VERVERICA PLATFORM

Stream processing for real-time business, powered by Apache Flink®

WHITEPAPER

ververica

VERVERCIA PLATFORM

About Ververica

Ververica was founded by the original creators of Apache Flink, a powerful open source framework for stateful stream processing.

In addition to supporting the Flink community, Ververica provides Ververica Platform, a complete stream processing infrastructure, that includes open source Apache Flink.

Ververica Platform makes it easier than ever for businesses to deploy and manage stream processing applications.

About this Whitepaper

This document is organized into 3 sections, your best starting point will depend on your level of familiarity with stateful stream processing and Apache Flink.

In the first section, we'll define **stateful stream processing** and explain why it's a natural **fit for real-time, event-driven products and services.**

In the second section, we'll introduce Apache Flink, **a powerful open source stream processing framework**, share real-world use cases and review the features that set Flink apart as a stream processor.

In the third section, we'll walk through Ververica Platform, an **enterprise-ready stream processing platform,** provided by Ververica, including open source Apache Flink

Ververica Platform is the first solution **purpose-built for stateful stream processing, unifying disparate components to provide seamless deployment and operations from start to finish.**

TABLE OF CONTENTS

The Emergence of Real-Time, Event-Driven Business

- **What is Stream Processing** 4
- **Stream Processing Unifies Data Processing** 6
- **Stream Processing For Batch & Real-Time Data** 7

Stateful Sream Processing with Apache Flink

- **Apache Flink: A High-Performance Open-Source Stream Processor With Powerful APIs and Libraries** 8
- **Real-World Applications Powered by Apache Flink** 8
  - **Alibaba:** Real-time Search Ranking on Singles' Day 8
  - **Netflix:** Real-Time Streaming for Recommendations 8
  - **Uber:** Streaming Analytics for Business 9
  - **ING:** Next-Generation Customer Communication 9
- **Why Apache Flink? A Review of Flink´s Key Features** 10
  - Performance 10
  - State Management 10
  - Fault Tolerance and Exactly-Once Semantics 10
  - Runs Everywhere 10
  - Powerful, User-friendly APIs 11
  - Easy Integrations with the Data Ecosystem 11
  - Easy to Operate 11
  - Sophisticated Time Handling 11

Ververica Platform: Production-Ready Stream Processing with Open Source Apache Flink

- **Ververica Platform is a Complete, Production-Grade Stream Processing Infrastructure** 12
- **Ververica Application Manager: Enabling Stateful Streaming Aware Deployment and Operations** 13
- **Ververica Platform: A Look Inside** 14
  - Unified Deployment on Kubernetes 15
  - Ververica Application Manager:
    Stateful-Streaming-Aware Orchestration 15
    Record-Keeping 17
    Interfaces 18
    Metrics and Logging Integration 19

Conclusion & Next Steps 20

Resources 21

THE EMERGENCE OF REAL-TIME, EVENT-DRIVEN BUSINESS

What is Stream Processing?

In a range of industries, customer interaction has evolved from transactional and product centric to relationship based and services centric:

- **A consumer bank** serving as a place to hold money and to occasionally provide a financial product such as a mortgage or student loan builds a push-based customer messaging platform to proactively notify users of overdraft risk, relevant savings products, account security concerns, and more. [1]
- **Auto insurance companies** offering customers an insurance policy with a fixed monthly rate, develop usage based insurance products where rates are determined by real-time analysis of time spent driving and driving behavior. [2]
- **Car manufacturers** launching a new vehicle every 6 years explore car-sharing services, where ownership is no longer the core model. [3]

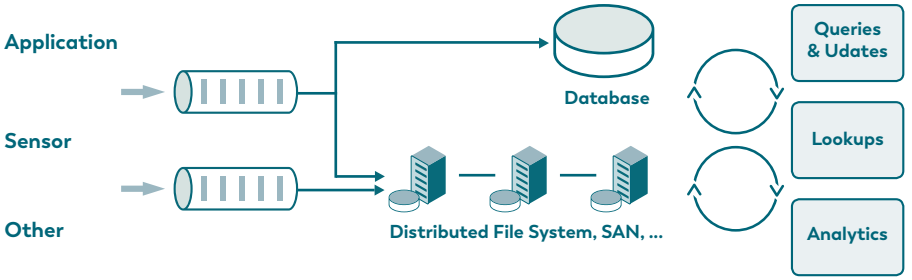
This transformation from a **transactional, product centric model to a relationship based, services centric model,** requires both a new way of thinking and new technological capabilities.

From a technology standpoint, businesses must be able to both ingest and process large quantities of data and respond to insights from data in real time. A delay of minutes or even seconds from data generation to response means missed opportunities to serve customers.

Stateful stream processing has emerged as a technological standard to enable this transformation. Stream processing is the processing of data in motion, in other words, computing on data as it is produced or received.

Many types of data are continuous streams: sensor events, user activity on a website or mobile app and financial trades are examples of data that are created as a continuous series of events over time.

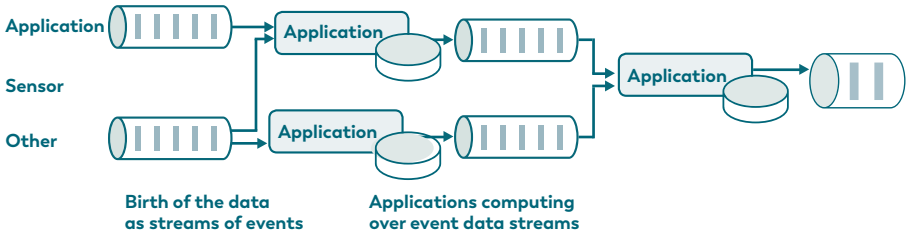
Before stream processing emerged as a standard for processing continuous datasets, these streams of data were often stored in a database, a file system, or other form of mass storage. Applications would then query the stored data or compute over the data as needed. The downside of this approach, broadly referred to as batch 'processing', is the delay between the creation of data and its use for analysis or action.



„Stream processing turns the paradigm around: The application logic, analytics and queries exist continuously and data flows through them continuously“.

Upon receiving an event from a data stream, a stream processing application reacts to the event immediately. The application might trigger an action, update an aggregate, or ‘remember‘ the event for future use.

Stream processing computations can also handle multiple data streams jointly, and each computation over the event data stream may produce other event data streams.



To summarize: stream processing drastically reduces the time from data creation to action in comparison to traditional, batch-oriented architectures. With stream processing, companies can immediately derive and respond to insights in data, taking action when the data is the most valuable.

The stream processing paradigm elegantly addresses many challenges that developers of real-time analytics and event-driven applications face today:

- **Applications and analytics react to events instantly:**
  There’s a minimal lag time between an event occuring -> insight derived -> action taken. Actions and analytics are up to date, reflecting the data when it is still fresh, meaningful, and most valuable.

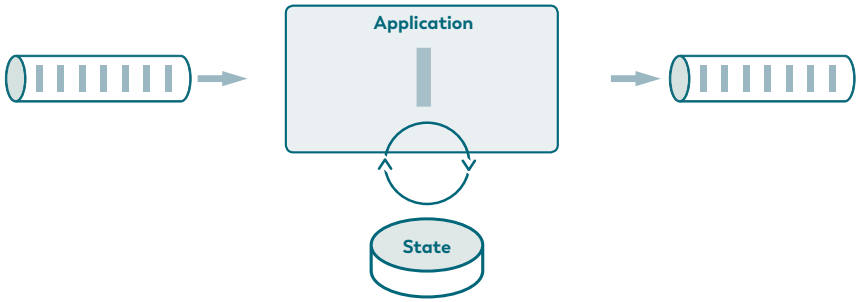
- **Stream processing naturally and easily models the continuous and timely nature of most data:**
  This is in contrast to scheduled (batch) queries and analytics on static data. Incrementally computing updates as new data are available, rather than periodic recomputation of all data, is a perfect fit for the stream processing model.

- **Stream processing decentralizes and decouples the infrastructure:**
  The streaming paradigm reduces the need for large and expensive shared databases. Instead, each stream processing application maintains its own data and application state, which are managed by the stream processing framework. In this way, a stream processing application fits naturally in a micro-services architecture.

Stream Processing Unifies Data Processing, Analytics, and Applications

Stateful stream processing is a category of stream processing in which a computation maintains contextual state, meaning that past events can influence the way future events are processed. Virtually all non-trivial stream processing applications require stateful stream processing:

- **A fraud prevention** application would keep the most recent transactions for each credit card along with parameters from fraud detection models in the state. Every new transaction is scored against this data in the state, then labeled as valid or fraudulent; lastly, the state is updated with the new transaction.
- **An online recommendation** application would store parameters that describe a user's preferences based on previous activity. Every action the user takes, generates an event that updates these parameters.
- **A microservice** that handles a playlist or shopping cart, receives events after each user interaction.



Conceptually, stateful stream processing combines the database or key-value store tables and the application logic, whether an analytics application or an event-driven application, into one tightly-integrated entity.

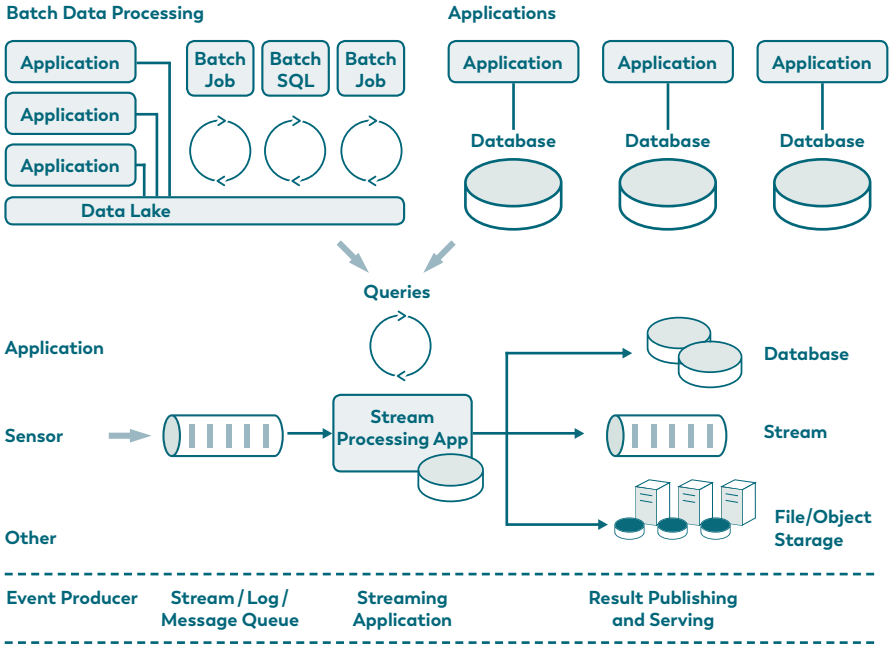
This integration of the state and execution of the application or analytics logic, results in very high performance, scalability, data consistency and operational simplicity.

Apache Flink, which we'll cover shortly, provides first-class support for stateful stream processing, including the ability to handle very large state size, elastic re-scaling of stateful streaming programs, state snapshots (for versioning and application updates), upgrade and schema evolution features.

You probably noticed that we mentioned both real-time analytics and event-driven applications in the previous sections. Are those not two different domains, with processing and analytics implemented via frameworks such as Hadoop or via SQL warehouses and applications implemented via application frameworks and databases?

Not necessarily. Modern approaches to data processing, analytics and event-driven applications have much in common.

Stream Processing for Batch & Real-Time Data Processing and Event Driven Applications



To produce analytical results in real time or near real time, a system must continuously compute and update results with each new record or event.

Modern applications and microservices also operate in an event-driven or 'reactive' fashion, meaning their logic and computation is triggered by events (where events are generated, for example, when a user interacts with a website or mobile app).

This simplifies data infrastructure because many types of systems can be built on a common architecture. In addition, a developer can build applications that use analytical results to respond to insights in the data and take action immediately.

For example:

- Classifying a credit card transaction as fraudulent, based on an analytical model, then automatically blocking this transaction.
- Sending push notifications to users, based on their actions that are scored against existing models about their behavior.
- Automatically adjusting the parameters of a machine based on real-time analysis of its sensor data.

STATEFUL STREAM PROCESSING WITH APACHE FLINK®

A High-Performance Open-Source Stream Processor with Powerful APIs and Libraries

Apache Flink is an open-source stream processing framework built for high-throughput, low-latency, stateful stream processing. In this section, we'll go into more detail on Flink use cases and the key features that enable these use cases.

Based on our experience, working with companies that run some of the largest stream processing deployments in the world, we've learned that a production-grade stream processing framework must offer superior performance and API expressiveness, in addition to operability. The Apache Flink community has long been focused on making Flink the most complete open-source stream processing framework available.

Real-World Applications Powered by Apache Flink

ALIBABA

Real-time Search Results Ranking on Singles' Day

Industry: Retail / E-Commerce

Sample Use Case: Real-Time Recommendations

Alibaba's data infrastructure runs on Apache Flink, with many Flink use cases in production within the company. One example: Singles' Day is China's largest online shopping holiday, comparable to Black Friday and Cyber Monday in the United States but on an even larger scale. [4] User shopping behavior changes considerably on Singles' Day, and existing search result ranking models are therefore no longer relevant. Alibaba uses Flink to train and deploy a Singles' Day-specific search ranking model in real-time, **which led to a 30 % increase in search-to-purchase conversion rate. [5]**

NETFLIX

A Move to Real-Time Streaming for Recommendations and More

Industry: Technology

Sample Use Case: Real-Time Recommendations

Netflix chose Apache Flink as the stream processing technology in its transition from batch ETL to real-time, event-based processing. [6] Flink is a core component in Keystone, Netflix's internal stream processing platform, which provides an interface where users can easily submit ad hoc stream proces-

sing jobs. [7] Flink powers a range of use cases, including a real-time recommendation engine on the Netflix homepage. **For example, Flink made it possible to train machine learning algorithms with real-time data while saving storage costs and easily integrating with other real-time systems.**

UBER

Company-wide Streaming Analytics for Business and Technical Users

Industry: Automotive / Transportation

Sample Use Case: Real-Time Business Intelligence

Uber's business runs on real-time streams of data, from rider requests, to driver location information, to ever-changing traffic conditions and more.

Uber built an Apache Flink based streaming analytics platform called AthenaX, which includes a SQL interface powered by Flink's streaming SQL API, so that business analysts and product managers can submit their own ad-hoc queries without needing help from software engineers. [8][9] In fact, Uber has found that 70% of their production streaming applications can be expressed in SQL. The platform helps Uber to provide more accurate arrival estimates to riders, to maintain a real-time count of orders per restaurant for UberEATS, and more.

ING

Next Generation Customer Communication

Industry: Consumer Finance

Sample Use Case: Fraud Detection

ING is a global bank serving 36 million customers in over 40 countries and uses Apache Flink to power streaming analytics solutions that change how it communicates with and serves its customers. ING has built a Flink-powered platform that provides high-throughput and low-latency and is well-suited for complex and demanding use cases in an international banking organization.

These use cases include customer notifications and real-time fraud detection, all of which require fast data processing and a sophisticated business rules engine or machine learning scoring system. [10]

„At ING, we believe that staying ahead of the game means changing how we interact with our customers, no longer using a traditional model of waiting for the customers to come to the bank through our website or apps, but actively reaching out with information that is relevant to customers, in order to make their financial life frictionless. Many of these changes are driven by reacting to events that are critical to the customer and using streaming analytics to be able to reach out in milliseconds after the event occurs. Apache Flink is key for ING to achieve this." [11]

Ferd Scheepers | Chief Information Architect | ING

Why Apache Flink?
A Review of Flink's Key Features

Now, we'll cover a selection of Flink's differentiating features to explain why companies have chosen Apache Flink to build real-time, event-driven applications.

Performance
Apache Flink's stream processing runtime is designed at all levels to deliver high throughput and low latency. For example, the network and serialization stack has been carefully optimized for performance and shuffles occur inside of Flink without a requirement to interact with an external system between operators.

Flink is highly configurable (while providing sensible defaults for out-of-the-box performance) empowering developers with the dials to find the right balance between latency and throughput. Future releases will further improve the default behavior of Flink in this regard, nearly eliminating the trade-off altogether.

State Management
Flink provides first-class support for stateful stream processing, meaning that Flink is purpose-built for applications where past events can influence the way future events are processed.

As mentioned in the previous section, state is an enabling concept for a majority of interesting use cases, including windowed computations, pattern matching across a stream of events, machine learning model serving, and more. Apache Flink provides powerful abstractions for stateful stream processing.

With pluggable state backends, Flink allows for keeping small state in memory for fast access and also provides a state backend for very large state that exceeds the available memory by using local disks. In addition, the state managed by Flink is part of the integrated fault tolerance mechanism to prevent data loss. This means Flink is capable of efficiently keeping terabytes of streaming applications state in a fault-tolerant way.

Both custom user applications as well as Flink's internal operators and libraries rely on Flink's state management.

Fault Tolerance and Exactly-Once Semantics
Flink is fault tolerant to machine or software failures and provides exactly-once guarantees for managed state. Flink relies on asynchronous, distributed snapshots called 'checkpoints' to provide fault tolerance, ensuring that a computation won't be affected by a machine or software failure.

In other words, an end result won't be missing any data, nor will it count data more than once, even when something goes wrong. Flink can also provide end-to-end exactly-once semantics with external systems such as Apache Kafka that provide a mechanism for transactions.

Runs Everywhere
Flink is widely deployed in the cloud (for example, AWS and Google Cloud Platform) as well as on-premises and integrates with many resources managers, including but not limited, to Kubernetes, Mesos, DC/OS, and YARN. It's also possible to run Flink in a custom setting using a 'standalone cluster' mode.

Powerful, User Friendly APIs
Flink's developer-friendly APIs and libraries support a wide range of use cases and user types. It's truly a one-stop-shop for data-driven applications. In addition to its core DataStream API for stream processing, Flink includes:

- The Table & streaming SQL API for submitting SQL queries on live streams of data, making it possible for business analysts or other non-developers to build real-time applications.
- The FlinkCEP library for complex event processing on live streams of data, which includes full support for Flink's state management features.
- The DataSet API for transformations of static datasets (batch processing).
- The ProcessFunction, a low-level stream processing operation that gives access to the basic building blocks of all (acyclic) streaming applications: events, state, and timers.

Easy Integration with the Data Ecosystem
It's easy to use Flink with many complimentary technologies. Flink offers pre-built connectors to common data sources (systems that send data into Flink) and data sinks (systems that Flink sends data to after processing) such as Apache Kafka, Amazon Kinesis, RabbitMQ, Apache Nifi, Amazon S3 and other file systems, Elasticsearch, Cassandra, and more. In addition, Flink provides a simple but powerful interface for writing custom connectors to external systems.

Easy to Operate
Flink provides very powerful tools for operations:

- A web-based user interface that provides information about the system's status.
- A metrics system that provides built-in reporters for a wide range of commonly-used metris tools such as InfluxDB, Datadog, Graphite; in addition, Flink has built-in system metrics, and users can access these metrics for their own applications.
- A feature called 'savepoints' provides tooling for managing the state of a streaming job; a savepoint is a copy of the state of a Flink job to a specified location, and savepoints can be used to migrate application state between jobs.

Sophisticated Time Handling
Flink makes it possible to process results based on event time, the time that an event actually occurred in the real-world,—even if events arrive at Flink from an upstream system out of event-time order. With awareness for event time in the system, operators such as the windowing operator, the CEP (complex event processing) library, or custom operators can handle events based on real-world event time.

This means, Flink can provide consistent results even in the case of out-of-order or very late arrival of data to the system. This is a big deal in stream processing; many users take advantage of Flink's streaming APIs for use cases traditionally reserved for batch processing, because Flink's event-time handling capabilities offer the same degree of completeness and consistency.

VERVERICA PLATFORM: PRODUCTION-READY STREAM PROCESSING WITH OPEN SOURCE APACHE FLINK®

Ververica Platform is a Complete, Production-grade Stream Processing Infrastructure

Including Apache Flink, the leading stateful stream processor and Ververica Application Manager, a state-aware stream processing orchestration component, Ververica Platform provides a production-ready stream processing infrastructure.

Companies running the largest stream processing deployments in the world have adopted Apache Flink because of its powerful model for stateful stream processing, enabling them to derive insights and take action on data, the very moment it's generated and when it's most valuable.

Ververica Platform is a purpose-built stateful stream processing architecture. **With Ververica Platform, operating powerful data systems is easier than ever before.** Ververica Platform and Ververica Application Manager offer an entirely new experience for deploying and managing stream processing applications.

It's our mission at Ververica to ensure that developers invest their time in improving their stream processing applications, not on maintenance and infrastructure.

Ververica Platform provides a stable, easy-to-use stream processing platform so that developer teams can focus exclusively on their use cases.

In this section, we'll first describe the 'why' behind Ververica Platform, and then we'll walk through the 'what' and the 'how'.

Let's walk through a few of the steps required to deploy and manage a stateful stream processing application in production, beyond the development of the core application logic.

- **Resource management**
  A developer must decide how to integrate Flink with the hardware at their disposal. A majority of Flink users deploy Flink jobs on shared resources, rather than on a dedicated cluster, which means integrating with a resource management platform such as Kubernetes, YARN or running dedicated VMs in a cloud environment, such as AWS.

- **Centralized logging and metrics:**
  This might require integration with a logging and metrics infrastructure that's already in use at a company or setting up logging and metrics from scratch if no such systems exist. A tight integration with these systems is crucial for debugging performance and correctness issues. It'll be necessary to set up servers for these components, also.

- **CI / CD pipeline**
  The streaming application needs to integrate with an organization's existing deployment infrastructure in order to maximize productivity.

- **Stateful application management**
  Over time, developers must update and improve business logic, fix bugs, and upgrade to new framework versions, all while ensuring that the application state is preserved throughout the process.

Application Manager: Enabling Stateful Streaming Aware Deployments & Operations

Even when using an open source framework like Flink, which provides a wide range of operability features, rolling out a stream processing infrastructure and managing continuous applications in production is a time intensive process.

This is the challenge that Ververica Platform addresses.

We designed Ververica Platform based on our collaboration with production users of Apache Flink, such as Netflix, Uber, Alibaba, ING and more. Ververica worked closely with these companies as they built large-scale stream processing platforms, and gained insights during this process into best practices for migrating existing data workloads onto real-time stream processing platforms as well as managing long-running streaming applications.

Ververica Platform makes it possible for anyone to start with a best-in-class stream processing infrastructure and tightly-integrated components.

The demand from organizations for a production-ready stream processing infrastructure was the motivation for the key component in Ververica Platform, the **Ververica Application Manager.**

Before Ververica Platform, a stateful stream processing infrastructure was made up of disparate tooling for deployment as well as for state management. Upgrading, scaling, or migrating applications required careful planning and tedious manual work (and often custom-built tooling) for both state management and resource management.

Application Manager unifies these two worlds by providing one tightly-integrated tool that manages application state and deployment together.

Ververica Application Manager is the component that makes stateful application lifecycle management easy and frictionless. It also provides a documented history of application versions, which, in addition to being a helpful troubleshooting tool, is a legal requirement in certain industries.

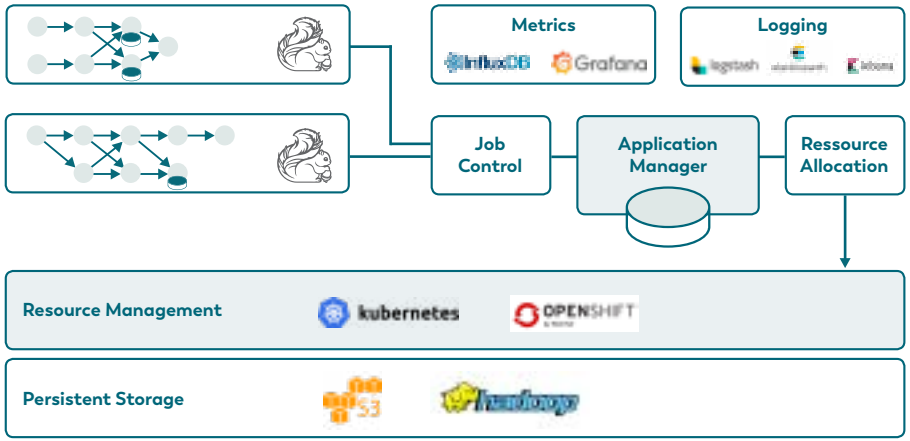
„Ververica Application Manager is the core orchestration component in Ververica Platform. It's what allows developers to easily manage, monitor, and configure streaming jobs without worrying about the underlying infrastructure. Ververica Application Manager is stateful-streaming-aware, thus simplifying common stream processing operations tasks such as upgrading applications in a consistent manner".

Next, we'll go into detail about how Ververica Platform is packaged and describe its capabilities.

Ververica Platform: A Look Inside

Ververica Platform is packaged as a set of Docker containers provided by Ververica. After installation using tools that are provided as part of the distribution, customers will have Ververica Application Manager as well as metrics and logging collection systems running in their Kubernetes cluster.

Ververica Platform ships with components for metrics and logging, but these components are mostly included for demonstration purposes or for use during pre-production testing. In production, it's a best practice to integrate with your organization's existing metrics and logging infrastructure. Ververica Platform provides configuration options for integration with existing systems.



As the central orchestration and coordination component, Ververica Application Manager will allocate Flink containers on Kubernetes in order to execute stateful streaming jobs.

Ververica Application Manager allocates a dedicated Flink cluster within Kubernetes for each Flink application, a best practice for stability and resource isolation.

Flink applications launched by Ververica Application Manager will be configured to be integrated with persistent storage, metrics, and logging plus other systems like Zookeeper or Kafka, depending on the requirements.

Unified Deployment on Kubernetes

Kubernetes is the common resource manager of Ververica Platform components. As mentioned above, all components are containerized using Docker, and Ververica Platform's setup tooling enables easy deployment of the platform.

Kubernetes allows for deploying complex, highly integrated, and distributed infrastructure components on a wide range of hardware: all major cloud providers support Kubernetes. Red Hat OpenShift and others provide solutions for on-premises deployments.

Without a resource manager like Kubernetes, building out a stream processing infrastructure, would entail manual setup and configuration of your Logstash, Kibana and Elasticsearch servers; your Grafana server; your InfluxDB server; and all other necessary components. In this way, Ververica Platform significantly shortens the infrastructure roll-out process.

Ververica Application Manager allows users to either deploy the Apache Flink Docker containers included with Ververica Platform, or to use custom containers built on top of the provided ones, meaning that users can include custom dependencies into their Flink images, or run Flink on their own base containers.

Ververica Application Manager: Stateful-Streaming-Aware Orchestration

The Ververica Application Manager is the main orchestrator between a user's requests, Kubernetes, and Apache Flink.

It provides an abstraction over streaming jobs, which in Ververica Application Manager are called 'Deployments'. A Deployment specifies:

- A Flink streaming job's code location (as a jar file)
- The target infrastructure
- Flink configuration details
- Resource requirements

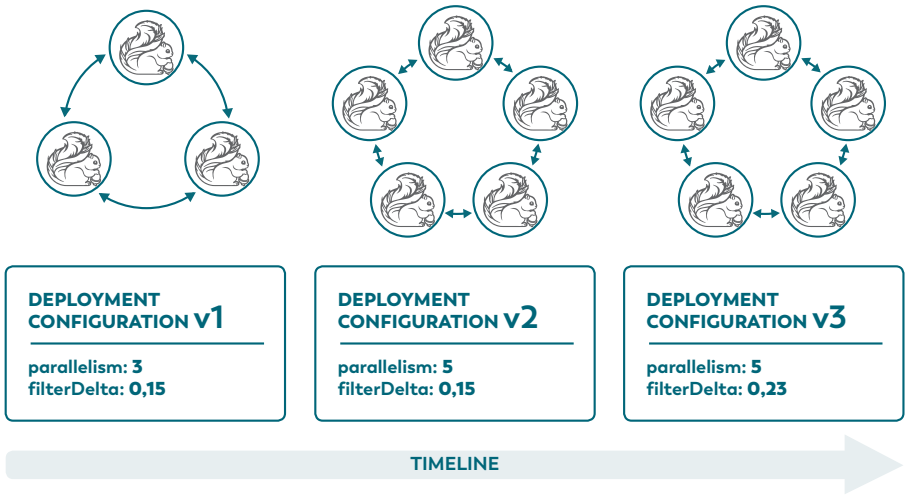
...and many other parameters.

Based on these specifications, an internal controller component of the Ververica Application Manager schedules an Apache Flink cluster on Kubernetes. Once that Flink cluster is available, an application is submitted and continuously monitored.

As soon as the specification of the deployment changes, the controller will detect that change and trigger the required operations, to achieve the new desired state.

This powerful model facilitates actions such as scaling up a Flink job to more machines by simply changing a setting in the deployment specification. Without the Ververica Application Manager, a user would need to manually perform a series of operations with Flink's APIs, manually handle error cases, and manually store metadata about the operations, to complete this task.

All changes to a deployment controlled by the Ververica Application Manager, such as updates to the configuration, updates to the job itself, or changes to resource allocations, are automatically reflected in the Flink cluster, orchestrated by the Ververica Application Manager.



In case of failures, the Ververica Application Manager will work to keep your stream processing applications going. For example, when performing a stateful upgrade of a Flink application, if creating a copy of the state fails in Flink, the Ververica Application Manager will keep the old Flink application running until it's able to create a full copy of the Flink state.

Because the Ververica Application Manager is aware of the stateful nature of Flink applications, it also takes care of keeping the state throughout changes to the deployment.

Configuration updates or parallelism changes will be carried out in a stateful way behind the scenes, meaning that in-flight data is not lost.

For example, a streaming application analyzing user session windows will not lose in-flight data as a result of scaling the application up to more machines.

Ververica Application Manager: Record-keeping

In addition to the declarative control model, the Ververica Application Manager keeps track of the Flink applications triggered, along with their configuration changes, savepoints, and all controller events in an event log.

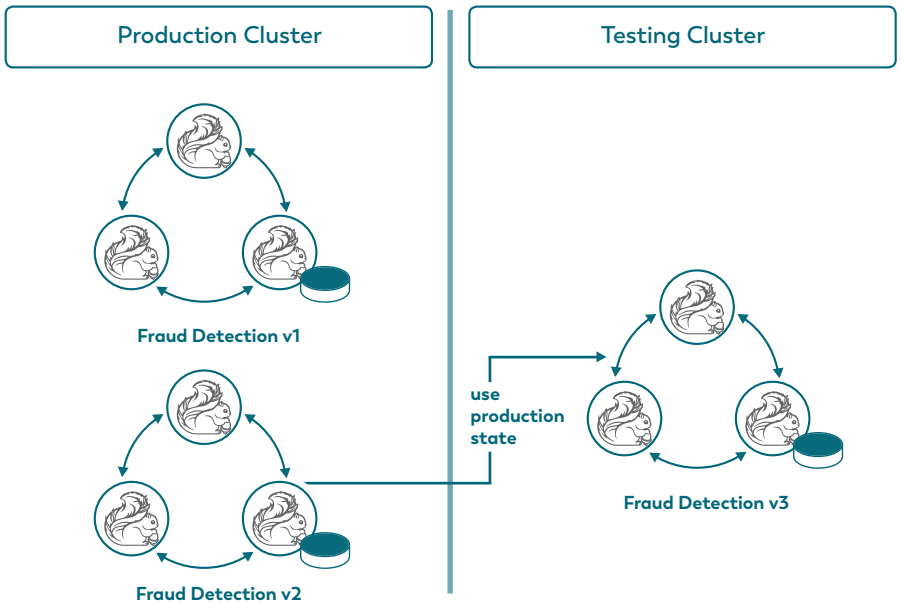
With **a history of configuration changes,** the Ververica Application Manager keeps a 'paper trail' of how the configuration has evolved. This allows different roles such as engineers and infrastructure operators to trace performance regressions or misbehavior back to specific configuration changes.

A managed **history of savepoints** allow a user to 'travel in time' by resetting a job back to a savepoint, thus setting the state to the time of the savepoint.

The Ververica Application Manager also supports forking a deployment with a savepoint. This allows for advanced scenarios such as deploying an updated Flink job on a pre-production environment, while using the state from a current production job.

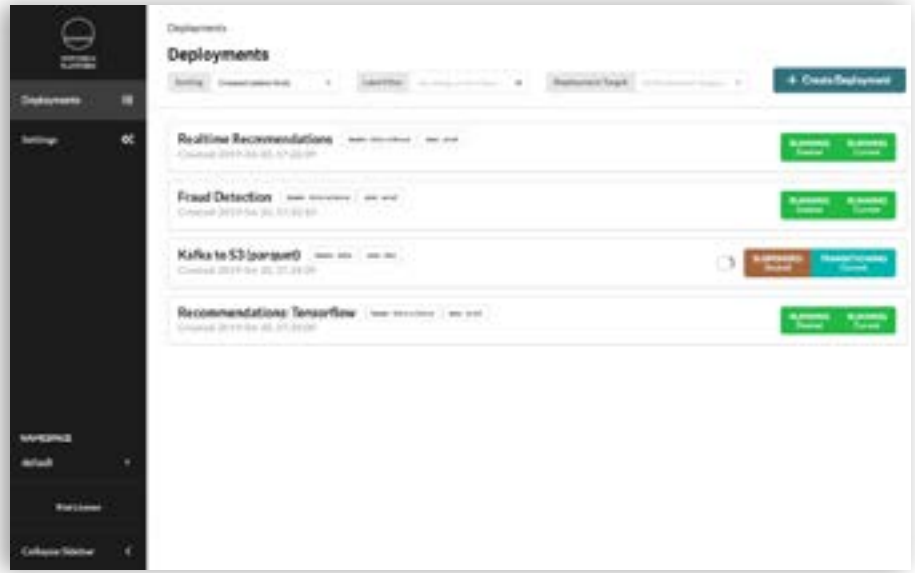
Ververica Application Manager's **event log** reports to the user on asynchronous operations such as deployment upgrades and the creation of savepoints.

Failures during such operations, but also in steady state, will be reported in the status of the resource and an event log. Therefore, operators of Flink applications can periodically check the event log or resource status, to see what happened recently within an application.

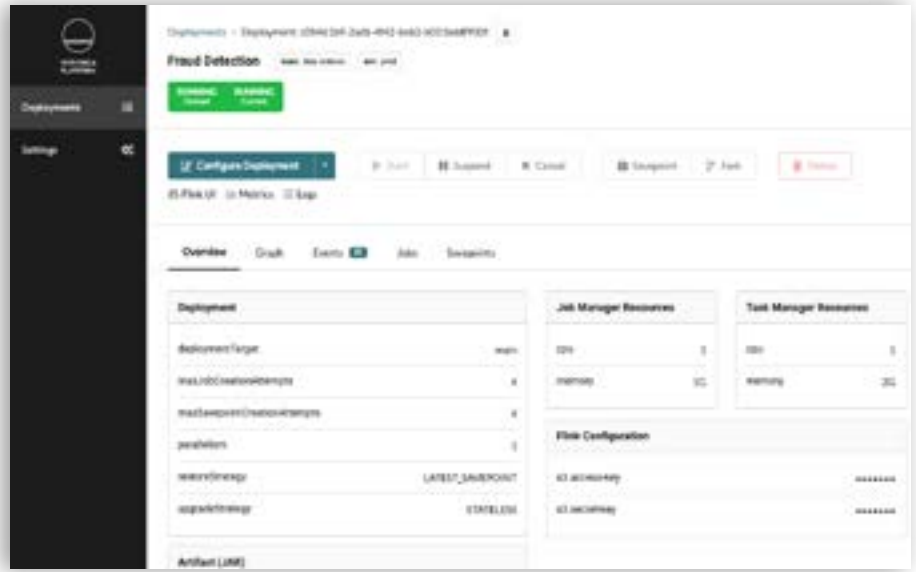


Ververica Application Manager: Interfaces

The Ververica Application Manager provides both a web-based user interface and a REST API. The user interface allows developers to easily monitor, control, and configure stream processing applications, without worrying about the underlying infrastructure.



The homepage lists all deployments and allows a developer to create new ones. The status of each deployment is shown in a 'Status' field.



The Overview page for each deployment contains controls for changing the application status, triggering savepoints, or forking deployments. There are also links that provide easy access to the Flink user interface, metrics and logging systems and tabs for the event log, application history, and savepoint overview.

In addition to the web-based user interface, the Ververica Application Manager provides a REST API. All features available in the web user interface are also available through this API.

We provide an API specification based on Swagger so that users can easily build their own integrations on top of the API.

One common use case of the REST API is the integration with automated deployment systems, for example, as part of a CI / CD pipeline.

This REST API is independent of Flink versions and is properly versioned and documented. Integrating internal deployment systems with the Ververica Application Manager is recommended for the following reasons:

- The declarative model of the Ververicas Application Manager does not require integrators to worry about error handling, timeouts, or synchronization between systems.

- The Ververica Application Manager API is independent of Flink releases, meaning that integrations remain stable through Flink releases and don't need to be revisited with each Flink release.

Ververica Application Manager: Metrics and Logging integration

All Flink clusters created by the Ververica Application Manager are automatically configured to use a centralized metrics and logging system, the Ververica Application Manager forwards a unique deployment and job identifier to these systems. This allows a user to set the scope of metrics and logs to specific time frames of a deployment.

The performance implications of certain configuration parameters are very easy to correlate via the Ververica Application Manager, because both the configuration changes and metrics are tracked and stored by a common system.

For example, there is a configuration parameter in Flink for managing the tradeoff between throughput and latency. With the Ververica Application Manager, its easy for a user to test different parameters in Flink and measure the impact on throughput and latency, arriving at the configuration that fits best, given individual application requirements.

Ververica Platform does not require customers to use the integrated logging and metrics components. **For production use, we provide configuration options and integrations with existing deployment and logging solutions.**

CONCLUSION AND NEXT STEPS

We hope you finished this report with an understanding of 3 key points:

- The transition from transactional and product-centric business models to relationship-based and services-centric business models is powered by new technologies such as Apache Flink, an open source framework for stateful stream processing.

- Stream processing with Apache Flink has enabled global enterprises in a range of industries to realize measurable business value by acting on their data in real time.

- Ververica Platform, built and supported by the original creators of Apache Flink at Ververica, provides an out-of-the-box stream processing infrastructure. Ververica Platform makes it easier than ever for businesses to deploy and manage stream processing applications in production.

  Of course, this was just an introduction to stream processing with Apache Flink and Ververica Platform. If you'd like to learn more, plesae reach out or refer to our resources on the next page.

Ready for the next steps?

Download a free Ververica Platform trial sandbox at:
ververica.com/download

If you'd like to learn more about Ververica Platform, we recommend the Ververica Platform documentation: docs.ververica.com

If you'd like to get in touch with someone from our team, send an email to: platform@ververica.com or fill out our contact form: ververica.com/contact

We'd love to hear from you!

RESOURCES

[1] https://berlin.flink-forward.org/kb_sessions/taking-away-customer-friction-through-streaming-analytics/

[2] https://www.iamagazine.com/markets/read/2017/04/17/what-s-next-for-usage-based-insurance

[3] https://www.mckinsey.com/business-functions/sustainability-and-resource-productivity/our-insights/an-integrated-perspective-on-the-future-of-mobility

[4] https://www.forbes.com/sites/augustrick/2017/12/03/singles-day-has-eclipsed-cyber-monday-and-black-friday-but-they-all-share-the-same-goal/

[5] http://www.dataversity.net/year-blink-alibaba/

[6] https://www.infoq.com/articles/netflix-migrating-stream-processing

[7] https://sf-2017.flink-forward.org/kb_sessions/keynote-stream-processing-with-flink-at-netflix/

[8] https://eng.uber.com/athenax/

[9] https://sf-2017.flink-forward.org/kb_sessions/athenax-ubers-streaming-processing-platform-on-flink/

[10] https://berlin-2017.flink-forward.org/kb_sessions/fast-data-at-ing-building-a-streaming-data-platform-with-flink-and-kafka/

[11] https://berlin-2017.flink-forward.org/kb_sessions/taking-away-customer-friction-through-streaming-analytics/

CONTACT US

contact@ververica.com

@VervericaData

www.ververica.com

DOWNLOAD FREE TRIAL

ververica