ALLIED BizTech SOLUTIONS

FIELD GUIDE · 2026

Rebuild Segment yourself.
The plan we follow.

For SaaS teams on 100K to 5M+ monthly tracked users who want to own the pipeline between their app and their tools, and stop paying per MTU.

- → Is it worth it, at your size
- → What to rebuild and what to drop
- → The architecture, two paths
- → Migration and go-live checklists
- → The event schema
- → A week-by-week build plan
- → Identity and delivery guarantees
- → Where teams get stuck

Segment charges per user. *A pipeline charges per server.*

Segment bills per monthly tracked user, so every new user raises the bill. The job underneath is a forwarding pipeline. RudderStack, its open-source equivalent, does that job on your AWS, and its cost follows servers, not users.

SIZE	SEGMENT A YEAR	REBUILT: YEAR ONE / AFTER	SAVED IN YEAR ONE
25K MTU · Team plan	~$1,440	~$12,600 / $1,000	Stay on Segment
100K MTU · Business plan	~$14,400	~$13,600 / $2,000	~$800; $12K a year after
250K MTU · Business + add-ons	~$48,000	~$17,600 / $6,000	~$30,400
1M MTU · Enterprise, Functions + Reverse ETL	~$144,000	~$29,600 / $18,000	~$114,400
5M+ MTU · Enterprise + custom	~$420,000+	~$84,600 / $58,000	~$335,400

Segment's Business pricing is not published; these figures (May 2026) extrapolate from the published Team tier and mid-market buyer reports, so put your own invoice in their place. "Rebuilt" runs RudderStack on AWS EC2 with Postgres, managed for $2,400 a year, plus our fixed build price: $11,600 up to 1M MTU, $26,600 (the top of the $18,300–$26,600 scale tier) from 5M. If your own team builds it, put your engineering cost in that place.

THE RULE OF THUMB

Below about 100K MTU, stay on Segment. At 100K, year one comes out ~$800 ahead and $12K a year comes back after. **At 250K MTU, the rebuild saves ~$30K in year one and ~$42K a year after.** From 1M MTU, year one alone saves more than $110K.

Who you need

- **A senior engineer** at home with Kubernetes, Postgres and the APIs of your tools. Production-grade RudderStack ops, destination configs and transformations take typically 2–3 weeks of that person's time.
- **Your data or growth lead**, two hours a week: the tracking plan, the destinations that matter, the parallel-run sign-off.
- **An owner after launch**: upgrades, destination API changes, failed deliveries and deletion requests.

02 · SCOPE

Rebuild the 15–25% you use. *Drop the rest.*

RudderStack is purpose-built as a Segment alternative: the same API, the same source and destination model. The work is moving what you already use.

REBUILD ONE FOR ONE

- **The SDK calls**: analytics.identify, .track, .page and .group, unchanged
- **Sources**: web, server and mobile, in JS, Node, Python, Java, Swift, Kotlin and Go
- **Destinations**: HubSpot, Intercom, Mixpanel, Salesforce, Snowflake, BigQuery and more, with field mapping
- **Transformations** in JavaScript or Python, per destination
- **Reverse ETL**: scheduled syncs from Snowflake, BigQuery or Redshift to your tools
- **Identity resolution**: anonymous-to-known user-stitching
- **PII filtering, GDPR and CCPA**: field-level redaction, suppression, deletion
- **Schema validation** on track events, with violation alerts

REBUILD, SIMPLER

- **Audiences**: basic ones in RudderStack; complex ones from your warehouse, by Reverse ETL
- **Functions**: the same JavaScript model, as transformations
- **Destinations**: 250+ against Segment's 350+; a missing one takes 1–2 days

DROP ON PURPOSE

- **Twilio Engage**: a separate project if you use it
- **Protocols**: Atlan or Collibra, or the basic schema validation above
- **Privacy Portal**: Transcend, OneTrust or DataGrail direct

ADD WHAT PER-MTU PRICING DISCOURAGES

- No per-MTU fee: user growth is product success, not a vendor tax
- Events stay in your VPC and your tenancy
- Replay any event range from S3 to any destination

Two paths. *The same Segment SDK.*

Nine times in ten, self-hosted RudderStack does the whole job. A custom Kafka router is the other path, for unusual event shapes or extreme scale. Both keep every analytics call in your code as it is.

PATH A · 90% OF CASES

RudderStack self-host

- Apache 2.0, Segment-SDK-compatible
- 250+ destinations pre-built
- Transformations in JavaScript or Python
- Reverse ETL, warehouse to tools
- Deployed on your AWS with Helm

PATH B · 10% OF CASES

Custom Kafka router

- For unusual event shapes or extreme scale
- Kafka, with transformations in Go
- Tighter control, slower to build
- Falls back to RudderStack if scope creeps

SHARED · ALL DEPLOYS

Your infrastructure

- **Postgres**: event metadata and the schema registry
- **S3**: long-term event storage and replay
- The data plane on EC2 or EKS
- OAuth credentials per destination

Decisions to make in week one

DECISION	OUR DEFAULT	WHY
Which path	RudderStack first	Destinations, retries and deletion APIs come built
Where it runs	Your AWS account, on EKS, inside your VPC	Events never leave your tenancy
The tracking plan	In Git, validated at ingest	Bad events stop before every tool stores them
The user key	One userId, your database ID	Every destination stitches users on the same key (page 7)
Raw events	Every event to S3, before any destination	Any outage or new tool is a replay, not a loss
Sign-in	Your SSO from day one	Offboarding happens where it already does

License: RudderStack is Apache 2.0. Run it, change it and fork it with no restriction. Keep every Helm chart, transformation and destination config in Git from day one.

Four calls, one user key. *A plan for every event.*

A pipeline stores little of its own, but every tool downstream inherits its schema. Agree the tracking plan and the identity rules before the first writeKey changes, because every destination, audience and report reads from them.

RECORD · STORE	KEY FIELDS
Events · S3 + warehouse	messageId, type (track, identify, page, group), event name, userId, anonymousId, properties, context, timestamp, receivedAt
Tracking plan · Git + Postgres	event name, required properties and their types, version, owner, allowed sources
Sources · Postgres	source, type (web, server, mobile), writeKey, tracking plan, environment
Destinations · Postgres	destination, OAuth credentials, event filter, field mapping, transformation
Identities · warehouse	anonymousId, userId, email as a trait, first seen, last seen
Accounts · warehouse	groupId, traits, the userIds that belong to it
Deliveries · Postgres	messageId, destination, status, attempts, last error, next retry
Deletion requests · Postgres	userId, regulation (delete or suppress), requested at, destinations confirmed

TWO RULES THAT SAVE A REBUILD

One userId, from your database: never an email and never a vendor's ID. Emails change; a database ID does not, and every tool can merge on it. **Name events from the plan, not from code**: object and action in the past tense (Order Completed), typed properties, and changes reviewed like code. A renamed event breaks every report that counts it.

Four weeks, *parallel run included.*

This is the order we build in. Each week ends with something your team can check, and the third runs both pipelines side by side before Segment goes quiet.

WEEK	BUILD	DONE WHEN
1	**Inventory and foundations.** List every source, writeKey, destination, transformation, Function and Reverse ETL job; export the tracking plan. EKS, Postgres and S3 in Terraform; RudderStack with Helm, behind your SSO.	The inventory is signed off; the team signs in
2	**Sources, destinations, transformations.** Source configs migrated; destinations wired with fresh OAuth credentials and field mapping; Functions ported to transformations; tracking-plan validation on; any missing destination built.	Staging events reach every destination, correctly mapped
3	**Parallel run.** Both pipelines receive production events for a week. Compare counts per event and per destination daily; set up identity resolution, Reverse ETL and deletion requests; work a fix list.	Counts match per destination; a test deletion reaches every tool
4	**Cutover.** New writeKey and endpoint URL per source, redeployed. Segment sources switched off; the runbook handed over.	Every destination receives from RudderStack alone

AFTER CUTOVER

Keep the Segment workspace **until your renewal date**, with its sources off, then cancel. Its debugger and history stay one click away while the team settles in, and nobody is in a hurry.

The writeKey swap

RudderStack's client libraries take the same calls as Segment's. Change the writeKey and the endpoint URL, redeploy, and existing web, mobile and server instrumentation works unchanged. Mobile apps pick it up with their next release, so ship that release early and keep both pipelines open for the window.

Clean events, one user, *every event delivered.*

Forwarding events is the easy part. A pipeline earns its keep on three things: events that match the plan, one user per person across every tool, and delivery that survives an outage at the other end.

1. **Validate at the door.** Check every track event against the tracking plan as it lands. Do not drop violations silently: tag them, route them to a quarantine destination and alert the owning team. A bad event fixed at the source is fixed in every tool at once.
2. **Stitch identities deterministically.** Call identify on signup and login, and reset on logout. Merge anonymousId into userId, and use email only as a trait. Deterministic rules on email, userId and anonymousId hold up in an audit; probabilistic matching does not.
3. **Send group for accounts.** B2B tools report by company. One group call per session, with a stable groupId from your database, keeps account traits current in every destination.
4. **Plan for at-least-once.** Destinations fail, throttle and time out. Retry with backoff per destination, park what still fails in a dead-letter queue, and let each tool dedupe on messageId. Never assume order: timestamps decide.
5. **Make every outage a replay.** Every raw event goes to S3 before any destination. When HubSpot is down for an afternoon, or a new warehouse needs a backfill, replay the range from S3 to that one destination.

Deletion is delivery too

A GDPR or CCPA deletion request is an event that must reach every tool. RudderStack's user-deletion APIs propagate it; record each destination's confirmation, so the audit trail is yours and complete.

Measure it

Track events a day per source, delivery success and lag per destination, plan violations a week per team, and the dead-letter queue, next to the AWS bill. Those numbers prove the pipeline works and the saving is real.

Move it, launch it, *run it.*

Migration

- ☐ Inventory sources, destinations, transformations, Functions and Reverse ETL jobs
- ☐ Export the tracking plan into Git
- ☐ Check destination coverage; build the missing one
- ☐ Fresh OAuth credentials for every destination
- ☐ Port Functions to JavaScript or Python transformations
- ☐ Move audiences to the warehouse, with Reverse ETL
- ☐ New writeKey and endpoint per source; mobile release early
- ☐ Run both pipelines for a week and compare counts

Go-live

- ☐ Every source on the new endpoint
- ☐ Delivery success per destination at parity
- ☐ Dead-letter queue empty, with an alert on it
- ☐ A test deletion confirmed by every destination
- ☐ The team signing in through your SSO
- ☐ Rollback ready: the old writeKey one config change away

Running it

- ☐ **Always on:** delivery and lag alerts per destination, security patches
- ☐ **Weekly:** plan violations by team, failed deliveries, the dead-letter queue
- ☐ **Monthly:** RudderStack upgrades through staging first; destination API changes
- ☐ **Quarterly:** cost review, the AWS bill against the Segment bill you replaced; a tracking plan review
- ☐ **Yearly:** a deletion audit and an access review

Who owns it

Name one person before launch. Every tool downstream trusts the pipeline; someone has to answer when a destination goes quiet and nobody notices for a week.

Five ways a rebuild *stalls.*

1. Switching to another enterprise CDP

mParticle and Tealium are competent, with comparable per-MTU enterprise pricing plus a migration. The same wrapper question survives the move.

2. Reverse ETL alone

Hightouch and Census sync the warehouse to your tools, and only that. Most teams need forward sync and Reverse ETL; RudderStack does both.

3. RudderStack Cloud past 500K MTU

RudderStack Cloud is good for teams under 100K MTU. Above ~500K, its pricing approaches Segment's, and self-hosting is far cheaper.

4. "It's just webhooks"

It is, plus 250+ destination integrations, retries, dead-letter queues, identity resolution and GDPR controls. RudderStack gives you all of that for free; a home-built router starts from none of it.

5. A self-host without the ops

The install is quick. Production-grade ops, destination configs and transformations are typically 2–3 weeks of senior time. Plan for them, along with the parallel run.

And one that is not a mistake: going direct

With only 2–3 destinations, calling their APIs directly is the cheapest path. Above 5 destinations, keeping point-to-point integrations alive costs more than running RudderStack.

RATHER HAVE IT BUILT?

We build this exact plan. *Fixed price.*

The same scope, stack and four weeks, delivered by a senior team that has shipped 250+ products.

$11,600 fixed up to 1M MTU and 30 destinations; $18,300–$26,600 for 5M+ MTU, multi-region; agreed before we start

---

Live in 3–4 weeks, with a one-week parallel run next to Segment before cutover

---

The same SDK calls: a new writeKey and endpoint; sources, destinations and transformations migrated

---

90-day warranty, and the code and Helm config are in your GitHub from day one

---

We can run it for $2,400 a year: monitoring, monthly upgrades and SLA support

alliedbiz.tech/rebuild/segment-alternative

Allied BizTech · 15+ years · ISO 9001:2015 · 250+ products shipped. Segment and Twilio are trademarks of their owner, named here for identification only; this guide is independent and not endorsed by Twilio Segment. Prices as of May 2026.