

Spark Setup Documentation

Abraham Sharum, Isaac Tunchez, Andrew Appleyard
Class: CS 30003 - Distributed Systems

December 12, 2025

Demonstration Guide

Installation Steps

1. Install Docker and Docker Compose.
2. Clone the repository.
3. Create a root `.env` with at least:
 - `FINNHUB_API_KEY=`
 - `INFLUXDB_TOKEN=`
 - `INFLUXDB_USERNAME=`
 - `INFLUXDB_PASSWORD=`
 - `INFLUXDB_ORG=`
 - `INFLUXDB_BUCKET=`
 - `GRAFANA_USER=`
 - `GRAFANA_PASSWORD=`
4. Ensure these ports are free:
 - 3000, 4000, 7077, 8001, 8080, 8081
 - 8082, 8086, 8800, 35000, 35001

Run the Containers (Exact Commands)

- Build and start: `docker compose up --build`
- Stop: `docker compose down`
- Rebuild a single service (example backend): `docker compose build backend`
- Spark services: master + two workers come up via compose; workers are limited to `SPARK_WORKER_CORES=2` and `SPARK_WORKER_MEMORY=2g`.
- Docker Swarm (optional deployment):
 - Init swarm: `sudo docker swarm init --advertise-addr <your-ip>`
 - Check status: `sudo docker info | grep -A3 "Swarm"`
 - List nodes: `sudo docker node ls`
 - Deploy stack: `sudo ./scripts/deploy_prod.sh`

- List services: `sudo docker stack services distributed`
- Same as above: `sudo docker service ls`
- Inspect service: `sudo docker service ps <service_name>` and logs:
`sudo docker service logs -f <service_name>`
- Remove stack: `sudo docker stack rm distributed`
- Leave swarm: `sudo docker swarm leave --force` (re-run init after leaving)

Run the CRUD Application

- Backend FastAPI: <http://localhost:4000> (Swagger UI at `/docs`).
- Frontend dashboard: <http://localhost:8800> (renders Grafana embed).
- Key endpoints:
 - `GET /portfolio` — list all trades.
 - `GET /balance` — show simulated cash balance.
 - `POST /trades` — create; debits balance (e.g. `{"symbol": "AAPL", "shares": 5, "buy_price": 180}`).
 - `PUT /trades/{symbol}` — update price/shares; adjusts balance (e.g. `{"new_price": 185, "delta_shares": 1, "current_price": 190}`).
 - `DELETE /trades/{symbol}` — delete/sell; optional body `{"sale_price": 190}` to compute net gain and credit balance.
 - `GET /cache` — inspect in-memory price cache (refreshed every 5s for tracked symbols; re-populates after trades are created/updated).
 - `GET /quote/{symbol}` — fetch latest quote via finnhub client; 502 if upstream fails.
 - Price polling (optional): polls `PRICE_SYMBOLS` every `PRICE_POLL_INTERVAL` seconds and writes to Influx `prices` when `INFLUXDB_TOKEN` is set.
- Finnhub quote API: <http://localhost:8001/quote/{symbol}> (uses `FINNHUB_API_KEY`).
- Grafana (optional dashboards): <http://localhost:3000>.

Reproduce the Custom Example

1. Start services: `docker compose up --build`.
2. Create a trade:

```
curl -X POST http://localhost:4000/trades \
-H "Content-Type: application/json" \
-d '{"symbol":"AAPL","shares":5,"buy_price":180}'
```

3. Update it:

```
curl -X PUT http://localhost:4000/trades/AAPL \
-H "Content-Type: application/json" \
-d '{"new_price":185}'
```

4. Delete it:

```
curl -X DELETE http://localhost:4000/trades/AAPL
```

5. Verify via GET /portfolio or refresh the frontend dashboard.