

Technology Day 2023 / 13.11.2023

# Data Versioning

**INNOQ**



**Rainer Jaspert**  
Senior Consultant



**Alexander Kniesz**  
Consultant

# Agenda

- Why do we care?
- What is data versioning?
- Approaches to data versioning
- Complexity of data version control
- Elaborating best practices

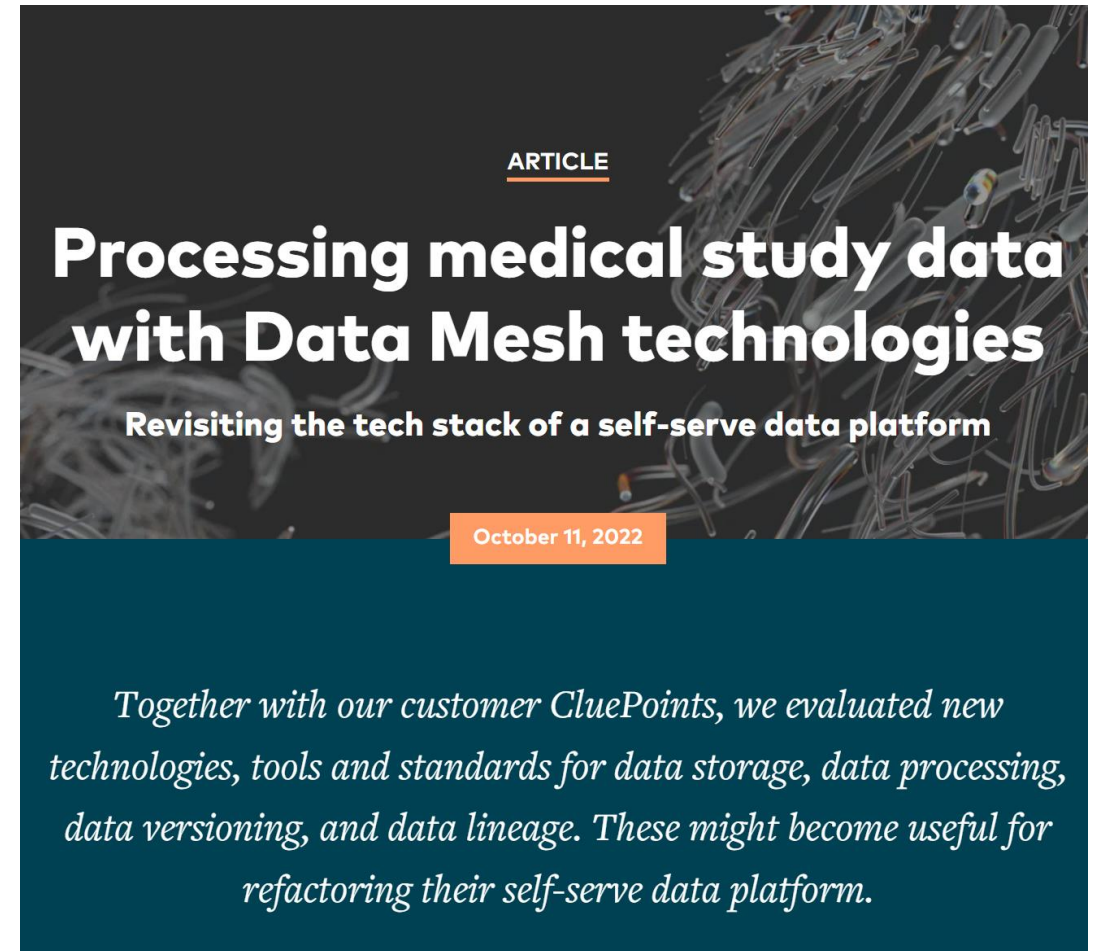




**Why do we care?**

# Project challenges

- Proof of Concept done with CluePoints
- Refactoring their platform for Risk-Based Quality Management (RBQM) of medical study data
- They were using full data duplication for each set of data to be analyzed
- We looked at data versioning tools (specifically lakeFS) to support this use case



**ARTICLE**

## Processing medical study data with Data Mesh technologies

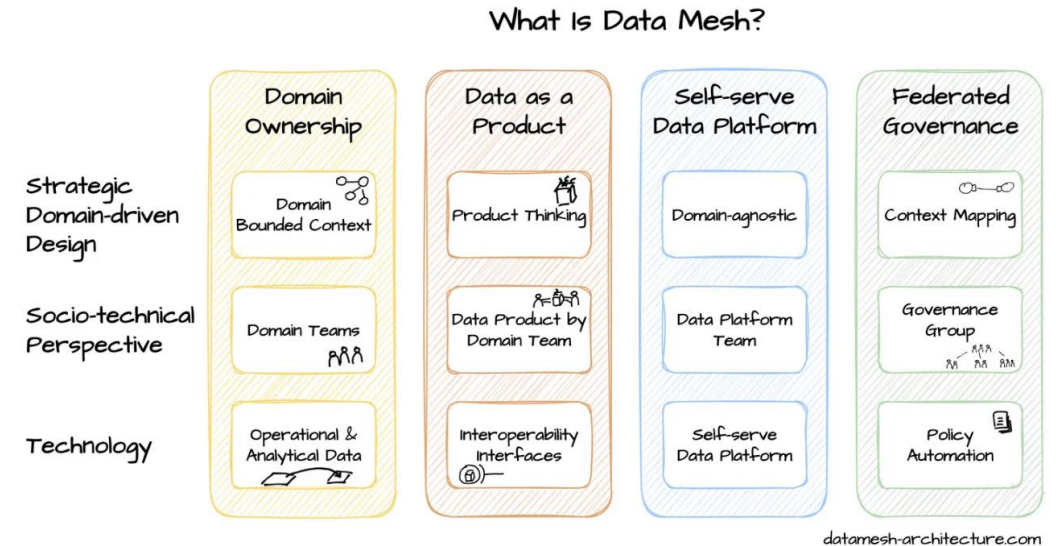
Revisiting the tech stack of a self-serve data platform

October 11, 2022

*Together with our customer CluePoints, we evaluated new technologies, tools and standards for data storage, data processing, data versioning, and data lineage. These might become useful for refactoring their self-serve data platform.*

# Data Mesh Architecture initiative

- We are interested in the Data Mesh approach with
  - Design and implementation of data products
  - Establishing a self-serve data platform
- Data versioning is usually needed for data products
- Data versioning tools provide useful features



The term *data mesh* was coined by [Zhamak Dehghani](#) in 2019 and is based on four fundamental principles that bundle well-known concepts:









The **domain ownership** principle mandates the domain teams to take responsibility for their data. According to this principle, analytical data should be composed around domains, similar to the team boundaries aligning with the system's bounded context. Following the domain-driven distributed architecture, analytical and operational data ownership is moved to the domain teams, away from the central data team.


# Multiple tool options

- Multiple data versioning tools have emerged in the last 10 years
  - Version control systems for large data
  - Data versioning tools for data lakes
  - DBMS with version control
  - Data versioning support in data pipeline tools for machine learning
- Existing comparisons and evaluations may be biased by tool vendors

A complete overview revealing a diverse range of strengths and weaknesses for each data versioning tool.

**Best 8 Data Version Control Tools for 2023**

	 DagsHub DDA	 DVC	 dolt	 LFS	 lakeFS	 neptune.ai	 Pachyderm	 DELTA LAKE
<b>Open Source</b>	✓	✓	✓	✓	✓	✓ ⚠	✓ ⚠	✓
<b>License</b>	MIT	Apache 2.0	Apache 2.0	MIT	Apache 2.0	Non-standard license	Non-standard license	Apache 2.0
<b>Release Date</b>	2022	2017	2018	2017	2020	2017	2014	2019
<b>Data Format Agnostic</b>	✓	✓	✗	✓	✓	✓	✓	✗
<b>Cloud/Storage Agnostic*</b> <small>(Support most common cloud and storage types)</small>	✓	✓	✓	✗	✓	✓	✓	✗
<b>Simple to Use</b>	✓	✓	✗	✗	✗	✗	✗	✗
<b>Easy Support for Big Data</b>	✓	✗	✗	✗	✓	✓	✓	✓

 DagsHub

*Best 8 data version control tools for 2023 (Source: DagsHub)*

# Non standardized terminology

- Data versioning
- Database versioning
- Data version control (concept and tool name)
- Git for data
- Versioned data lake
- ...

# Personal interest



Consultant at INNOQ  
for 3 years.

Master in Data Science  
achieved while working  
at a student at INNOQ.

Strong interest in  
machine learning and  
automating ML  
pipelines (MLOps).



Senior consultant at  
INNOQ for more than 15  
years.

Mostly working as a  
software architect.

Strong interest in  
analytical data processing  
(OLAP), data lakes and  
data warehouses.

Both supporting our data mesh architecture initiative at INNOQ believing it will increase the project opportunities in our favorite areas.





**What is data  
versioning?**

# What does a data version provide?

- The state of an entity instance as it was known at a given time
  - What did we know about a clinical trial participant 6 weeks ago?
  - State of patient (eg. names, date of birth, contact data, profession, previous illnesses, ...) from 1. Oct 23
- ... together with its specific context
  - What did we additionally know about the patient's participation in the clinical trial?
  - Data from visits already recorded at that time
  - Results from physical examinations already finished by that time
- ... for all instances (regarding to a given repository)
  - What did we know about all patients 6 weeks ago?

# ... and the effective entity state?

- Is the state of an entity instance at a given time as known by today also provided by a data version?
  - What do we know today about the health status of a participant as it was 6 weeks ago?
- Typically no, but
  - may be derived from multiple versions of the participants health status
- Yes, if
  - operational system maintains the effective state. In our use case there would be a need to support retroactive changes, eg by maintaining a bitemporal history

# Data versioning features

## Basic

- Maintain and access multiple data versions
- Revert to a former data version

## Enhanced

- Create copies (forks) of a data version
- Create branches to allow for separate histories of a data version
- Compare two data versions and identify the diffs
- Merge two data versions and identify conflicts

# What is data versioning needed for?



Descriptive analytics

Predictive analytics

Diagnostic analytics

Prescriptive analytics

Audit / Rollback data changes

Data Set Citation



# **Approaches to data versioning**

# Approaches to Data Versioning

- Full duplication
  - Explicitly naming and maintaining each data version
- Temporal attributes
- Version Control System

# Temporal Attributes

- Relying on DBMS features
  - Uses DB tables to maintain data versions as data records
  - Uses ACID guarantees to create the sequence of data changes
- Storing historical versions of database records
  - Explicitly adding a history table with valid-from-, valid-to-fields
  - Or using system-versioned temporal tables (which do the same)
- Querying with „in-between“ predicates
- Mainly suitable for „basic“ data versioning features
- Often used in multidimensional modelling of Data Warehouse DBs
  - Slowly Changing Dimensions – SCD Type 4
- Typically loaded through ETL- (or ELT-) Processing



# Version Control System

- Relying on a metadata repository for all data versions
  - Provides a unique reference for each data version
  - Links to the data storage component holding the actual data
- Repository supports operations to maintain the history of data versions, eg
  - checkout to retrieve a working copy of a data version
  - commit to apply atomic units of change (extending the data version history)
  - branch to create and work on different data version histories
  - merge to combine different data version histories into one
- Suitable for all data versioning features
- Standard version control systems (svn, git, ...) do not fit well to
  - large amounts of data
  - binary data representations
  - trace schema changes

# When to use which approach?

## Temporal attributes

- Recording of changes
- Reproducing past states
- Typically used for analyses on structured data – Online Analytical Processing (OLAP)

## Version control system

- Putting together and forming data sets for specific purposes
- Collaborating on the capturing of data
- Test data
- Application configuration or initialisation data
- Unstructured data – binaries

Might be outdated –  
Is this still valid?

# Evolving Technology

- Object storage on immutable file formats (eg. Parquet) replacing database storage
  - allowing to maintain large data volumes
- Open table formats (eg. Iceberg) supporting ACID guarantees
  - allowing to collaborate on data versions
- Data versioning enhancements for these file or open table formats (eg. lakeFS)
- Data pipeline tools with integrated data versioning
  - allowing to easily combine versions of source and derived data sets (eg. training data and derived ML model)
- Version control systems specifically designed for large data volumes (eg. Git LFS)
- DBMS with integrated data versioning (eg. dolt)

# No more need for temporal fields?

- Data version control systems do provide good support for data analytics
  - Directly refer to specific data versions
  - Compare data versions on different branches
- How to deal with event data?
  - New event data appears in the first data version created after event creation
  - Data versioning may not be needed as event data is immutable
- Combination of facts (multidimensional modeling) with different data versions
  - „as is“ vs „as-was“ analyses: Use a past dimensional context on current facts
  - Predictive analyses: Use a future dimensional context on current facts
  - Possible solution: Combine versioned dimensional context data with non versioned fact data



**Complexity of version  
control for data**

# Technology Stack

- Which or how well do technical components combine with a data versioning tool
  - Storage components (File system or object storage, Cloud-based or on-promises)
  - File formats (parquet, avro, orc)
  - Open table formats (Iceberg, hudi, delta)
  - Query-engines (SQL, OLAP, Spark, Map-Reduce, ...)
  - Batch and/or stream-processing
  - Container orchestration (kubernetes, ...)
- Technical components to be combined with data versioning tool
  - Specific In-/Output-/Delete-processing (eg Spark)
  - Standard Version Control System (eg git)
  - Standard RDBMS for transactional state

# Also to be considered

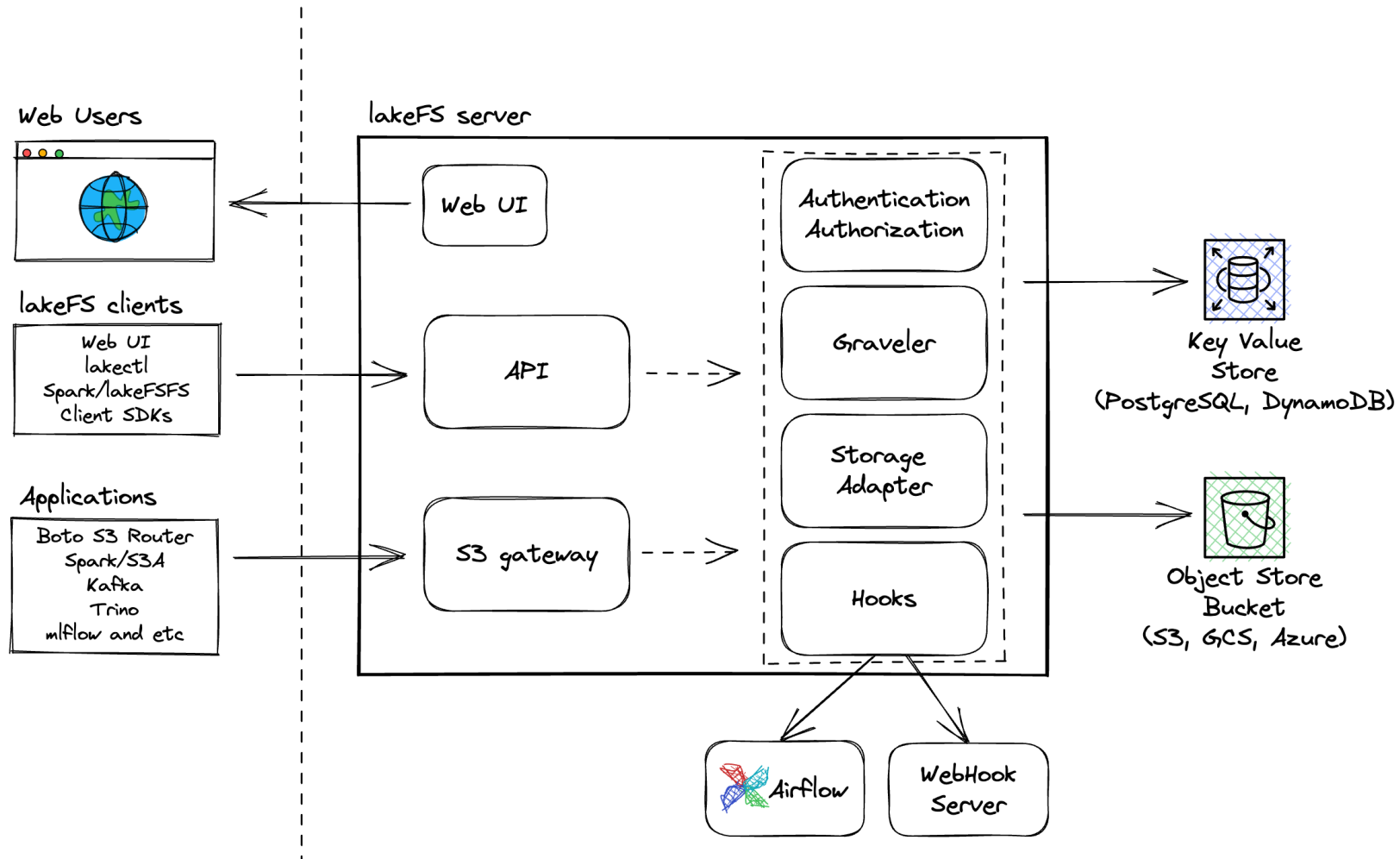
- Distributed data
- Support for schema changes
- Scalability
- Data protection
  - Data removal (DSGVO)
- Integration with data processing tools



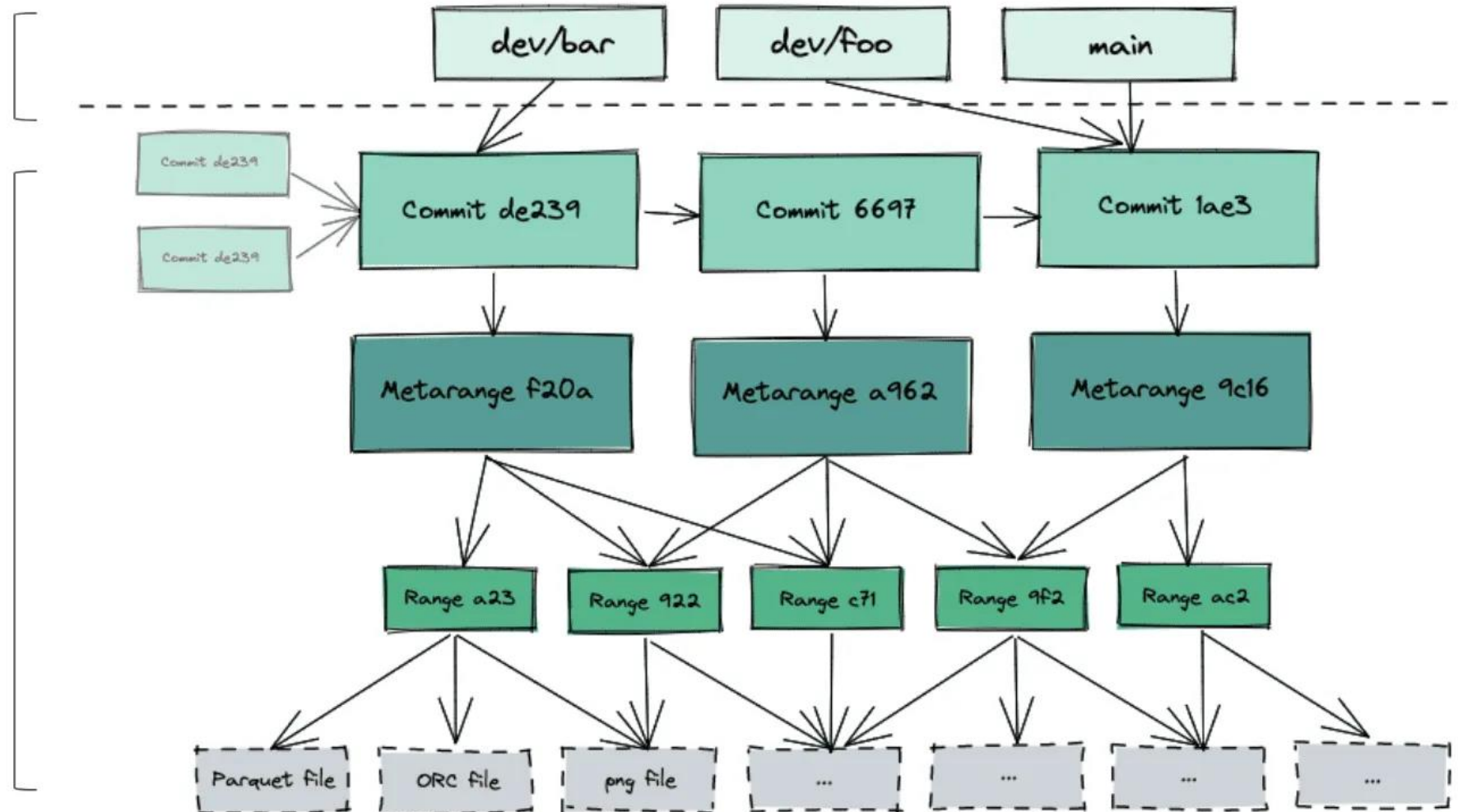
**lakeFS**



# Manage data with git-like operations



# Data versioning in lakeFS





**Further work**

# Elaborate best practices

- Participate in data mesh projects ...
- When is it still recommendable to version data by temporal attributes?
- Which data versioning tool do we prefer for typical circumstances?
- Which technology stacks fit to our preferred data versioning tools?

# Publish findings

- Publish our data versioning content to a new microsite or adding it to <https://www.datamesh-architecture.com>
- Have a talk about data versioning published in our INNOQ podcast series
- On the longer run: Working on a Data Versioning Primer

# Danke! Fragen?



Rainer Jaspert  
rainer.jaspert@innoq.com



Alexander Kniesz  
alexander.kniesz@innoq.com

## innoQ Deutschland GmbH

Krischerstr. 100  
40789 Monheim  
+49 2173 3366-0

Ohlauer Str. 43  
10999 Berlin

Ludwigstr. 180E  
63067 Offenbach

Kreuzstr. 16  
80331 München

Wendenstraße 130  
20573 Hamburg

Spichernstrasse 44  
50672 Köln

Königstorgraben 11  
90402 Nürnberg