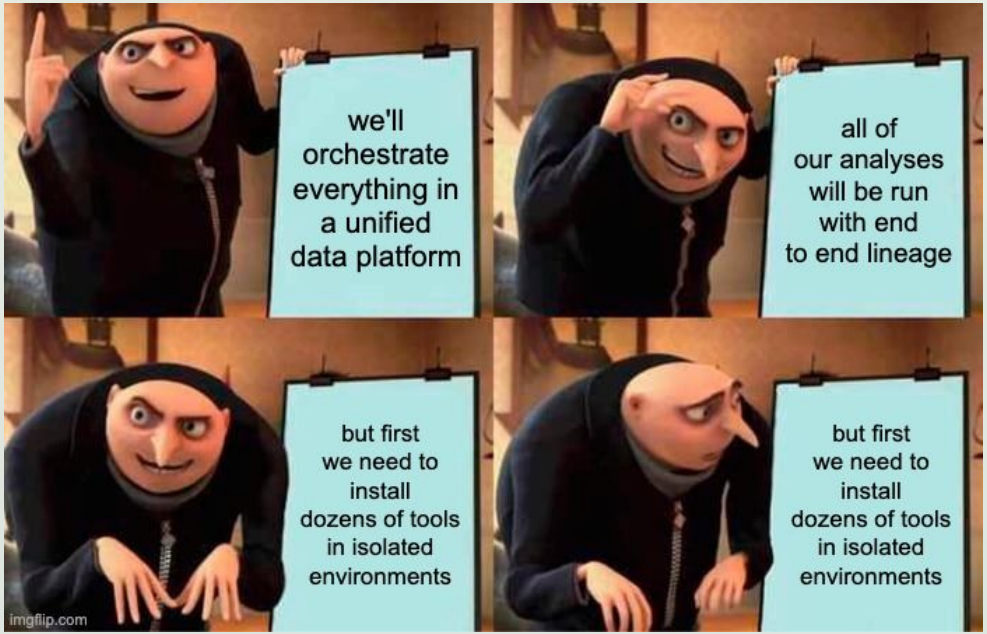
Practical Strategies and Lessons Learnt on Managing Complexity

©2025 BASECAMP RESEARCH

Challenge 1: Tooling Complexity

Beyond uv add

- **Sheer tool volume** dozens of bioinformatics tools, with conflicting dependencies
- **Fragmented channels** across pip/uv, multi-channel Conda, system packages, GitHub forks, and pre-built binaries—no single install path.
- **Long-tail customisation** of tools often require source builds, manual patches, or CUDA driver tweaks



©2025 BASECAMP RESEARCH

Strategy 1a: Containers as Assets

Avoiding a dagster code location sprawl

01

Prebuilt Public containers

Existing predefined containers from public repos

```python
@basedata.asset()
def public_container_example(
    context: dg.AssetExecutionContext,
    container_builder: dg.ResourceParam[BaseContainerBuilder],
) -> dg.Output[ContainerImage]:
    """
    Public container example.
    """
    return container_builder.build(
        context=context,
        image_definition=ExternalContainerDefinition(
            image=ContainerImage(
                repository="public-container-registry/public-repo",
                tag="1.2.3",
            )
        )
    )
```

02

Fast Path for bioconda & Pip

80% of our use cases use this installation pattern

```python
@basedata.asset()
def conda_container_example(
    context: dg.AssetExecutionContext,
    container_builder: dg.ResourceParam[BaseContainerBuilder],
) -> dg.Output[ContainerImage]:
    """
    Conda container example.
    """
    return container_builder.build(
        context=context,
        image_definition=CondaContainerDefinition(
            conda_dependencies=[
                CondaDependency(
                    channel="bioconda",
                    package="example-dependency",
                    version="1.2.3"
                ),
            ],
        ),
    )
```

03

Fully customizable Dockerfile or image

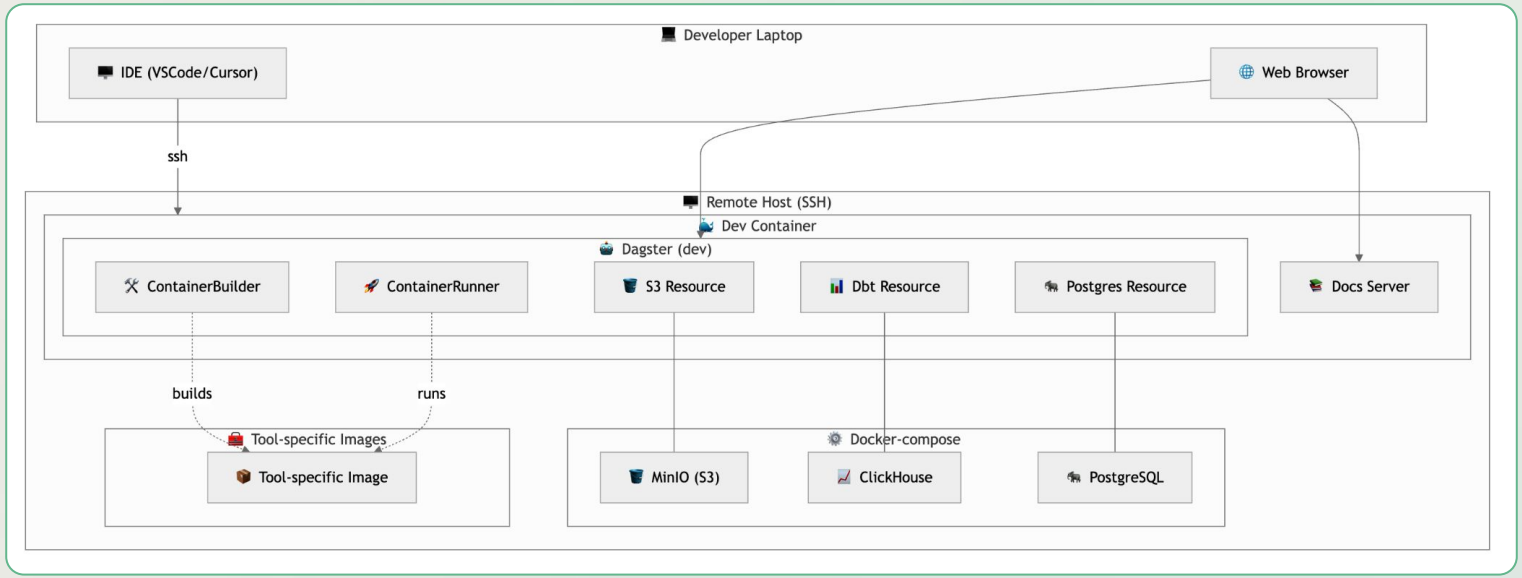
From dockerfile or internal image repo

```python
basedata.asset()
def custom_container_example(
    context: dg.AssetExecutionContext,
    container_builder: dg.ResourceParam[BaseContainerBuilder],
) -> dg.Output[ContainerImage]:
    """
    Custom container example.
    """
    return container_builder.build(
        context=context,
        image_definition=ContainerDefinition(
            image_uri=(
                "1234567890.dkr.ecr.us-east-1.amazonaws.com/"
                "custom-tool:1.2.3"
            ),
        ),
    )
```

©2025 BASECAMP RESEARCH

Strategy 1b: Local Development Setup

Dev containers + ContainerBuilder & ContainerRunner resources



©2025 BASECAMP RESEARCH

Build Once, Run Anywhere

To scientists:
Immutable Container Images, Reproducible Findings

Distribute every Bioconda tool, CUDA build and patched fork as a tagged container image—anyone can docker pull, rerun the exact environment and scale it from laptop to HPC, today or five years from now.

To engineers:
Developer experience counts

Use dev-containers + ContainerBuilder/Runner to spin up identical environments locally, in CI, or in prod. No more “works on my laptop” installs; engineers focus on SQL & Python, not CUDA drivers.

Scientists: e.g. data scientists, product analysts, domain experts
Engineers: e.g. data (platform) engineers, analytics engineers, ML engineers

©2025 BASECAMP RESEARCH

Challenge 2: Pipeline Complexity

Strings all the way down

- **File-format sprawl** dozens of specialised bioinformatics formats, each with its own quirks and edge-cases.
- **Path-based pipelines** CLI tools pass raw paths & directories instead of typed objects, making lineage and refactoring painful.
- **Drifting schemas** TSV/CSV/Parquet outputs mutate silently with every tool version or flag tweak.

©2025 BASECAMP RESEARCH

Strategy 2a: Typing

01
Files
Define specific file type classes

```python
class FnaFile(File): # DNA sequences
    ...

class FaaFile(File): # Protein sequences
    ...

class PdbFile(File): # Protein structure
    ...

class BamFile(File): # Sequences mapped against reference
    ...

class SvgFile(File): # Plots
    ...
```

02
CLI Commands
Specify expected input and output file types

```python
class ExampleToolCommand(CliCommand):
    @property
    def name(self) -> List[str]:
        return ["example-tool"]

    input_file: FnaFile = CliArg(
        flag="-i",
        description="The input file to annotate",
    )
    output_gff_file: GffFile = CliArg(
        flag="-o",
        description="The output GFF file",
    )
    output_faa_file: FaaFile = CliArg(
        flag="-a",
        description="The output FAA file",
    )
    some_other_option: bool = CliArg(
        flag="-s",
        description="Some other option",
    )
```

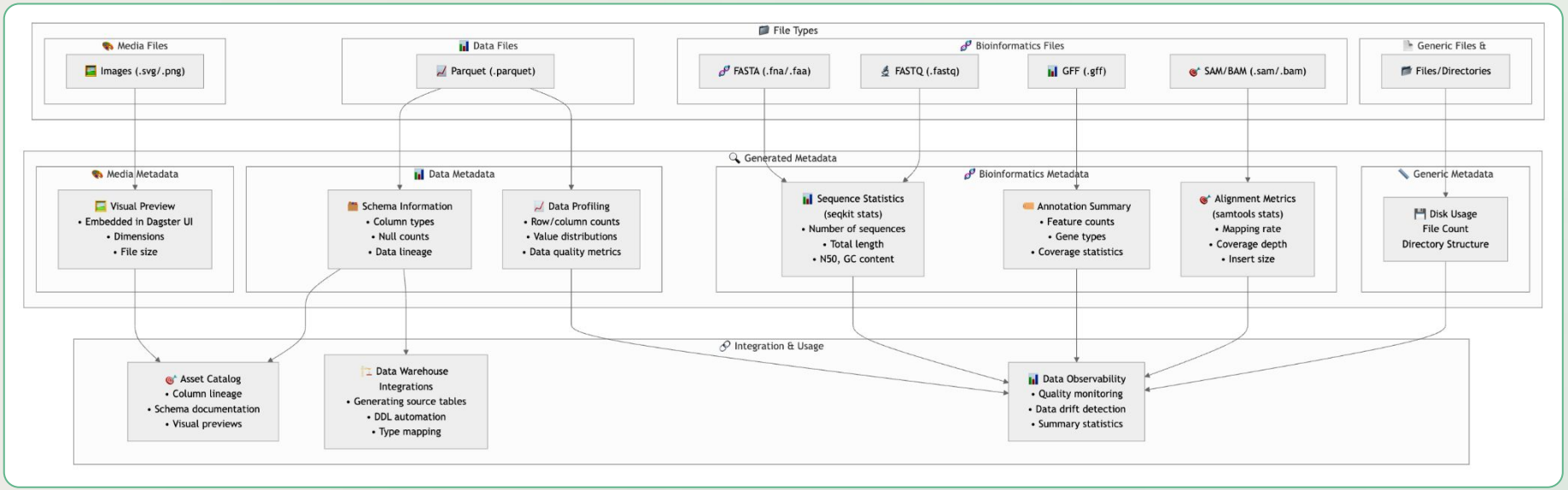
03
Tabular Schema
With parquet, polars and pandera

```python
class ExampleToolParquetFile(ParquetFile):

    def pandera_schema(self) -> DataFrameSchema:

        class ExampleToolParquetFileSchema(DataFrameModel):
            entity_id: str = Field(description="...")
            hit_significance: float = Field(description="...")
            category: str = Field(
                description="...",
                isin=["A", "B", "C"]
            )

        return ExampleToolParquetFileSchema.to_pandera_schema()
```

Types as AI agent context
Typing code allow LSP or type checkers to catch basic code errors, allowing agents to be autonomous for longer

©2025 BASECAMP RESEARCH

Strategy 2b: Custom Metadata



©2025 BASECAMP RESEARCH

Blueprints-in-Code Accelerate Delivery

To scientists:
Document your intent in code to move faster

Explicit output schemas act as living documentation; moving a little slower today lets a larger team move much faster tomorrow.

To engineers:
Invest in domain specific platform features

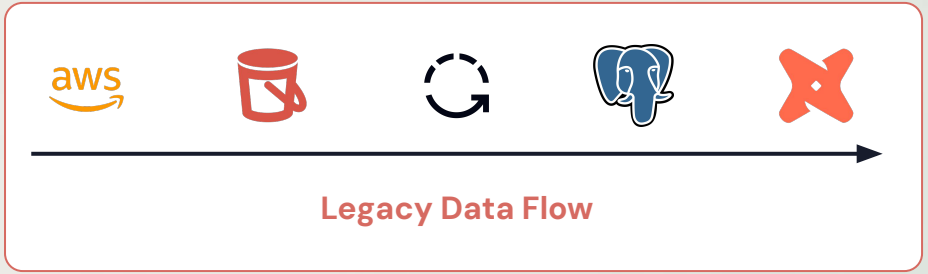
Create domain-specific helpers (typed files, asset sensors, partition wrappers) so devs focus on business code while the platform handles consistency & ops.

Scientists: e.g. data scientists, product analysts, domain experts
Engineers: e.g. data (platform) engineers, analytics engineers, ML engineers

©2025 BASECAMP RESEARCH

Challenge 3: (dbt) Operations Complexity

Siloed Pipelines and the Backfill Bottleneck

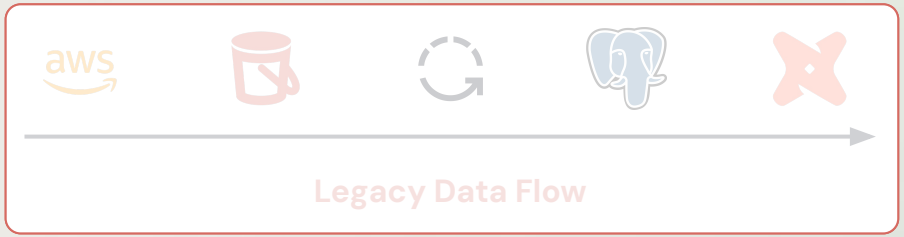


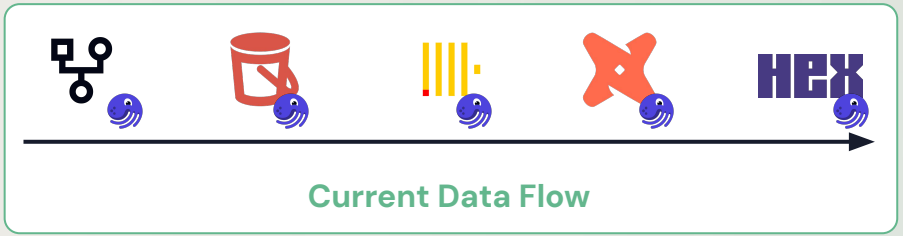
- **Broken Lineage & Observability** across bio + analytics portions of the pipeline
- **Separate manual backfills** needed every time a new asset was added
- **Inefficient Dbt runs** even with incremental macros

©2025 BASECAMP RESEARCH

Strategy 3a: Partitioned DBT

Orchestrating end to end with partitions





Dagster dynamic partitions across all assets - partitioning built in

Partitioned Dbt using dbt vars to define stateful, partitioned dbt assets[1] with upstream lineage

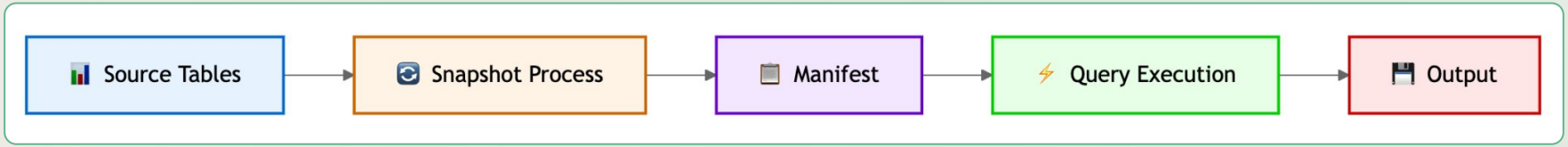
dbt-clickhouse incremental_strategy of insert_overwrite gives us the option to atomically materialise per partition[2]

[1] https://docs.dagster.io/integrations/libraries/dbt/reference#building-incremental-models-using-partitions
[2] https://github.com/ClickHouse/dbt-clickhouse?tab=readme-ov-file#the-insert_overwrite-strategy-experimental

©2025 BASECAMP RESEARCH

Strategy 3b: Snapshot Partitions

Converting from sample to point in time snapshot partition schemes



Our parsed, validated outputs are published to S3 where they are accessed via S3TableEngine.

Snapshot identifies the relevant tables and the point in time state at a per partition basis.

This intermediate gives us a point in time manifest that describes the state for a given table + partition.

With this, we can apply the snapshot query template, filtering on the relevant time snapshots.

Outputs are available as partitioned parquet files (s3 or local) and as a table in clickhouse.

NOTE: there are better options now
We were investigating this before table formats like iceberg had good support in the libraries we use. Use something like that for point in time queries

©2025 BASECAMP RESEARCH

Snapshot

```
GENOMICS_SUMMARY_SNAPSHOT_QUERY = """
WITH annotations AS (
    SELECT
        source_id,
        contig_id,
        COUNT(*) as annotation_count
    FROM src_genomics_annotation
    WHERE _path = '{{src_genomics_annotation.path}}'
    GROUP BY source_id, contig_id
),
sequences AS (
    SELECT
        source_id,
        contig_id,
        length(sequence) as sequence_length,
        gc_content
    FROM src_sequence_metadata
    WHERE _path = '{{src_sequence_metadata.path}}'
)
SELECT
    annotations.source_id,
    annotations.contig_id,
    annotations.annotation_count,
    sequences.sequence_length,
    sequences.gc_content,
    '{{partition_id}}' as partition_id
FROM annotations
JOIN sequences
    ON annotations.source_id = sequences.source_id
    AND annotations.contig_id = sequences.contig_id
"""

```

```
# Create the Dagster asset
@asset(
    partitions_def=MonthlyPartitionsDefinition(start_date="2024-01-01"),
    tags={"asset_type": "snapshot"},
)
def genomics_summary_snapshot(
    context: AssetExecutionContext,
    clickhouse: ResourceParam[ClickhouseResource],
    config: SnapshotConfig,
) -> Output[ClickhouseSnapshot]:
    """Point-in-time snapshot of genomics summary data.

    This asset creates a daily snapshot that joins annotation and sequence
    data, ensuring consistency across all partitions up to the snapshot date.
    """
    date = context.partition_key

    # The snapshot method handles all the complexity:
    # 1. Queries metadata to find valid partitions
    # 2. Generates manifest of files to process
    # 3. Executes queries in parallel per partition
    # 4. Publishes results to S3 and Clickhouse
    snapshot = clickhouse.snapshot(
        context=context,
        code_version="v1",
        query_template=ClickhouseQueryTemplate.from_template(
            template=GENOMICS_SUMMARY_SNAPSHOT_QUERY
        ),
        output_parquet_class=GenomicsSummarySnapshotParquetFile,
        snapshot_table_prefix="sst_genomics_summary",
        date=date,
    )

    return Output(snapshot)
```

©2025 BASECAMP RESEARCH

Partitioned Pipelines, Faster Findings

To scientists:
Show your science, automate the "last mile".

Consider how your results will be consumed and communicated. Publish tables, dashboards and point-in-time snapshots right after the pipeline so collaborators see impact without waiting.

To engineers:
Think in partitions, not full refreshes.

Dynamic partitions & snapshot assets slice 1-TB Postgres/dbt jobs from "all-day" to minutes and make backfills targeted instead of heroic.

Scientists: e.g. data scientists, product analysts, domain experts
Engineers: e.g. data (platform) engineers, analytics engineers, ML engineers

©2025 BASECAMP RESEARCH

TL;DR

Theme	For scientists	For engineers
Tooling	Immutable Images, Reproducible Findings	Developer experience counts
Pipeline	Document your intent in code to move faster	Invest in domain specific platform features
Operations	Show your science, automate the “last mile”.	Think in partitions, not full refreshes.

BaseData™ is a brand name and technology of Basecamp Research.

BCR BASECAMP RESEARCH

basecamp-research.com

Keith Kam
Data Product Manager

Gus Minto-Cowcher
Lead Platform Engineer

Carla Greco
Sr Bioinformatician

Saif Ur-Rehman
Principal Data Engineer

We’re hiring!

linkedin.com/in/keith-kam-yk/
linkedin.com/company/basecamp-research

Appendix

©2025 BASECAMP RESEARCH

BCR BASECAMP RESEARCH

Breaking Through Biology's Data Wall: Expanding the Known Tree of Life by Over 10x using a Global Biodiscovery Pipeline

Oliver Vince*, Phoebe Oldach, Valerio Pereno, Marcus H Y Leung, Carla Greco, Gus Minto-Cowcher, Saif Ur-Rehman, Keith Y K Kam, William Chow, Emma Bolton, Bupe R Mwambingu, Nadine Greenhalgh, Ineke Knot, Leif Christoffersen, Marlon Clark, Robert Pecoraro, Aaron W Kollasch, Tanggis Bohnuud, Matthew Bakalar, Philipp Lorenz, Glen Gowers

*corresponding author: oliver [at] basecamp-research.com

Short Abstract

Progress of AI in biology is now being limited by the availability of high-quality biological sequence data from nature. To get past this data wall, we introduce BaseData™, a new biological database built on top of a global, scalable biodiscovery pipeline. As of 2024, BaseData had already expanded the known protein universe by over 10x.

©2025 BASECAMP RESEARCH

GEN Biotechnology > Vol. 3, No. 4

Research Article | OPEN ACCESS | | Published Online: 16 August 2024

The Natural Future for AI in Biotech: The Next Generation of Machine Learning Demands Partnership with Biodiversity

Authors: Oliver Vince, Glen Gowers, and Siân McGibbon | AUTHORS INFO & AFFILIATIONS

Publication: GEN Biotechnology • https://doi.org/10.1089/genbio.2024.0018

2,916

Permissions & Citations PDF/EPUB

Abstract

Progress in biotechnology is critically dependent on continued access to new biological “components” (genes, proteins, organisms) from nature. Over recent decades, the way that researchers access and use these components has changed dramatically in response to similarly dramatic developments in technology and regulation. The net effect of these changes has been to severely restrict the availability of high-quality genetic data from biodiversity. This bottleneck limits the potential of machine learning (AI) in biotechnology and is a threat to progress across the industry. We suggest that the inevitable demand for high-quality genetic data to train the next generation of biological AI models has the potential to align the economic and technical interests of the bioeconomy with those of biodiverse “provider” countries and communities. The impending era of big data in biotechnology will therefore require the industry to break its dependence on “digital biopiracy” and embrace sustainable partnership-based data supply chains.

©2025 BASECAMP RESEARCH

Using dbt vars for partitioned assets

```python
@dbt_assets(
        manifest=my_dbt_project.manifest_path,
        partitions_def=DailyPartitionsDefinition(start_date="2023-01-01"),
    )
    def partitionshop_dbt_assets(context: OpExecutionContext, dbt: DbtCliResource):
        start, end = context.partition_time_window

        dbt_vars = {"min_date": start.isoformat(), "max_date": end.isoformat()}
        dbt_build_args = ["build", "--vars", json.dumps(dbt_vars)]

        yield from dbt.cli(dbt_build_args, context=context).stream()
```

©2025 BASECAMP RESEARCH