Get rid of traditional ETL, Move to Spark!

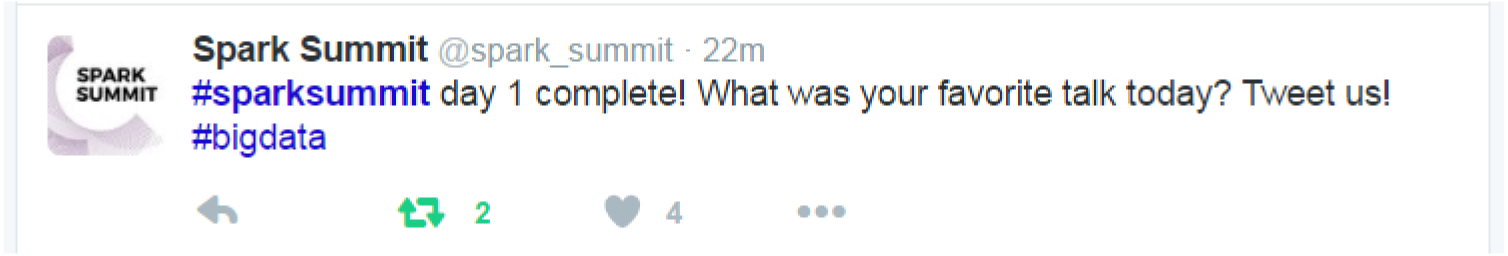
Bas Geerdink
ING

Who am I?

Bas Geerdink

- Chapter Lead in Analytics area at ING
- Master degree in Artificial Intelligence and Informatics
- Spark Certified Developer

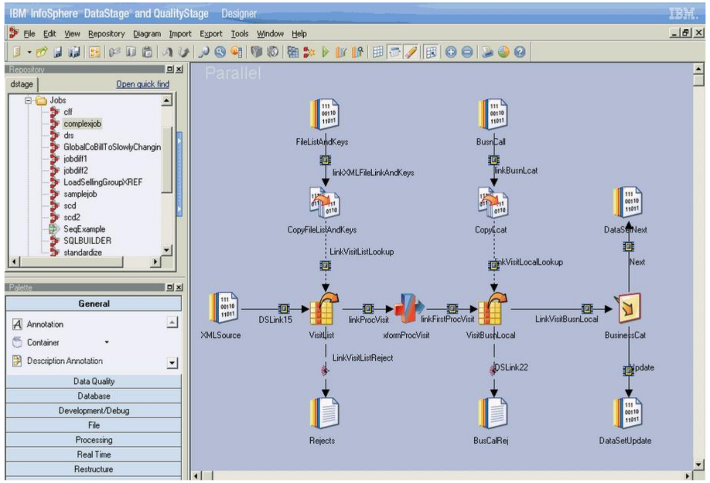
- @bgeerdink
- https://www.linkedin.com/in/geerdink

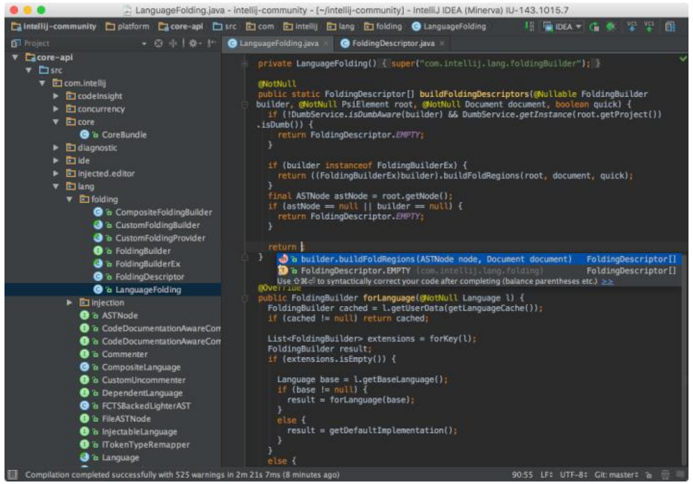


Spark Summit @spark_summit · 22m

#sparksummit day 1 complete! What was your favorite talk today? Tweet us! #bigdata

2 4





```
  private LanguageFolding() { super("com.intellij.lang.foldingBuilder"); }

  @NotNull
  public static FoldingDescriptor[] buildFoldingDescriptors(@Nullable FoldingBuilder
builder, @NotNull PsiElement root, @NotNull Document document, boolean quick) {
    if (!DumbService.isDumbAware(builder) && DumbService.getInstance(root.getProject())
.isDumb()) {
      return FoldingDescriptor.EMPTY;
    }

    if (builder instanceof FoldingBuilderEx) {
      return ((FoldingBuilderEx)builder).buildFoldRegions(root, document, quick);
    }
    final ASTNode astNode = root.getNode();
    if (astNode == null || builder == null) {
      return FoldingDescriptor.EMPTY;
    }

    return |
  }   builder.buildFoldRegions(ASTNode node, Document document)    FoldingDescriptor[]
      FoldingDescriptor.EMPTY (com.intellij.lang.folding)          FoldingDescriptor[]
      Use ⇧⌘⏎ to syntactically correct your code after completing (balance parentheses etc.) >>
  @Override
  public FoldingBuilder forLanguage(@NotNull Language l) {
    FoldingBuilder cached = l.getUserData(getLanguageCache());
    if (cached != null) return cached;

    List<FoldingBuilder> extensions = forKey(l);
    FoldingBuilder result;
    if (extensions.isEmpty()) {

      Language base = l.getBaseLanguage();
      if (base != null) {
        result = forLanguage(base);
      }
      else {
        result = getDefaultImplementation();
      }
    }
    else {
```

```
Compilation completed successfully with 525 warnings in 2m 21s 7ms (8 minutes ago)    90:55  LF÷  UTF-8÷  Git: master÷
```

SPARK SUMMIT
EUROPE 2016

Definition of ETL

“A repeatable programmed data movement”

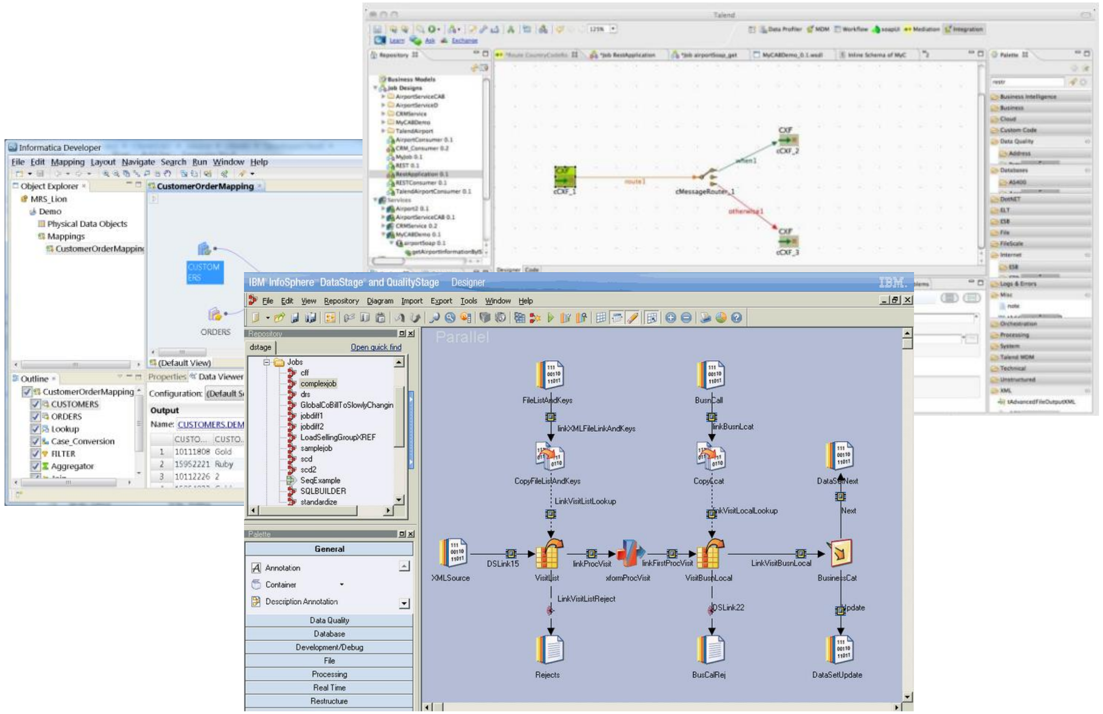
Extract:	get data from source systems
Transform:	filter/map/enrich/combine/validate/sort/…
Load:	store data in a data warehouse or data mart

Use cases:

- Data loading
- Data migration
- Data ingestion
- …

ETL Tools

- IBM InfoSphere DataStage
- Oracle Warehouse Builder
- Pervasive Data Integrator
- PowerCenter Informatica
- SAS Data Management
- Talend Open Studio
- SAP Data Services
- Microsoft SSIS
- Syncsort DMX
- CloverETL
- Jaspersoft
- Pentaho
- Nifi



What has changed?

Business Intelligence	→	Big Data
Data Warehouse	→	Data Lake
Applications	→	Microservices
ETL	→	…

The Future of ETL Tools

- Only develop connectors for integration?
- Rebuild entire back-end to Hadoop/Spark/Flink?
- Provide a GUI with code generation?

A Quiz!

What is the most *difficult* part for developers?

What is the most *resource intensive* part?

E / T / L

ETL Hell

- Data getting out of sync
- Performance issues
- Waste of server resources (peak performance)
- Plain-text code in hidden stages
- Click, click, click, click, click (RSI danger!)
- CSV files are not type-safe
- All-or-nothing approach in batch jobs
- Legacy code
- …

Is NO-ETL The Future?

- Why move data around?
- Alternative: keep data at the source, make it available in API's (microservices architecture)
- ETL is an intermediary step, and at each ETL step you can introduce errors and risk:
  - ETL can lose data
  - ETL can duplicate data after failover
  - ETL tools can cost millions of dollars
  - ETL decreases throughput
  - ETL increases the complexity of the pipeline

(source: noetl.org)

Intermediate: use Spark for ETL

- Parallel processing is built-in
- Runs on top of Hadoop, which is probably your data source anyway
- It’s *just* Scala code (or Python, or Java)
- Machine learning can be thrown in to do more interesting things
- Good support for security, unit testing, performance measurement, exception handling, monitoring, etc.

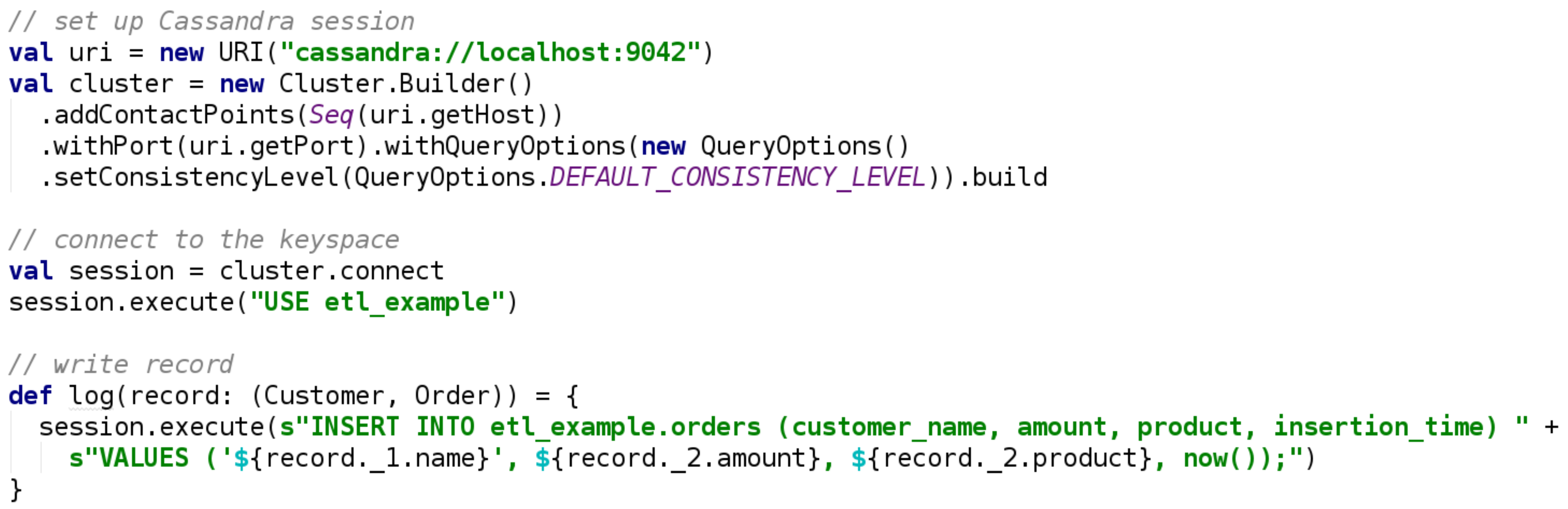
Code example #1: EXTRACT
Get data from HDFS

```scala
// initialize Spark for batch processing
val spark = SparkSession.builder
  .appName("spark-etl")
  .master("local[*]")
  .getOrCreate()

// get customer data from HDFS in Spark SQL Dataset
val rawCustomers = spark.read
  .textFile("hdfs://data/customers.csv")

// cache to reuse the dataset
val customerData = rawCustomers.cache()
```

Code example #2: TRANSFORM Filter, Map, Join

```scala
// get business classes
import spark.sqlContext.implicits._
val customers = customerData.as[Customer].as("CUSTOMERS")
val orders = orderData.as[Order].as("ORDERS")

val records = customers
  // only load premium customer over 18
  .filter(_.premium)
  .filter(_.age > 18)

  // combine with orders, creates a dataset of tuples (customer, order)
  .joinWith(orders, $"ORDERS.customerId" === $"CUSTOMERS.id", "left outer")

// calculate total price
val total = records.map(r => r._2.amount * _._2.product.price).reduce(_ + _)

// output in a nice format
records
  .map(r => s"${r._1.name} has ordered ${r._2.amount} units of ${r._2.product.name}s, for a total price of $total")
```

Code example #3: LOAD
Store transformed data in Cassandra



```
// set up Cassandra session
val uri = new URI("cassandra://localhost:9042")
val cluster = new Cluster.Builder()
  .addContactPoints(Seq(uri.getHost))
  .withPort(uri.getPort).withQueryOptions(new QueryOptions()
  .setConsistencyLevel(QueryOptions.DEFAULT_CONSISTENCY_LEVEL)).build

// connect to the keyspace
val session = cluster.connect
session.execute("USE etl_example")

// write record
def log(record: (Customer, Order)) = {
  session.execute(s"INSERT INTO etl_example.orders (customer_name, amount, product, insertion_time) " +
    s"VALUES ('${record._1.name}', ${record._2.amount}, ${record._2.product}, now());")
}
```

Code example #4: Continuous ETL Stream from file or message bus

```scala
// initialize Spark Streaming
val conf = new SparkConf().setAppName("fast-data").setMaster("local[*]")
val ssc = new StreamingContext(conf, Seconds(5))

// extract
val stream = ssc.fileStream("hdfs://data/customers.csv")

// transform...
// load...

ssc.start() // tell the StreamingContext to start receiving data
ssc.awaitTermination()  // wait for the job to finish
```

What to choose?

- Technology is just… technology
- Choose a mindset / culture / way of working

- Do you really need a full Hadoop/Spark cluster for your average ETL?
- Do you really need an expensive vendor enterprise tool for your ETL?

Considerations…

- Testing (unit, functional, performance)
- Need for visualization and explanaition
- Flexibility: Continous Delivery, Automation, Reusability
- Simplification leads to less errors
- Tool vs framework
- A hybrid solution? E.g. code generation tools

Key takeaways

1. Pick one: ETL, ELT, ELTL, …
2. Treat all data equal: batch and stream
3. *Continuous* ETL: don’t wait for a phase to complete
4. Don’t just transform; enrich, alert, *predict*
5. Build for scale: distribute data and logic
6. Automate everything
7. Think about NoETL: each copy is a risk!

SPARK SUMMIT
EUROPE 2016

#SparkSummit

THANK YOU!

ING is hiring ☺