# Curriculum Vitae

Lars Albertsson
Mapflat
lalle@mapflat.com
www.mapflat.com

## Data processing consultancy

I help companies create value out of a flood of data by building scalable and sustainable data processing solutions – big data systems that scale up or down, depending on client needs. I offer product design (backend), architecture and implementation for data-driven products and features, such as recommendation systems and business insight systems. I have worked with systems that range in scale from a few data sources and developers, to terabytes per day of incoming data that drive the work of dozens of developers and data scientists.

I also provide strategic advice for how to collect and create business value from large amounts of data, as well as technical strategy advice.

I always aim to build state of the art engineering, but to deliver incrementally, starting with small and simple minimum viable products that yield end-to-end value, and then improve functionality and scalability driven by customer needs and priorities in an iterative and lean manner. I strive to continuously increase quality, automation, team efficiency, and my own productivity, through reflection and improvement to work processes. I favour simple and straightforward solutions over complexity, proven and open components over proprietary and exotic solutions, team work over individualism, and simple and agile processes over heavy-weight or arbitrary procedures.

I explore and learn new techniques and technology, enjoy learning from my peers, and take time to help my colleagues where I can. I enjoy working in teams that reduce complexity and build strong abstractions, take pride in good engineering, and consistently apply the methods, structure, and processes required to reach project goals.

I have picked up a toolbox of techniques for technical productivity and efficient agile team collaboration, and help teams apply structure to address efficiency shortcomings, in a manner that harmonises with team culture and focuses on value delivery over rituals. I do not strive for classic leadership positions, but often end up as informal technical lead, product owner, or Scrum master.

# Skill areas

## Data processing

Scalable architecture, batch process development, stream processing, data collection, data pipelines, workflow managment, continuous delivery & deployment, test strategy, data quality engineering, data modelling, data exploration & visualisation.

## Software engineering

Large scale applications, distributed systems programming, multithreading and concurrency, debugging and troubleshooting, continuous delivery, software architecture, object-oriented design, automated testing, test-driven development, development & code analysis tools, configuration management, test & build systems, compilers & toolchains.

## Ground level leadership

Agile development workflows, Scrum, customising tools (e.g. Jira) to team workflows, democratic & consensus-based techniques for decision making and prioritisation, workshop facilitation techniques.

## Domain experience

Software-as-a-service applications, financial software, trading systems, data privacy, natural language processing, machine learning, media applications.

## Other technical experience

Computer architecture, Unix/Linux system administration automation, TCP/IP networking, video/audio conferencing.

## Component experience by category

**Data processing** Spark, Hadoop (Cloudera, Hortonworks, MapR, Amazon EMR), MapReduce, Crunch, Scalding, Kafka, Storm, Spark Streaming, Akka Streams, Cassandra, HDFS, Amazon S3, Google BigQuery, Google Cloud Storage, Hive, Luigi, Sqoop, GFS, Parquet, Avro.

**Cloud environments** Amazon Web Services, Google Cloud Platform, Google Borg, Yarn

**Service-oriented components** Guice, Spring, REST, Akka, Zookeeper, Consul

**Programming languages** Java (10 years), C++/C (10+ years), Python (15+ years), Scala (4 years), Groovy

**Development tools** IntelliJ, Eclipse, Maven, Gradle, SBT, SCons, Pex, GDB, GCC, Make, Ant, GNU autotools, Bazel/Blaze

**Test & process tools** JUnit, TestNG, Mockito, jMock, Mox, Scalatest, Specs2, Tox, Nosetest, Google proprietary, Jenkins, TeamCity

**Version control** Mercurial, Git, Subversion, CVS, Clearcase, Perforce

**Virtualisation** Vagrant, Docker, Simics, Valgrind

**Databases** Cassandra, MySQL, Postgres, Redis

**Web frameworks** Play, Django, Tornado

**Infrastructure-as-code tools** Terraform, Packer, CloudFormation, Troposphere, Ansible, Puppet, Chef, Cuisine

## Work experience

For details on data processing experience and Mapflat clients, please see next section.

January 2016 – present: Mapflat, founder and consultant. Building data processing solutions on a freelance basis. I have helped companies with technical implementation of data processing systems, as well as strategy for how to create business value from data, how to organise for building data-driven products, and technical strategy and architecture for data processing systems. See next section for list of client projects.

August 2014 – January 2016: Schibsted Media Group, data architect. I designed and implemented a data processing platform that collects data from Schibsted's subsidiary companies (150+M active users in total), and produced refined datasets, e.g. user profiles for ad matching, recommendations, and business intelligence. The technology components used include Spark, Kafka, Storm, Cassandra, Secor, Kinesis, Luigi, Hadoop, Amazon AWS (S3, EC2, EMR).

April 2013 – August 2014: Spotify, data systems engineer. I developed data processing software that receives user-generated data coming from client applications (50 million users, generating 5-10 TB / day), and refines and combines that data to make it useful for Spotify's analysis services and data-driven features: business intelligence, recommendation systems, record label reporting, top lists, etc. The data processing software uses a flora of big data technology components: Hadoop, HDFS, Hive, Sqoop, Pig, Kafka, Storm, Luigi, Crunch, Cassandra. I was working in the team responsible for developing the core data pipelines, operating the data processing services, and for providing a data

processing environment to other teams. I also participated in building a new generation of the music metadata ingestion and curation service.

January 2011 – March 2013: Cinnober Financial Technology, software architect. I was a member of the core team, building a next generation, microseconds latency trading engine (in Java!). I also modernised the development tools suite and test automation framework, and developed a new marketplace for the Quadriserv securities lending exchange.

February 2010 – January 2011: Recorded Future, software engineer. I developed an RSS and web crawler, which performs natural language processing, including date+entity+event detection, and transforms the data obtained to RF's model of news events. I also created customised crawlers, e.g. for Twitter. I defined the company's test and data quality strategy, and built integration test harnesses.

2007 – February 2010: Google, software engineer. My team developed Google's video conferencing service Hangouts, integrated in Gmail, Google Drive, Google+, and smartphones. I created tools, infrastructure, and processes that improve product quality and team productivity. I designed and implemented an integration test framework for chat and video conferencing services. It is still actively maintained, and used for testing the servers for Hangouts, Google Talk, Google+ notifications, and chat in Gmail.

1997 – 2007: Swedish Institute of Computer Science, researcher. My main research area was testing and debugging tools for real-time, parallel, and distributed software, based on instruction-set simulation. I have created Nornir, a test and debug environment consisting of some 50,000 lines of C++ and C code, ˜400,000 lines of generated C++ code, ˜20,000 lines of Python code, and about 20 million lines of source in integrated open-source components. I have designed Nornir and wrote more than 90% of the code, excluding imported components. The primary inventions in Nornir are:

- A tool for provoking, reproducing, and debugging race conditions during testing.
- Runtime analysis tool that detects race conditions in operating systems and embedded applications. This tool was demonstrated on the Linux kernel and found several previously unknown errors.
- Virtual memory translation infrastructure enabling the use of standard debuggers (GDB) for debugging Unix applications in simulated computers.
- Infrastructure for automated testing of distributed software by comparing internal state in multiple processes in simulated systems without affecting test execution.

Other projects at SICS: Data dependency profiling tool for converting programs to use multithreading. TCP/IP implementation in the research operating system Nemesis. IPv6 and IPsec implementation in HP-UX.

2001 – 2004: 1 year internship at Sun Microsystems in Boston, followed by part-time remote work from Sweden. I worked with system verification of high-end servers and developed simulation-based verification environments.

1996 – 1997: Six month internship at the Center for Parallel Computers, Stockholm. Interactive visualisation using supercomputers.

1995 – 2003: Founder of Peritiam Konsult HB. Software and web development, customer solutions and IT support. Customers: IBM, Owell (now Logica), KTH, Försäkringskassan (Swedish social security).

1993 – 1995: IBM, part time work. Customer solutions in OS/2 environment.

## Data processing experience

### Mapflat

Clients during 2018:

- Bonnier News. Product owner and architect for data platform project.
- Major Swedish bank. Same as 2017.

Clients during 2017:

- creative.ai (deep learning startup). Technical bootstrap of a real-time data processing environment for deep learning services. Technology used: AWS EC2, AWS S3, AWS VPC, Terraform, Docker, Packer, Python.
- Major Swedish bank. Technical and management strategy regarding a large data lake project. Data processing architecture. Technical GDPR implementation. Assistance with recruitment. Data engineering education.
- Spotify. Same as 2016.
- Swedish bank. Technical strategy.
- Swedish bank. Technical strategy.
- ChyronHego. Data processing architecture.
- Combient (Swedish industry digitalisation innovation joint venture). Technical strategy.
- Small Giant Games. Data processing architecture.
- Offerta. Technical strategy.
- Axel Johnson. Technical strategy.
- CodeBuilders. Data processing architecture.

- Baymarkets. Data processing architecture.

Clients during 2016:

- Spotify. I have been building batch processing pipelines for the purposes of royalty calculations, record label reporting, user insights, and analytics. During this project, I have used Apache Crunch, Apache Scalding, Hive, Luigi, PostgreSQL, Docker, Scala, Java, Python, Google BigQuery, Google Cloud Storage, Kibana, Qlik Sense, Puppet, Scio, Apache Beam, Kubernetes.
- US mobile game company. I worked on batch and stream processing analytics solutions using Apache Spark Streaming, Apache Kafka, Scala, Python, Redis, Docker, Vagrant, Ansible, Luigi, Hive, AWS EC2, AWS S3.
- Aptoide (Android app market, based in Portugal). Data processing architecture and strategy.
- Shobr (digital grocery market, based in Denmark). Data processing architecture and strategy.
- Combient (Swedish industry digitalisation innovation joint venture). Technical strategy.
- Hitta.se (digital yellow pages). Technical strategy.
- Swedish bank. Technical strategy.

In addition to helping clients, I have also held presentations at international conferences on multiple aspects of data processing: Architecture, test & quality, developer techniques, strategy, and team organisation. Slides and video recordings of a subset of my presentations can be found at http://www.slideshare.net/lallea/. Readers of this resume may also find my data engineering resource compilation useful: http://www.mapflat.com/lands/resources/reading-list/.

## Schibsted

Schibsted Products and Technology (SPT) is an effort to create a common data-driven technology platform for all Schibsted-owned companies. I joined as the second engineer, and we grew to hundred people in little over a year. I took the role of a data architect, and made a plan for how to enable many engineers and data scientists to use data, while still maintaining user privacy, and keeping cloud costs under control. The implemented architecture is similar to what is described in http://www.slideshare.net/lallea/data-pipelines-from-zero-to-solid, using Kinesis as data collection transport, S3 with EMRFS for storage, Spark on Amazon EMR for batch processing and Zeppelin as data science workbench.

While we were still very small, we started out developing an end-to-end coherent system with a data sink REST endpoint built with spray.io, pushing incoming data to Kafka, read by Secor to S3, and then processed by Spark, with Cassandra as egress target. We built the chain completely with Docker, using Weave for software-defined networking, and Consul for service discovery. The networking Docker components were not sufficiently mature, however, and the cluster components (Kafka, Spark) required too much work to work within Docker, so we pivoted from parts of this solution. As the company grew, in multiple locations, we decoupled the technology, and data collection was instead implemented with Amazon SQS and Kinesis. I continued working on building data pipelines infrastructure with Spark on EMR using S3 with EMRFS for storage.

In order to enable data scientists and engineers that are not Spark experts, and also to preserve privacy constraints, we created a gateway tool that encapsulated all relevant data functionality in a transparent and extensible manner. We used it to build dynamic creation of EMR/Spark clusters using CloudFormation + Troposphere, manage datasets, install development toolchains, create Zeppelin workbenches, etc. The gateway tool is basically configuration as code, making engineers enabled without requiring knowledge of operational configuration. It also allows us to audit and limit data access for privacy protection.

We built a continuous deployment pipeline for Spark jobs, similar to the one at Spotify, but using different technology: GHE, Travis, Artfactory, Pip, Luigi, and EMR clusters.

I spent most of 2015 using, and continuously improving, the platform we built in order to build pipelines for user profile modelling - using user behaviour to infer attributes such as age, gender, location and interests. I worked directly with data scientists and together developed machine learning models, assessed their validity, and converted them to production pipelines. They are primarily used to profile and segment users for Schibsted's online advertisement market.

I also maintained a production system serving churn analytics for the Swedish paper Aftonbladet. It is built on MapR Hadoop, Hive and Luigi.

Since I started early in SPT, I drove a number of organisational and leadership aspects: Source code control and code review, establishing agile practices with the use of Jira, recruitment interview processes, and data processing test practices.

Most of the business logic and data processing code at SPT was written in Scala, using Python for integration components. We added inflow from subsidiary companies one by one, and the data volumes grew quickly. As of January 2016, we were processing on the order of tens of TB of incoming data per day, representing tens of billions of user events, generated from tens of millions of unique users in a few dozen different Schibsted companies. The Leboncoin classified ads market in France was the biggest contributor.

Technology used: Spark, Kafka, Cassandra, Secor, Kinesis, Luigi, Hive, Zeppelin, Parquet, Amazon AWS (S3, EC2, EMR), Scala, Python, Hadoop (MapR), HDFS, Jira, Docker, Consul, Weave, IntelliJ, git, GHE, Jenkins, Travis, Pip, Artifactory.

## Spotify

I was the most senior member of the core data team at Spotify. Our responsibilities included the data processing infrastructure, which was used by ~100 developers and data scientists. Our team also owned the production of core datasets, e.g. all songs played, user accounts, playlist changes, etc. These were produced daily and used downstream by many teams.

The data infrastructure at Spotify included Kafka for ingesting user and system events to a Hadoop cluster (600 nodes at the time). Most data pipelines were originally written in Python MapReduce, using Luigi for both business logic and workflow management. This solution had issues with developer productivity, stability and performance. We evaluated alternative solutions; I ported jobs to Hive, but found it less suitable for production pipelines than for interactive analytics. My team instead initiated and rolled out a new data processing framework based on Java and Apache Crunch. We kept Luigi as pipeline description DSL and for workflow management. We created a development workflow where pipeline developers could spend most of their time coding business logic and test pipeline jobs locally, rather than waiting for cluster time for test runs. We replaced the legacy job deployment processes based on Puppet and Debian packages with a continuous deployment pipeline based on GitHub Enterprise, Jenkins, Luigi and an integration layer with Crunch.

During this period, we also transitioned the data processing environment from CSV storage format to Avro, and from Cloudera Hadoop 1.x to Hortonworks Hadoop 2.x running on Yarn.

There were production stability issues with the core data pipelines when I joined, and we stabilised them in an agile manner, measured different factors, and addressed the worst problems first. As part of the stabilisation, we created solutions for dumping the user account databases from Postgres (using Sqoop) and Cassandra databases (custom at first, moving to Aegisthus) to HDFS in a robust manner. We also improved the stability of the event collection mechanism. It is based on Kafka, with a custom end to end transport layer on top. End to end semantics on top of Kafka is an anti-pattern, but it could not be removed for non-technical reasons. We made an attempt at stabilising it by integrating with the host monitoring system built on Riemann, which turned out not to be stable enough to use as base. In the end, we had to address it with a combination of technical, organisational, and social solutions.

In order to smoothen the rollout of the Crunch-based platform, I embedded one of the early adopter teams, which was responsible for music metadata quality and curation. I eventually transitioned to the metadata team, and we built a new curation solution using data pipelines, to replace the legacy solution, which was based on continuous mutation of a relational database.

I also initiated and drove an engineering productivity assessment at Spotify. We assembled engineers that had experience of large productive (US) companies, identified

productivity aspects where we were lacking, and formed a backlog of improvements and handed to leadership.

Technologies used: Hadoop (Hortonworks, Cloudera), HDFS, MapReduce, Hive, Sqoop, Pig, Kafka, Storm, Luigi, Crunch, Cassandra, Postgres, Java, Python, Scala, Yarn, Avro, Jira, git, GHE, Jenkins, IntelliJ.

## Education

Master of Science in Electrical Engineering 1999 from the Royal Institute of Technology, Stockholm.

## Miscellaneous

Male, born 1974 in Stockholm, Sweden. Languages: Swedish (native), English (fluent), French (tourist level). Swedish driver's licence, class B (passenger car). "Kustskepparintyg" (maritime navigational certificate for private boats). I play volleyball in the Swedish national series, and I used to play bridge at international level (EU junior teams champion 1998, Swedish national league champion 1999).