ALLIED BizTech SOLUTIONS

FIELD GUIDE · 2026

Rebuild Salesforce yourself.

The plan we follow.

For sales orgs of 50 to 500+ seats who want to own their accounts, pipeline, forecasts and reports, keep every integration working, and stop paying per seat.

- → Is it worth it, at your size
- → What to rebuild and what to drop
- → The architecture, in three layers
- → Migration and go-live checklists
- → The data model
- → A week-by-week build plan
- → Permissions and reporting parity
- → Where teams get stuck

Sales Cloud charges for every rep. *Postgres charges for data.*

Salesforce lists Pro Suite at $100, Enterprise at $165, Unlimited at $330 and Einstein 1 Sales at $500 per seat per month, with Sales Engagement, CPQ and Maps as add-ons. A rebuild has no seat price: the AWS bill follows your data volume.

ORG	SALES CLOUD A YEAR	REBUILT: YEAR ONE / AFTER	SAVED IN YEAR ONE
25 seats · Pro	~$49,500	~$53,800 / $14,000	Stay on Salesforce
50 seats · Enterprise	~$198,000	~$53,800 / $14,000	~$144,200
100 seats · Enterprise + Sales Engagement	~$396,000	~$59,800 / $20,000	~$336,200
200 seats · Unlimited + Einstein 1	~$1,200,000	~$121,100 / $40,000	~$1,078,900
500+ seats · enterprise contract	~$3,000,000+	~$247,100 / $96,000	~$2,752,900

Sales Cloud list prices, May 2026. "Rebuilt" runs Postgres on RDS, a Next.js front end and a Salesforce-compatible API, plus our fixed build price: $39,800 up to 100 seats, $81,100 for 100–300, $151,100 above 300 (the tops of those tiers). If your own team builds it, put your engineering cost in that place.

THE RULE OF THUMB

At 25 seats, stay: Salesforce's ecosystem outweighs the cost. **At 100 seats, the rebuild saves ~$336K in year one and ~$376K a year after.** From 50 seats, year one alone saves more than $140K.

Who you need

- **Senior full-stack engineers** at home with Postgres, row-level security, the Salesforce REST API and large data migrations. Our team ships this in 10–14 weeks.
- **Your Salesforce admin and a RevOps lead**, a few hours a week: the objects, roles and reports that matter, the pilot team.
- **An owner after launch**: upgrades, access reviews, integrations and the reports.

02 · SCOPE

Rebuild the ~20 features in daily use. *Drop the rest.*

Each edition ships hundreds of features. A sales org uses about 20 every day: the 15–25% it pays for and uses. Those are the project, built to your workflow.

REBUILD ONE FOR ONE

- **Leads, contacts, accounts, opportunities**: your hierarchy, custom fields and custom objects
- **Activities**: calls, emails, meetings; Outlook and Gmail sync, email-to-record
- **Pipeline and forecast**: your stages, probability %, close-date logic, weighted forecast
- **Reports and dashboards**: the ~15 saved reports leadership opens
- **Workflow automation**: triggers, actions, time-based and scheduled
- **Approvals**: multi-level, conditional, with an audit trail
- **Lead routing and territories**: round-robin, weighted, territory rules
- **SSO, SCIM, roles and permissions**
- **Outreach, Gong, Marketo**: unchanged, through the Salesforce-compatible API

REBUILD, SIMPLER

- **AppExchange**: the 2–5 apps you use, on direct integrations
- **Einstein AI**: Anthropic or OpenAI direct, plus rule-based scoring
- **Apex and Lightning**: TypeScript or Python in your codebase

DROP ON PURPOSE

- **CPQ**: PandaDoc or DocuSign CLM; basic quotes in scope
- **Predictive ML scoring**: MadKudu or your data team
- **Service Cloud, Marketing Cloud, Pardot**: separate rebuilds
- **Trailhead and admin certifications**

ADD WHAT PER-SEAT PRICING DISCOURAGES

- New reps at no seat fee
- SQL on every record, no API rate limits
- Workflows in code: reviewed, tested, versioned

Three layers. *Integrations untouched.*

A CRM on Postgres with a REST API that answers the way Salesforce's does. Tools built for Salesforce talk to the same endpoints with the same auth, and never notice the change. Migration becomes a back-end swap, not a vendor renegotiation.

LAYER 1 · REPS USE

The sales workspace

- React UI, built to your workflow
- Lead, contact, account, opportunity views
- Pipeline and forecast dashboards
- Activity: email, call, meeting
- Custom objects
- Mobile web app (PWA)

LAYER 2 · DATA + WORKFLOW

Postgres-backed CRM

- Every object and custom field
- Workflow engine: triggers, actions
- Multi-level approvals
- Lead routing and territories
- Forecast engine
- Audit log of every change

LAYER 3 · API + INTEGRATION

Salesforce-compatible

- REST API on Salesforce's surface
- Outreach, Gong, Marketo unchanged
- ZoomInfo, Apollo, 6sense
- DocuSign, PandaDoc
- Stripe or billing sync
- Webhooks downstream

Decisions to make in week one

DECISION	OUR DEFAULT	WHY
Where it runs	Your AWS account; GovCloud where FedRAMP applies	Data residency fully under your control
API surface	Only the endpoints and queries your integrations call	Log a week of real traffic first; build to that, not to all of Salesforce
IDs and names	Salesforce's record IDs and field API names, kept	Integrations and old links resolve without a mapping table
Access	Postgres row-level security, enforced under the API too	The UI and every integration see the same rules
Sign-in	Your SSO, with SCIM provisioning	Offboarding happens where it already does

Compliance: on your own AWS account you inherit AWS's SOC 2, ISO 27001 and HIPAA-eligible posture (with a BAA). Write the SOC 2 control documentation as you build.

Ten tables *carry the whole product.*

Model these before writing any screen. Keep Salesforce's 18-character ID on every imported row, so the import can run twice, integrations resolve, and every record traces back to its source.

TABLE	KEY COLUMNS
`accounts`	id, sfdc_id, name, parent_id, owner_id, territory_id, record_type, industry, custom fields as typed columns
`contacts`, `leads`	id, sfdc_id, account_id, name, email, owner_id, status, source; leads: converted_contact_id, converted_at
`opportunities`	id, sfdc_id, account_id, owner_id, stage, probability, amount, close_date, forecast_category; line items in `opportunity_lines`
`activities`	id, sfdc_id, type (call, email, meeting, task), who (contact or lead), what (account, opportunity or custom), owner_id, due_at, done_at
`users`, `roles`	user: id, sfdc_id, email, role_id, profile, active; role: id, parent_id (the hierarchy)
`permissions`	profile or permission set, object, field, read, create, edit, delete
`shares`	record_id, object, grantee (user, role, team, territory), access level, reason
`pipeline_snapshots`	date, opportunity_id, stage, amount, close_date, owner_id, forecast_category
`workflow_rules`	id, object, trigger, conditions (jsonb), actions (jsonb), version, translated_from (the Apex or Flow name)
`audit_log`	object, record_id, field, old_value, new_value, user_id, via (UI, API, workflow), at

TWO RULES THAT SAVE A REBUILD

Custom fields become typed columns, not a JSON bag: reports filter, sort and sum on them, and the API must return them with their Salesforce names. **Snapshot the pipeline every day from day one**: week-over-week pipeline change and forecast accuracy need history you cannot rebuild later.

Fourteen weeks to build. *Eight more to switch.*

This is the order we build in. Each stage ends with something your RevOps lead can try. Then one team pilots it for four weeks, both systems run side by side for four more, and cutover lands at your renewal: about six months from kickoff to Salesforce-free.

WEEKS	BUILD	DONE WHEN
1–2	**Inventory and model.** Objects, fields, record types, flows, Apex, profiles, sharing rules and reports listed. The schema. Data Loader export into staging with IDs.	Counts in staging match Salesforce
3–4	**The workspace.** Account, contact, lead and opportunity views, activities, SSO and SCIM.	A rep can work a day in staging
5–6	**Access and workflow.** Roles, sharing, territories and field access; triggers, approvals, lead routing.	Access matches Salesforce for every test user
7–9	**The compatible API.** The endpoints your integrations call, auth, Apex translated. Outreach, Gong and Marketo connected in staging.	Every integration syncs without a config change
10–12	**Pipeline and reports.** Forecast engine, daily snapshots, the ~15 saved reports, Outlook and Gmail sync.	Reports match Salesforce's on the same data
13–14	**Rehearse.** A full migration dry run, load tests, training recorded.	The dry run finishes inside the cutover window
+4 / +4	**Pilot, then parallel run.** One team for four weeks; then both systems for four, a delta export, cutover at renewal.	Pipeline and forecast agree to the dollar

Same records, same access, *same numbers.*

Moving the data is mechanical. The work is proving three things on the same data: every user sees exactly the records they saw in Salesforce, every integration reads the same shape, and every report leadership trusts gives the same number.

1. **Export the effective access, not the settings.** Profiles, permission sets, the role hierarchy, org-wide defaults, sharing rules and territories combine in ways nobody can recite. For 20 sample users, export which records and fields each can see, and make that matrix your test.
2. **Enforce access in the database.** Row-level security in Postgres, plus field access in the API, so the UI, the compatible API and every integration obey one set of rules.
3. **Keep the shape integrations expect.** Same IDs, field API names, picklist values and record types. Replay a week of logged integration traffic against the new API before you switch anything.
4. **Rebuild the reports leadership reads, then prove them.** The ~15 saved reports and the forecast, with your stages, probability % and close-date logic. Run both systems on the same day's data and reconcile every total.
5. **Start pipeline history on day one.** Daily opportunity snapshots feed week-over-week change and forecast accuracy. Import Salesforce's field history too.

The parallel run is the proof

Four weeks of both systems turns parity from a claim into a report: pipeline, forecast and access checks, compared daily, signed off by your RevOps lead before renewal.

Move it, launch it, *run it.*

Migration

- ☐ Export twice: everything early, the changes since at cutover
- ☐ Keep Salesforce's ID on every row
- ☐ Create users, roles and picklists before records
- ☐ Load parents before children: accounts, contacts, opportunities, activities
- ☐ Bring field history and attachments with each record
- ☐ Translate only the Apex and flows that ran in the last 90 days
- ☐ Rebuild the 2–5 AppExchange apps you use as direct integrations
- ☐ Check counts, then walk 50 accounts end to end

Go-live

- ☐ Every integration re-pointed and syncing
- ☐ Reps signing in through your SSO
- ☐ The access matrix passing for every test user
- ☐ Outlook and Gmail sync live
- ☐ Backups proven by a restore
- ☐ Rollback ready until renewal

Running it

- ☐ **Always on:** uptime, API error and sync alerts, security patches
- ☐ **Weekly:** integration sync report to RevOps
- ☐ **Monthly:** upgrades, a report reconciliation, new fields reviewed
- ☐ **Quarterly:** an access review, AWS against the Salesforce bill you replaced
- ☐ **Yearly:** a restore drill and the SOC 2 evidence

Who owns it

Name one person before launch, usually the Salesforce admin you already have. The forecast your board sees runs on this system; someone has to answer when a number looks wrong.

Five ways a rebuild *stalls.*

1. Switching vendors and calling it a saving

HubSpot Sales Hub Enterprise is $150 a seat, but migration and ecosystem lock-in make it a multi-year project. Dynamics prices per seat much like Salesforce. Pipedrive, Close and Zoho converge above 50 seats. Each replaces one per-seat bill with another.

2. Paying an integrator to tune Salesforce

Implementation partners charge $500K–$3M for a typical enterprise rollout. That buys a better-tuned Salesforce, not less of it.

3. Forking open source

SuiteCRM is genuinely good and AGPL-licensed. Its UX is dated, its ecosystem small, and its upkeep real work. A focused CRM on Postgres is less code to own.

4. Re-pointing every integration

Rewiring Outreach, Gong and Marketo one by one turns a back-end swap into a vendor project. Build the compatible API, and they keep working.

5. Cutting over before the numbers match

A forecast that is off by one deal loses the CFO's trust. Run both systems until pipeline, forecast and access agree.

And one that is not a mistake: negotiating

Salesforce discounts Sales Cloud 15–30% on multi-year and multi-cloud bundles. Do it if the rebuild is a year away; it captures less than a quarter of what a rebuild does.

RATHER HAVE IT BUILT?

We build this exact plan. *Fixed price.*

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

$39,800 fixed up to 100 seats, $53,600–$81,100 for 100–300, $101,700–$151,100 above 300

Live in 10–14 weeks, then a four-week pilot on one sales team

Every object moves with full history and IDs; Apex becomes TypeScript or Python

90-day warranty, quarterly check-ins through year one, the code in your GitHub

We can run it for $6,000–$18,000 a year: hosting, monitoring, monthly upgrades

alliedbiz.tech/rebuild/salesforce-sales-cloud-alternative

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