retraced ORACLE

Technical Brief

Building next-gen microservices today to meet the demand of tomorrow: retraced

Envisioning a world where fashion companies work together on a single platform to manage and streamline their sustainability efforts, eliminating all communication noise and having a common truth about reliable sustainability information.

November, 2021, Version 1.0

ORACLE

Table of contents

List of images

Vision

Retraced envisions a world where fashion companies work together on a single platform to manage and streamline their sustainability efforts, eliminating all communication noise and having a common truth about reliable sustainability information; a comprehensive and unifying platform that connects all the parties in the complex fashion supply chain. Primarily, such a solution demands 24/7 availability, web and smartphone access, and interoperability with existing on-prem and cloud solutions. In addition, it must be built with an open API and with the potential to scale to the demand given by the seasonal nature of production. This paper aims to elaborate on the decisions made by retraced about the utilization of microservices and the different layers of the architecture, to clarify what these systems are solving, and explain how together they provide on-demand scalability while being fast and extensible at the same time.

Image 1. The sustainability management platform where impact meets efficiency.

retraced: The sustainability management platform

retraced is a sustainability management platform for the fashion industry. Its goal is to make sustainable information management easy by providing easy means to collect all relevant information, process it and then communicate it with all the stakeholders across the supply chain. A standout service offered by the retraced platform is communication of sustainability information also to the end-consumer in a straightforward and easy-to-understand manner and backed by verifiable proofs. Indeed, making the customer aware of one's sustainability efforts is a potent way to properly communicate the brand image in an understandable and effective way and further increase brand awareness and brand recognition.

Retraced is driven by an unyielding commitment to quality and transparency; printing certification labels on garments does not empower consumers to make educated purchasing decisions, nor does it appraise them of the true history of the product itself. Thus, a badge-based system of eleven badges was developed, indicating a product's various sustainability metrics. These badges are based on eleven of the UN Sustainable Development Goals [1] and are derived automatically from the information available about the product such as the supply chain chronology and the different stages it went through, from harvesting the raw materials all the way to the finished garment. The platform provides visibility into the entire supply chain, including all companies that have contributed to the final product in one way or the other. Each individual company may also have certifications like Global Organic Textile Standard [2], audits like from Elevate [3] or memberships like the Fair Wear Foundation [4] membership. Additionally, all product-specific documents can be stored on the platform as well, such as lab proofs of leather components for the use of non-hazardous chemicals. All this information is processed to automatically generate the badges for a given product. A client can even choose to link its online store to the retraced system and display the badges directly on each product's store page, providing the customers an always up-to-date sustainability status.

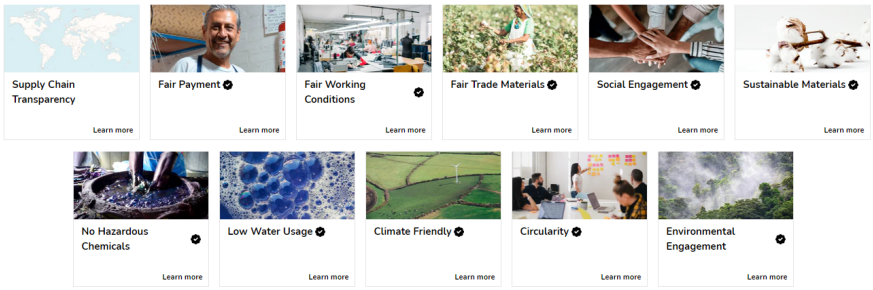

Image 2. The eleven Sustainable Development Goals badges are awarded automatically based on the certifications and other proofs of sustainability in a product supply chain.
Example: https://retraced-example.myshopify.com/products/huarache-benito

To collect all proofs in a supply chain and trace garments back to raw materials, i.e., down to the farm level, the system has been engineered to allow every party to invite the next known parties that they work with to the platform as well. Each individual party is maintaining its own profile and can populate it with the relevant sustainability information. If one party decides not to participate, other network members can choose to collaborate and keep the non-participating party's sustainability profile up to date. Based on this network, each tier in a supply chain is usually also evaluating the next tier through a preset or bespoke sustainability assessment. This assessment is essentially a

"With retraced our individual efforts in this direction become part of a bigger movement which has the potential to set new standards and maybe one day force even bigger players to take responsibility for their value chains and the people involved."

Carolin Hofer
Co-CEO
Jyoti – Fair Works

Sustainable Development Goals

Are the foundation for the mission at retraced. They comprise all areas of sustainability and clear targets of all nations to foster sustainability in those areas.
https://sdgs.un.org/goals

questionnaire about common sustainability aspects, like water usage, energy consumption, certifications, etc. This gives each platform user the possibility to assess its partner network in accordance with its own sustainability requirements. For example, a company that requires all its suppliers to have a carbon offsetting program in place can easily verify that by creating a Supplier Assessment for its network.

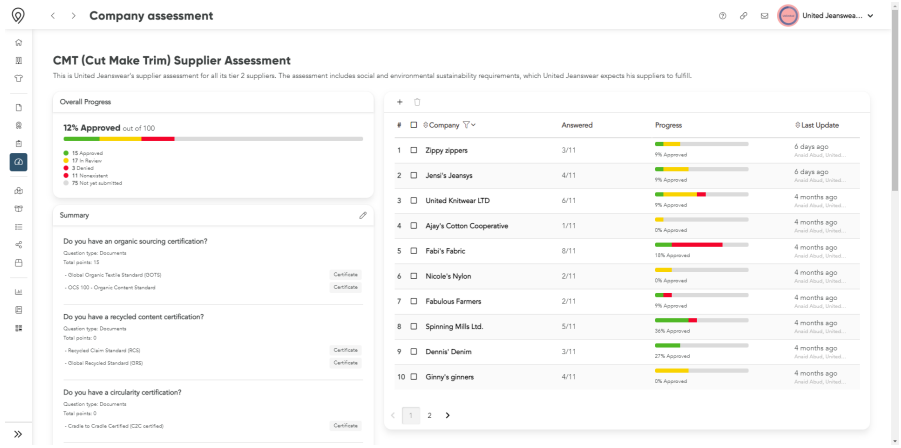

Image 3. A supplier assessment example in the retraced web back-office. This depicts a typical overview of the progress of the suppliers answering their own questionnaires and filling out requirements.

This assessment and the network management functionality, plus all the certifications, audits, and extensive document management make retraced a comprehensive sustainability management platform.

Having now understood what the platform shall achieve, the technical architecture and infrastructure will be covered in depth.

Building a globally accessible and data flexible infrastructure

To decide on an architecture, first, the interfaces of the technology customers will want to use have to be identified. To this end, two types of clients were identified; **controlled clients**, which are applications that retraced is providing and **uncontrolled clients**, which are systems connecting directly to the retraced API, like the ERP solutions of customers. The following is a list of client technologies:

- **iOS and Android app for businesses**
  Customers like farmers cannot be expected to own a laptop and use the desktop/ web version of the platform, but given the growing level of smartphone penetration in the world, they can be expected to be able to access the platform through a mobile app. In addition, a smartphone-native app provides system capabilities like a phone camera for product QR code scanning and push notifications.

- **iOS and Android app for consumers**
  Originally, this app was targeted at consumers to scan for Near Field Communication (NFC) identifiers which can be hidden inside buttons or embedded within shoe soles. A mobile app was the only feasible way to provide Near Field Communication (NFC) interaction with a satisfactory user experience. However, because NFC tags can be hard for consumers to locate and scan, this idea was dropped and as of today, the consumer app has been marked as deprecated with the NFC tagging neither advertised nor advised any more.

- **Platform website**
  A cloud-based frontend providing easy access to the complete suite of the retraced service offerings.

- **Web shop components**
  These are small web components which can be embedded in any website to display sustainability badges of products. Adding the client shop's host URL and product Shelf Keeping Unit number (SKU; an internal product identifier used for warehouse management) into the retraced platform will be automatically represented as cards, icons, or any other visualization of the eleven badges. retraced also offers one-click integrations developed for Shopify, Prestashop, Shopware and WooCommerce, which simplify and automate this process.

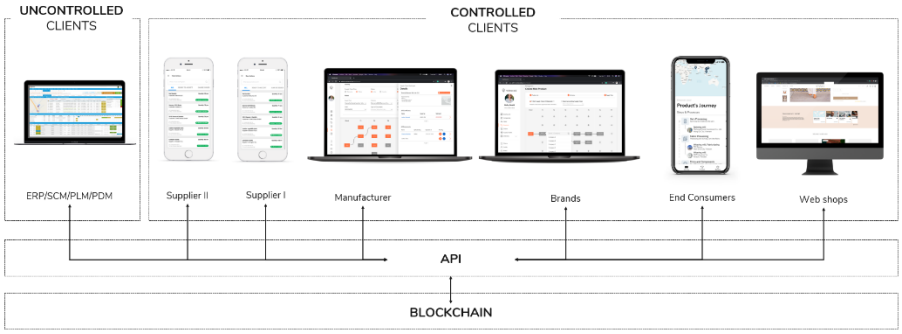

Image 4. Overview of interfaces to the retraced API.

At the heart of the retraced architecture lies a blockchain network – a Hyperledger Fabric network. Providing a tamper-proof chronological log of all noteworthy events, it allows for easy and trusted verification of certifications by the relevant authorities later. This aspect is critical and necessary since most certification authorities do not provide a public API for certificate verification and hence only with active participation in a blockchain network can the validity of a certificate be ascertained.

Networking flow of a client request

When a client issues a request, it is bundled into a single access point. This reduces exposure and simplifies development as regards access security. This is the retraced API.

Before even reaching the API, a network call must be DNS-resolved for api.retraced.co to the "A" record of the load balancer IP within the Oracle Cloud Infrastructure. The call is then directed towards that load balancer, which in turn forwards it to one of the compute instances that are running the Kubernetes cluster. Kubernetes veils the deployment complexities of the microservices and abstracts the process of a request arriving at one of the compute instances.

Inside Kubernetes, ingress rules have been set up that define how traffic must be forwarded within Kubernetes. The rules are simple: if traffic arrives at api.retraced.co, forward it to one of the running API containers.

That's it!

There is no need to tell Kubernetes how to distribute the traffic internally; it does that automatically. Of course, one can always define everything for maximum customization, however such bespoke configuration is seldom needed.

Web components

Web components provide a manipulation safe and sandboxed shadow DOM. They are the base for Google's Amplified Mobile Pages (amp.dev). They use Web Workers to parallelize browser resource capacity.

Microservices

Microservices are tiny programs which are focused on a specific technical use case like email sending, SMS sending, web analytics accumulation or others. They became very popular with Docker and Kubernetes as everything can be quickly dockerized and a lot of Docker containers run in parallel handled by Kubernetes as "container orchestration system". Thus, Kubernetes made microservices finally easily accessible for every developer.

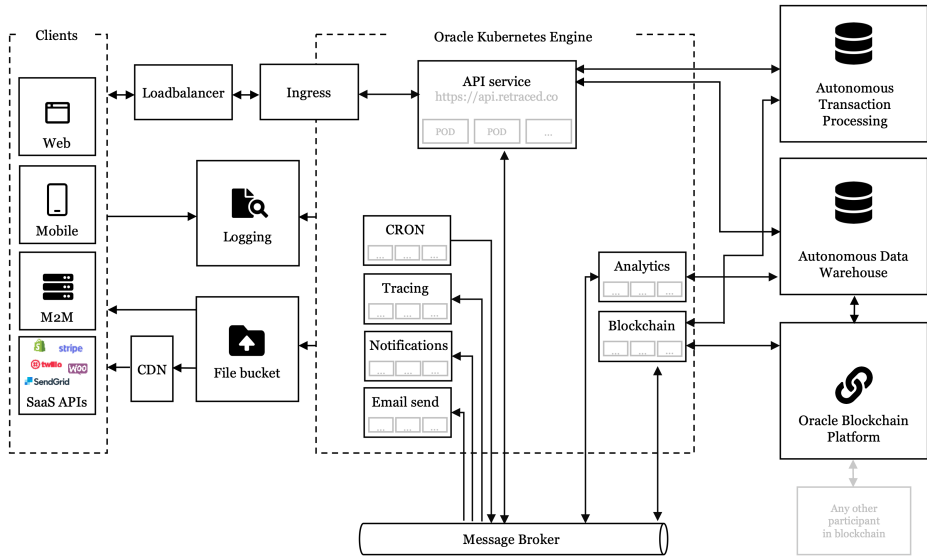

Image 5. The retraced architecture overview.

It should be mentioned that despite this setup, any container within Kubernetes can still make calls to the public internet.

Ensure authenticity of raw material origin with traceability

A core functionality for sustainability management is product traceability. Traceability means that all companies in a supply chain can populate their own relevant data to prove the transfer of goods and the system can trace these transfers all the way to their origin or final garment. Besides their receipt and shipments, most clients of retraced go even a step further and trace their in-house production processes. This in-house tracing is the enabler for full end-to-end traceability from receipt of raw materials to shipping intermediate products out. The combination of all transfers of goods between companies and the in-house tracing within the companies themselves is the powerful combination enabling farm to garment traceability.

A major architectural decision for the retraced platform was the **blockchain platform** to ensure tamper-proof audit logs and the possibility of decentralization of data validation. On the one hand, a tamper-proof system is an inevitable requirement when handling compliance information. On the other hand, a blockchain platform enables easy future participation of other companies in the retraced solution as part of the data validation process. If a company decides to participate at a later stage, it will just become another participant in an already existing blockchain with no need for manual data integration. The drawback of the blockchain platform (Hyperledger Fabric in this case) is that a synchronous write takes **several seconds** to fully complete, including a block confirmation, in just a very simplistic test setup. This delay is not acceptable for typical user interfaces. Thus, this forced the creation of a gateway service to handle temporary validation of data.

Another user interaction is the quick lookup of the whole chain of transfers for a single product. The traceability chain cannot be built up on the fly in a user-acceptable timeframe, as it requires a very complicated recursive query covering multiple technical assets of shipment, receipt, warehousing, in-house tracing, transportation and so on. Hence, it was clear that a runtime update of a traceability snapshot is needed.

Kubernetes

Is the orchestration of Docker containers over a flexible amount of compute instances. Kubernetes provides a default networking setup on top of a node cluster of compute instances. One only must define the container which should be running and how many instances of those, and Kubernetes will take care to distribute them automatically and equally over all nodes.

Supply chain mapping vs tracing

Mapping a supply chain is just building an assumption where goods were sourced from the view of the final product. It does not involve any of the other parties. Whereas traceability ensures that each party in the chain contributes to themselves relevant proofs of the transfer of goods. Thus, traceability makes the proof of origin very strong as cheating would require all parties to cheat.

With this understanding, Image 6 depicts the flow of information in the technical infrastructure.

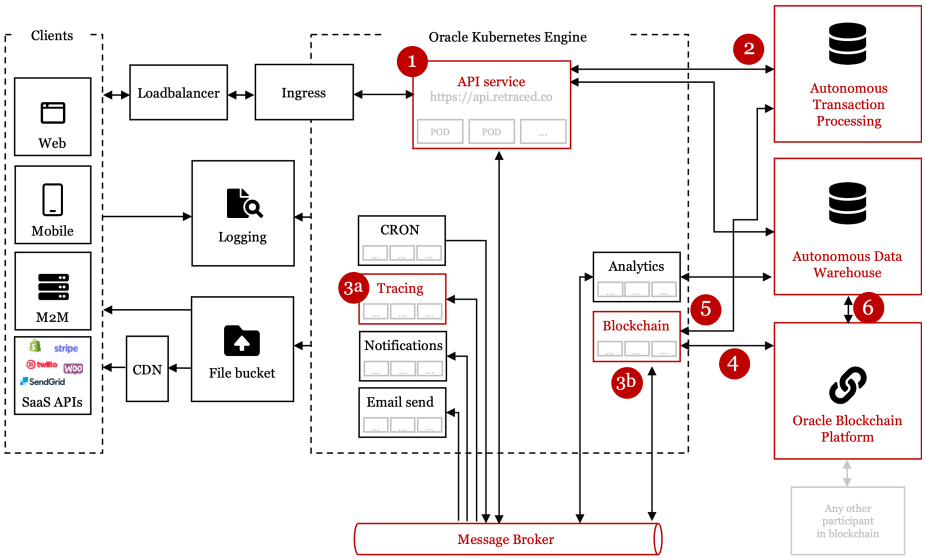

Image 6. Typical flow to ensure authenticity of collected data and build on-the-fly fast retrievable product history.

All requests start at the API service for authentication and authorization (1). There, all system critical data is stored synchronously in the Autonomous Transaction Processing cloud service. If data must be submitted to the blockchain for verification and audit purposes – not all data must go through this like, for example, analytics data – the data point is persisted into the Autonomous Transaction Processing cloud service but marked as "*temporarily accepted and pending blockchain verification*" (2).

As soon as the API has finished the business-critical operations like authentication and authorization checks as well as the actual operations of the endpoint like writing to users, companies, or orders tables, the (temporarily correct) result is returned to the caller. Additionally, messages are dispatched to the relevant microservices to be further digested. For the update of a shipment, for example, a message is dispatched to the blockchain as well as the tracing service (3a), (3b).

The tracing service picks up the message (3a) and extends the information for future lookups to be able to quickly retrieve the shipment operation and all previous operations that occurred on the goods of a shipment. As new snapshots are built over time, previous already existing snapshots can be used to eliminate any on-the-fly calculation and speed up the process by 100X and more. Likewise, when tracing information for a shipment is requested, the tracing snapshot is already there ready to be returned to a caller immediately.

Meanwhile, the blockchain service executes the call to the blockchain platform and waits for the result (4). Here, waiting for the synchronous answer of the blockchain platform is not a concern because the (temporarily correct) result has already been returned. If there are more messages coming onto the message broker than the blockchain service can digest, another instance of the service is created. Once the call returns from the blockchain platform, the blockchain service updates the asset marked as "*temporarily accepted and pending blockchain verification*" in the Autonomous Transaction Processing cloud service as fully completed (5).

The neat addition in the setup is the Autonomous Data Warehouse cloud service, which is linked directly as a rich history database to the blockchain platform (6). This is a unique attribute of the Oracle Cloud with the purpose of being able to work with the data on the blockchain. As mentioned earlier, the blockchain platform interaction is quite slow as it requires execution of a smart contract every time it is interacted with. The rich history database connection will transmit every newly created block on the blockchain directly into the Autonomous Data Warehouse cloud service. This allows business intelligence and other analytics to be executed on blockchain data efficiently and via SQL.

Technical challenges and architectural decisions

Microservices

Initially, a monolithic approach was adopted for the API service. However, this pattern soon revealed its caveats. For instance, a single call to the underlying blockchain service took almost 7 seconds for the full success response. Luckily, this circumstance was identified early on, and a course correction was made to throttle and delay the final response from the blockchain to the user, as it turned out to be not mission critical. A decision was made to instead enqueue messages on a message bus and process these asynchronously. This netted two benefits: (1) throttled calls to the blockchain service and the little compute power required for this and, (2) decoupled logic of handling complex data processing in another microservice.

Since then, new microservices have been developed for each specialized task: Notifications, CRON jobs, message posting, email sending, analytics, data aggregation, and many more.

Monolithic database

A consideration of microservices is that each service should use its own database to be flexible and scalable. Otherwise, notwithstanding the benefits of having microservices in place, the bottleneck arising from the underlying monolithic database will immediately materialize.

However, this is remedied with the Oracle Autonomous Database, due to the enterprise grade Autonomous Transaction Processing and Autonomous Data Warehouse Cloud Services offered. The Autonomous Database supports database clustering out of the box and can be scaled to a great extent and fully online, alleviating the need for separate database instances.

Furthermore, each microservice is allowed to read from every table, but is only allowed to write to its own ones. The reason for the latter is simple: it would never be clear which microservice was *responsible* for the data in a table. It is not about dead locking, nor dirty writes and reads. Oracle Database is amazingly smart enough not to cause any of these issues unlike other databases. But data could, of course, still be overwritten by other microservices without executing the necessary business rules. Hence, to avoid such a situation, microservices only have write privileges on their owned tables, as mentioned above. A benefit, however, of allowing read access from all tables is that no further API calls nor data transfer between the microservices is necessary to access a piece of information. Despite what conventional wisdom in the microservices space says, this is of great benefit to the retraced system and architecture.

Database: JSON vs relational

Neither of the two data formats, JSON or relational, has been fully adopted throughout the system. Instead, the one format that is best suited for a task is used. Thus, JSON might be used for answers of questionnaire builders, and relational when all data can be clearly identified, and fast access, generic is needed. Since the Oracle Autonomous Database provides easy JSON dot notation and JSON generation, easy future migration possibilities remain open in both directions.

SQL JSON dot notation

When running a database query, Oracle offers easy dot notation access to JSON data when using a table alias in the query and the column has an IS JSON constraint. Thus, the following is valid SQL retrieving the "latitude" attribute from a JSON document:

```
SELECT u.location.latitude
FROM users u;
```

Database connection pools with microservices

The database connections took some time to optimize and set properly for the connection pool in each NodeJS process. An investigation revealed that a consequence of setting the connection pool max size to the NodeJS environment variable UV_THREADPOOL_SIZE necessitated setting the latter to the amount of virtual CPU cores available. For example, on a 4 core Intel machine with Hyperthreading enabled, it would be set to 8. This will allow the Node-oracledb driver to utilize all available cores in parallel to the best extent. Due to the low number of cores, pool min size was set to pool max size (and no increment size) to keep the connection count constant.

The downside of this approach is that every new container will immediately occupy 8 connections to the database. With 10 services at the time of writing this paper, and the total of services and replicas being 30, there are 30 * 16 (8 hyper-thread CPU compute in production cluster), totaling **480** connections**.**

Although there are approaches to handle the pool on the database side (Database Resident Connection Pooling [5]), it is not trivial to setup as it would require a pre-initialization on the database side. Hence, it is recommended to increase the Oracle-CPU count for the database to prevent any performance bottleneck here.

Cloud first and only

At retraced, all services are generally run as cloud services and in the best case, as managed or autonomous services. This principle is to keep the engineering department lean and focused on true value creation through product enhancements, rather than hardware and system maintenance.

The only drawback to this approach is the more complex replication required for a local development setup. However, this is mitigated to a great extent with the use of Docker containers.

Why Oracle?

Ever since the genesis of retraced back in 2019, the need for an extensive tech infrastructure was present. The aim was to stay lean with the available development resources and financial funds. Given these constraints, Oracle stood out as the frontrunner for a cloud blockchain solution, as besides IBM it was the only other cloud provider with a managed blockchain service. As such, and after a successful application to the Oracle for Start-ups program [6], the retraced journey went full steam ahead.

The partnership with Oracle started with immediate effect and a hackathon with several Oracle members was held in Duesseldorf, Germany. Some members even came from Ireland! It was a highly potent session with solution architects from all areas helping us identify the tools and the best setup within the Oracle Cloud for the retraced platform.

Besides the architectural help, Oracle has been very keen to introduce the retraced philosophy to many circles. It started with an invitation to Oracle Open World in San Francisco where retraced co-founder Lukas Pünder was given a chance to speak [7] and other events like the recent Oracle Cloud Infrastructure Customer Summit where he was also a featured speaker [8].

This forthcoming and helpful attitude of Oracle has allowed retraced to constantly innovate and stay ahead with the latest technological advancements.

References

[1] https://sdgs.un.org/goals

[2] https://global-standard.org

[3] https://www.elevatelimited.com

[4] https://www.fairwear.org

[5] https://oracle.github.io/node-oracledb/doc/api.html#drcp

[6] https://www.oracle.com/startup/

[7] https://blogs.oracle.com/blockchain/post/big-week-for-blockchain-at-openworld-2019

[8] https://videohub.oracle.com/media/Oracle+Cloud+Infrastructure+Community+Summit/1_1737bx2h

Author: Peter Merkert, Co-Founder and CTO, retraced

Co-Author: Gerald Venzl, Distinguished Product Manager, Oracle