Visual Beam dev with Hop

By Matt Casters
Apache Hop PMC, co-founder
Neo4j Chief Solutions Architect
Program

● Apache Hop introduction
● Demo: Core Apache Hop concepts
● Demo: GCP Dataflow and AWS AKS
● Ongoing and future developments
● Questions
Apache Hop introduction
Data integration bridging the gap

Organizations

Tech / Devs
Concerns of organizations

- Setup costs
- Maintenance costs
- Running costs
- Time to market
- Resource availability & the bus factor
- DevOps
- Solution stability
Concerns of developers

● Ability to succeed
● Have a fun development environment
● Ability to learn new things
● Work with new technology
● Use best development practices
These concerns guide Apache Hop

“Facilitates all aspects of data and metadata orchestration”
Use-cases

- Data integration / Data orchestration / ETL
- Data migration
- Message processing
- Data synchronization
- Master & Metadata Management
- IoT, Big Data, ...
- File handling
- Workflow / BPM
What’s in a name?

- Recursive acronym: Hop Orchestration Platform
- An intuitive and productive toolset for data engineers
- Orchestration:
  - Data: pipelines and workflows
  - Metadata: editing, handling, management,…
  - Insights: data/execution lineage, logging, …
  - Configurations: handling ecosystem complexity
- Platform:
  - GUI, commands, server, scripts, docker, API, documentation, community, …
Apache Hop history

• Community lead initiative
• Starting point was Kettle 8.2 + WebSpoon + patches + plugins + … → Representing 21 years of software development!
• New scalable GUI
• New architecture, metadata back-end
• Simplified toolset
• Code refactored, renamed, trimmed down, …
• Extra plugins: Projects, Testing, Apache Beam, Debugging,
• …
• Years of work!
Apache Software Foundation

- Hop is a Top Level Project at the Apache Software Foundation
- Homepage: http://hop.apache.org
- Source: github.com/apache/hop/
- Building and IT on Apache Jenkins CI
- Released 2.0.0: http://hop.apache.org/download
- Working on 2.1.0
- Fast growing and active community
- Check the website for regular updates & our Hot Hop Hangouts (3Hx)
Why Apache Hop?

- Lower development time and cost
- Lower maintenance time and cost
- Increase transparency
- Improve stability
- Make the learning curve steeper
- Protect against brain-drain
- ...
Metadata abstraction levels

1. Pure code
2. Code templates generating code
3. Metadata generating code
4. Engines executing metadata from ...
   a. Human generated
   b. Metadata templates
   c. Code and other data

Get rid of
- Code generation
- Compilation
- Packaging
- Deployment
Metadata driven architecture

- No code generation, compilation, packaging, deployment cycle
- Execute requirements metadata as is without translation
- Easier to manage, debug, use, ...
- Pluggable execution engines to translate metadata into work
- Predictable outcomes
- Version control friendly
- Platform independent
- ...

Austin, 2022
Metadata sources

Describe tasks, don’t program them!
The description of the tasks, transforms, actions, connections, …
⇒ metadata
This metadata comes from:
● The Hop GUI
● Other data sources
● Programmatically
Metadata execution

Hop metadata can be executed in a variety of ways:

- In the user interface
- Using scripts
- On a remote Hop server
- Embedded in your Java code
- Called as a web-service
- Inside an Apache Spark, Apache Flink or GCP Dataflow cluster
- Inside your scheduler
- With Jenkins, Apache Airflow, …
- In a docker container
- On Kubernetes, docker-compose, …
Guiding principles

We aim to make data orchestration better for organisations and developers:

- **Cheap**: low cost of setup, creation, config, maintenance, ...
- **Easy**: setup, build, maintenance, deployment, ...
- **Fast**: startup time, supporting Spark, Flink & DataFlow, ...
- **Transparent**: before, during and after execution
- **Predictable**: unit and integration testing
- **Innovative**: need for the latest tech (digital transformation)
- **Supporting best practices**: support version control, testing, CI/CD, projects, lifecycle management, ...
Key features

- License: Apache Public License v2.0
- Metadata driven: no code generation
- Modular pluggable architecture: scale back to <30MB
- Fast startup, minimal overhead
- Apache Beam with support for Apache Spark, Apache Flink and GCP Dataflow runners
- Version controlled documentation
- Ease of use: GUI, transparent naming and easy to use tools
- Integration tests: critical components are tested daily with integration tests
  → runtime compatibility, stability, ...
Key GUI features

- Pluggable GUI features
- Scalable interface for high DPI displays or visually impaired
- Perspectives for easy fast context switching
- Designed for web browsers and mobile users
- Single click mode for faster navigation
- 4 platforms: Windows, OSX, Linux & Web
- “dark mode” supported on all platforms
Hop GUI in a browser

docker run \
  --publish 8080:8080 \
  --env HOP_WEB_THEME=dark \
apache/hop-web

Then: browse to http://localhost:8080
Demo: core concepts
Core Apache Hop concepts

- Website: https://hop.apache.org
- Download: https://hop.apache.org/download/
- Tools overview
- Hop GUI
- Pipelines and transforms
- Workflows and actions
Demo: Dataflow and AKS
Beam pipelines

- Getting started with Apache Beam
- Unit testing
- Samples
- How to run: pipeline run configurations
- Run a pipeline using GCP Dataflow
- Run a pipeline using Apache Flink on AKS (k8s)
Ongoing & future developments
Execution Information

- We need more information about what’s running!
  - Sample rows (first, last, random, samples)
  - Statistics
  - Isolated logging text
  - Data profiling information
  - Execution lineage
  - Environment information: memory, JVM, disk, CPU, ...
  - Better user interfaces, tools, API, web services, ...

- HOP-4024 : Create a new execution information platform
Beam Pipeline validation

● Novice users need better advice
  ○ Embarrassingly parallel nature trips folks up
  ○ Locality of files (C:/Users/ is not available on Dataflow)
  ○ Some transform usages might make little sense
● Give advice when developing (GUI) and when running
● HOP-3863, HOP-3984, HOP-4063, HOP-3997, HOP-2053, …
Beam Pipeline improvements

- HOP-3971: Push Hop config details and variables to Beam code
- HOP-3689: Investigate splitting up Spark, Flink and Beam libraries
- HOP-2814: Split the Beam plugin into separate modules
- New IO support for Snowflake, Splunk, MQTT, Debezium, ...
Questions?

Contact info:
Twitter: @mattcasters
Linkedin: mattcasters
Github: mattcasters
Apache: mcasters