AI Trading Infrastructure – Master Notion Guide

> Copy–paste the whole page into Notion. Each top-level heading can become a
>
> **Page**
>
> **Toggle**
>
> **Database**

🌐 Overview

- **Mission:** Build a low-latency, contradiction-aware trading infrastructure that unifies *BICEP* (Brownian Inspired Computationally Efficient Parallelization), *Fusion Alpha* (semantic-technical FusionNet pipeline), and the forthcoming *Nimbus Book* (research notebook + paper) into a single, reproducible codebase.
- **Guiding Principles:** Clarity ▸ Modularization ▸ Reproducibility ▸ Robustness ▸ Documentation-first.
- **Quick Links:** README │ run_pipeline.py │ docs/ │ data/raw/.

🗺️ Roadmap & Milestones

(Use a Notion **Timeline** or **Board** database with the following properties.)

- **Milestone** (title)
- **Due Date** (date)
- **Owner** (person)
- **Status** (select: Idea ▸ In Progress ▸ Blocked ▸ Done)

- **Relevant Module** (multi-select: BICEP, Fusion Alpha, Nimbus Book, Infra)

✨ 2025 Q2 Highlights

- **June 15 — BICEP v1.0 kernel passes unit tests**
- **June 30 — Fusion Alpha walk-forward validation complete**
- **July 10 — Nimbus Book initial chapters drafted**
- **July 31 — Internal alpha of live router on Tesla+8-ticker dataset**

🔨 Workspaces & Core Databases

1. **Backlog** – Kanban of every task (Task, Description, Module, Priority, Points, Sprint, Status, Dependency).
2. **Experiments** – Scientific record (Hypothesis, Variant ID, Metrics, Result IMG, Verdict, Merge-candidate?).
3. **Research Library** – Papers/links with tags (Topic, Relevance, Read By, Notes).
4. **Decision Log** – Architecture & trade-off log (Date, Decision, Context, Alternatives, Consequences).

🧠 Module: BICEP

- **Goal:** Ultra-efficient, Brownian-inspired data-parallel kernels for PyTorch & CUDA.
- **Key Files:** src/bicep/kernels/*.cu, train_bicep.py, tests/test_bicep.py.
- **Implementation Checklist:**
    - Finalise random walk bit-mapping logic
    - GPU parallelisation benchmarks (RTX 4090 vs A100)
    - Integrate into FusionNet as drop-in replacement
    - Document theoretical derivation (Nimbus Book §2)
- **Validation Metrics:** TFLOPS utilisation, epoch time, loss parity with baseline.

🔗 Module: Fusion Alpha

- **Goal:** Fuse FinBERT sentiment embeddings with technical indicators to detect price–sentiment contradictions and generate alpha.
- **Key Scripts:** prepare_dataset.py, train_fusion.py, tune_fusion.py, predict_fusion.py.
- **Pipeline Stages:**
  1. **Data Ingest** – data/raw/*.csv → OHLCV + news scrape
  2. **Feature Build** – technical_features, finbert_embeddings
  3. **FusionNet Training** – Multi-head attention + contradiction severity
  4. **Evaluation** – k-fold CV, walk-forward, benchmark strategies
  5. **Live Router** – live_router.py auto-fetch + inference
- **KPIs:** Sharpe, max drawdown, hit-rate vs benchmark.

📔 Module: Nimbus Book

- **Form:** Jupyter-Book in docs/, exported as static site & PDF.
- **Sections:**
  - Intro & Literature Survey
  - BICEP Theoretical Foundations
  - Fusion Alpha Methodology
  - Experiments & Ablation Studies
  - Future Work & PhD Directions

🖥️ Development Workflow

1. **Environment Setup**

```
conda env create -f environment.yml  # or pip install -r requirements.txt
pre-commit install                   # enforce hooks
```

1.
2. **Branching Model –** main │ dev │ feat/* │ fix/* │ exp/* (no direct commits to main).
3. **Commit Messages** – Conventional Commits (feat(bicep): add cuda kernel #24).
4. **CI/CD** – GitHub Actions: lint → test → build Docker → deploy docs.
5. **Code Review** – 1 approving review + green CI required.

🗄️ Data Management

- **Versioning:** DVC with remote on S3 (dvc remote add -d s3 s3://ai-trading-data).
- **Layers:** raw ➔ interim ➔ processed ➔ features ➔ predictions.
- **Schemas:** store as Parquet w/ schema files in schemas/.
- **Privacy:** Strip PII, encrypt sensitive feeds with age-encryption.

🏗️ Folder Structure Explained

```
.
├── data/             # Datasets & embeddings
│   ├── raw/          # Unmodified source CSVs
│   ├── interim/      # Cleaned but not feature-engineered
│   ├── processed/    # Analysis-ready
│   └── features/     # Numpy/torch arrays
├── models/           # Saved .pth checkpoints & model cards
│   ├── checkpoints/
│   └── experiments/
├── src/              # Library code
│   ├── bicep/
│   ├── fusion_alpha/
│   ├── nimbus_book/
│   └── common/
```

```
├── notebooks/        # Exploratory analysis
├── scripts/          # CLI helpers (e.g., run_pipeline.sh)
├── tests/            # pytest suites
├── docker/           # Dockerfiles & compose
└── docs/             # Auto-generated site (Nimbus Book)
```

Usage Ritual

1. git pull --rebase
2. Activate env → dvc pull → python scripts/run_pipeline.py --stage ingest
3. Develop feature → pytest -q → pre-commit run --all-files.
4. Push PR → code review → merge.
5. Trigger deployment (docs + Docker image tag).

⚙️ CLI Reference

> Tip:

- **Run full pipeline**

  python run_pipeline.py --ticker TSLA --start 2015-01-01 --mode train
- **Hyperparameter tuning**

  python tune_fusion.py --trials 50 --sampler tpe
- **Live route**

  python live_router.py --tickers TSLA AAPL NVDA --interval 1m --log live.log
- **Benchmark**

  python evaluate_strategy.py --baseline buy_and_hold

📅 Daily Logs & Meeting Notes Template

```
### Daily Stand-Up (YYYY-MM-DD)
- **Yesterday:**
- **Today:**
- **Blockers:**
- **Next Milestone:**
```

🗂️ Glossary

Term	Meaning
BICEP	Brownian Inspired Computationally Efficient Parallelisation
Fusion Alpha	Sentiment-Technical fusion network for alpha generation
Nimbus Book	Comprehensive research notebook/documentation
Contradiction Score (CS)	Measure of sentiment-price divergence

🛠️ Quick-Access Checklists

- Conda env recreated after any environment.yml change
- dvc push after adding data artefacts
- Draft decision logged within 24 h of change
- PR merged ≤ 250 LOC unless justified

🔮 Future Enhancements

- Integration with FPGA low-latency order gateway
- Real-time FinBERT inference via ONNX-Runtime
- Switch to duckDB for local columnar analytics
- Deploy on k8s cluster with GPU node autoscaling

> End of Guide – happy building!