Steigern Tech

FIELD GUIDE · 2026 EDITION

MIGRATION · UPGRADE · AUTOMATION

Odoo ERP Migration & Upgradation

A field guide to **database migration** — moving fragmented business operations out of spreadsheets, legacy software and disconnected tools into automated **Odoo Inventory, Accounting and Supply Chain** modules.



6 phases
a proven path from discovery to hypercare

3×
full rehearsal migrations before go-live

0 diff
trial balance and stock value reconciled to the unit

1
single source of truth for stock, money and orders

Published by Steigern Tech LLP · steigerntech.com

14 pages · Field guide edition

INSIDE THIS GUIDE

From scattered tools to one automated ERP

For founders, finance heads, operations managers and IT leads planning a move into Odoo — or stuck on an old Odoo version — and who cannot afford to lose a single invoice, stock unit or customer record on the way.

03 **Are you ready?**
Warning signs, readiness score, edition & hosting choice

04 **The 6-phase methodology**
Timeline, owners and deliverables per phase

05 **Fit-gap & the customisation ladder**
Configure before you customise

06 **Data migration architecture**
The ETL pipeline and the load order that never breaks

07 **Cleansing, mapping & loading**
Field maps, external IDs and an idempotent loader

08 **Inventory module**
Warehouses, routes, valuation and opening stock

09 **Accounting module**
Chart of accounts, opening balances, tie-out proof

10 **Supply chain automation**
Procure-to-pay and order-to-cash, end to end

11 **Version upgrade playbook**
Upgrading Odoo itself and porting custom modules

12 **Testing & cutover weekend**
Reconciliation tests and an hour-by-hour runbook

13 **Go-live checklist & pitfalls**
40 checks and the 6 mistakes that sink projects

14 **Work with Steigern Tech**
Migration, upgrade and custom module services

TWO DIFFERENT JOURNEYS

Migration

OTHER SYSTEM → ODOO

- Source: spreadsheets, Tally, QuickBooks, SAP B1, a home-grown app — often several at once.
- Hard part: **data quality and mapping**. Every system models customers, items and taxes differently.
- Also a **process redesign**: you adopt Odoo's flows instead of copying old workarounds.
- Typical duration: **6–16 weeks** depending on modules and data volume.

Upgradation

ODOO VN → LATEST

- Source: an older Odoo database (Community or Enterprise).
- Hard part: **custom modules and Studio changes** that must be ported to the new version's code and views.
- Data is transformed by Odoo's upgrade service or OpenUpgrade scripts, then tested.
- Typical duration: **2–8 weeks**, driven by how much custom code you carry.

Rule of thumb: budget roughly **40% of effort for data**, 30% for configuration and custom work, 20% for testing and 10% for training and cutover. Projects fail when the first number is quietly set to zero.

01 · READINESS ASSESSMENT

Is your business ready to move?

Fragmentation has a cost you pay every day — in double entry, stock surprises and month-end panic. Count the symptoms, then score your readiness honestly.

2–4×
Same data typed repeatedly — order in one sheet, invoice in accounting, dispatch in a register.

?
Nobody trusts stock numbers. Physical counts and system counts disagree every month.

10+ days
Month-end close spent reconciling exports instead of reviewing results.

READINESS SCORECARD — SCORE EACH 1 TO 5

DIMENSION	HEALTHY LOOKS LIKE	SCORE
Executive sponsor	Named owner with authority to decide	●●●●●
Documented processes	As-is flows written down	●●●○○
Master data quality	No duplicate customers / items	●●○○○
Key-user availability	~30% time for 2–3 months	●●●●○
Clean accounting books	Last year closed and audited	●●●●○
Change appetite	Willing to adopt standard flows	●●●○○

Example scores shown. Total ≥ 22/30: start now · 15–21: fix the weakest two first · under 15: run a 2-week discovery before committing.

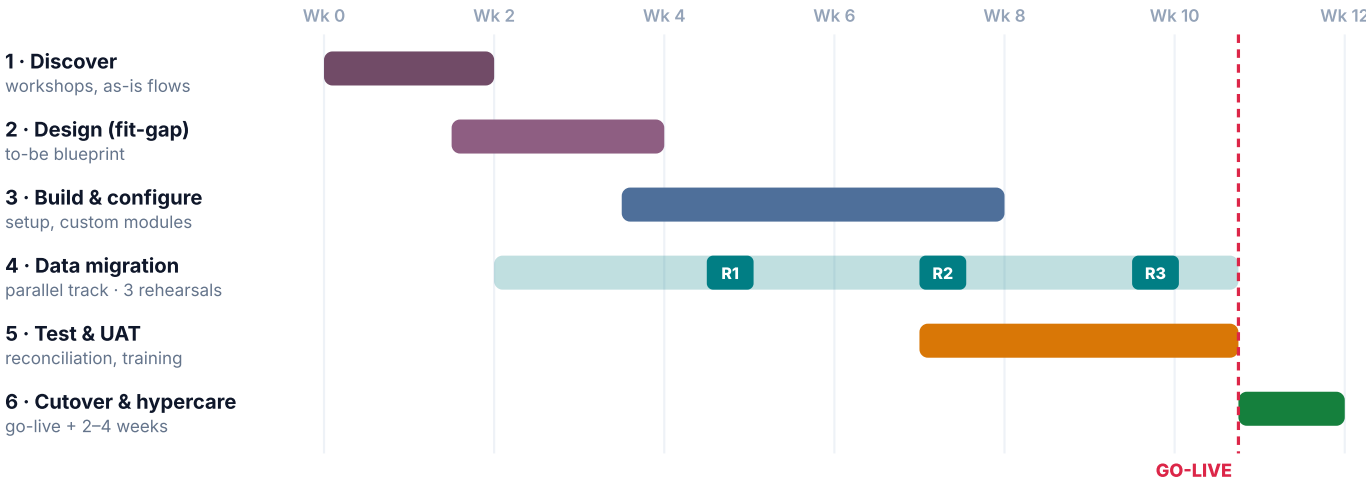
CHOOSE EDITION & HOSTING

OPTION	BEST FOR	TRADE-OFF
Odoo Online (SaaS)	Standard processes, fastest start, no server to manage	No custom Python modules; upgrades on Odoo's schedule
Odoo.sh	Enterprise with custom modules, Git-based staging branches	Subscription cost; works within Odoo.sh's platform limits
Self-hosted Enterprise	Data residency, deep integrations, full control	You own servers, backups, security and upgrades
Self-hosted Community	Tight budgets with in-house or partner developers	No Enterprise apps (e.g. full Accounting, Studio); upgrades via OpenUpgrade

02 · METHODOLOGY

Six phases, no surprises

Each phase ends with a sign-off. Data migration runs as a parallel track from week 2, so by cutover you have loaded the data three times already.



Illustrative 12-week plan for 3–5 core modules and a single company; multi-company or heavy manufacturing typically extends phases 3–5.

PHASE 1–2 · DELIVERABLES

Blueprint

- As-is process maps
- Fit-gap register with decisions
- Data inventory & owners
- Signed scope and cutover date

PHASE 3–4 · DELIVERABLES

Working system

- Configured staging database
- Custom modules with tests
- Mapping workbook & load scripts
- Rehearsal reconciliation reports

PHASE 5–6 · DELIVERABLES

Live business

- UAT sign-off per department
- Trained users & quick guides
- Cutover runbook, executed
- Hypercare log, then handover

WHO DOES WHAT

ROLE	RESPONSIBILITY
Sponsor	Owns scope and budget, breaks ties, signs off each phase
Key users (per dept)	Explain current process, validate data, run UAT, train colleagues
Functional consultant	Fit-gap, Odoo configuration, process design, training material
Developer / data engineer	Custom modules, integrations, ETL scripts, performance

PHASE GATES — DON'T MOVE ON UNTIL YOU CAN ANSWER YES

Gate 1 · after Design

Every requirement has a fit-gap decision, and the sponsor signed the scope and the cutover date.

Gate 2 · after Build

Key users completed each core flow in staging without consultant help.

Gate 3 · after Rehearsal 3

Zero reconciliation differences, and the full load fits inside the cutover window.

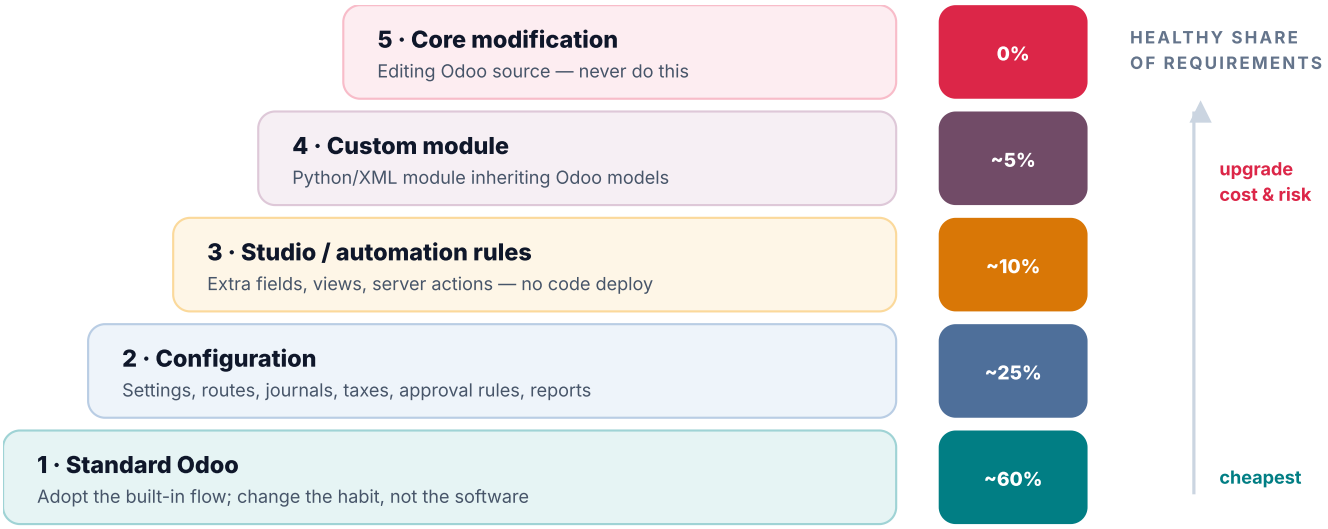
Gate 4 · go / no-go

UAT signed by every department; no open severity-1 defects; rollback plan agreed.

03 · FIT-GAP ANALYSIS

Configure before you customise

Every requirement climbs the ladder only as far as it must. Each step up adds cost — not just to build it, but to test it and carry it through every future Odoo upgrade.



A FIT-GAP REGISTER ENTRY

REQUIREMENT	ODOO STANDARD?	DECISION	LEVEL
Approve POs above a value	Yes — Purchase approval setting	Configure threshold per company	2
Batch & expiry tracking	Yes — Lots/Serials + expiration	Enable, train warehouse team	2
Extra "transporter" field on delivery	No	Studio field + report change	3
Commission rule unique to your trade	Partially	Small custom module with tests	4
Old invoice layout "exactly as before"	Close enough	Adopt standard layout + logo	1

Questions that shrink custom work

- Is this a legal requirement or a habit?
- What does it cost us if we do it Odoo's way?
- Can an app from the Odoo Apps store cover it?
- Will we still need it in two years?

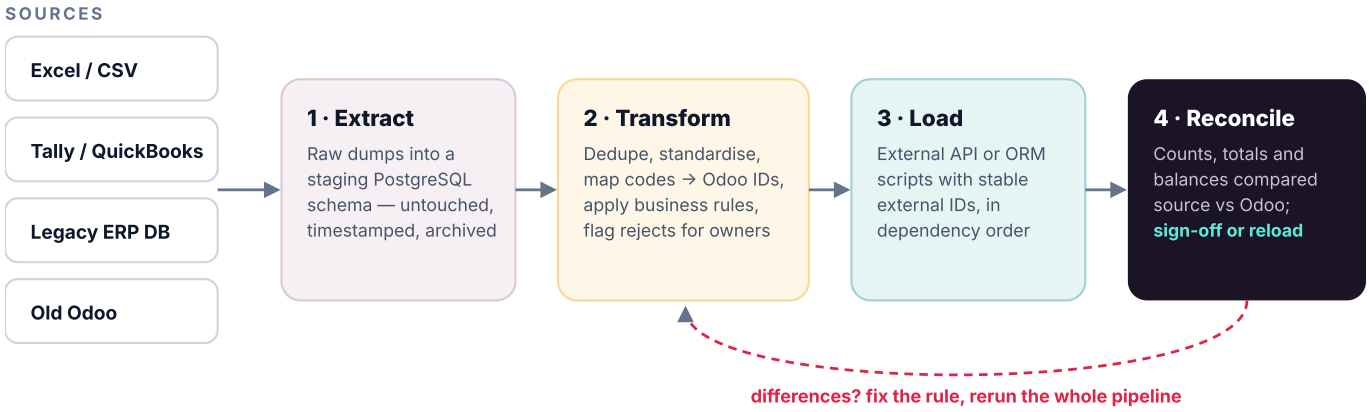
When custom code is the right call

- A genuine competitive process no ERP ships
- Integrations: e-commerce, logistics, banks, government portals
- Industry rules — pricing, compliance, traceability
- Always: own Git repo, tests, and an upgrade owner

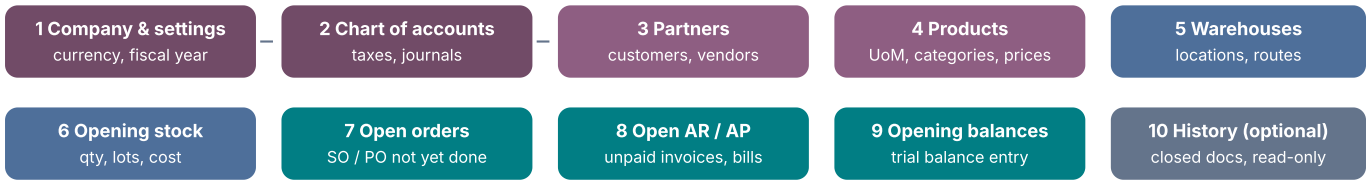
04 · DATA MIGRATION ARCHITECTURE

The pipeline that moves your history

Never import straight from the old system into production. Stage, cleanse and map in between, so every rehearsal is repeatable and every number is traceable back to its source.



LOAD ORDER — PARENTS BEFORE CHILDREN, ALWAYS



A product that references a missing category, or an invoice that references a missing customer, fails — or worse, silently creates a duplicate. The order above removes both problems.

Migrate

Master data, open transactions, opening balances and stock — everything needed to **operate on day one**.

Summarise

Closed history: carry monthly totals for reports, not every old line, unless analytics truly need it.

Archive

Keep the old system read-only (or a full export) for the legally required retention period.

WHAT TO MIGRATE, ENTITY BY ENTITY

ENTITY	BRING OVER	NOTES
Customers & vendors	Active only	Active = transacted in the last 2–3 years or has an open balance
Products	Active + in stock	Include variants, UoM conversions, barcodes, vendor prices
Stock	Cutover quantities	From a physical count, per location and lot
Sales & purchase orders	Open only	Remaining quantities, original numbers kept as reference
Invoices & bills	Unpaid only	Partially paid ones carry the open amount
General ledger	Opening balances	Monthly totals of prior years only if comparative reports need them

05 · CLEANSING, MAPPING & LOADING

Clean data in, trusted reports out

UNIQUE
One record per customer, vendor and item — merge "ACME", "Acme Ltd." and "ACME LTD"

COMPLETE
Tax IDs, addresses, UoM, cost and HSN/tax codes filled before load

VALID
Formats, dates and codes checked against Odoo's allowed values

CURRENT
Inactive items and dormant partners archived, not migrated

SAMPLE FIELD MAPPING WORKBOOK

LEGACY SOURCE	ODOO MODEL · FIELD	RULE
Ledger: Sundry Debtors	`res.partner` · name, vat, customer_rank	Dedupe on tax ID, then normalised name
Stock Item master	`product.template` · default_code, uom_id, categ_id	Item code becomes Internal Reference
Godown / store	`stock.location` under a `stock.warehouse`	One warehouse per physical site
Ledger groups	`account.account` · code, account_type	Map to localisation chart, never 1:1 blindly
Tax ledgers	`account.tax` (localisation)	Use localisation taxes; map rates explicitly
Pending sales bills	`account.move` (out_invoice)	Only unpaid; original number kept in reference

LOAD_PARTNERS.PY — IDEMPOTENT LOADER VIA ODOO'S EXTERNAL API

```python
import xmlrpc.client   # shown for broad version support; newer Odoo versions also offer a JSON-2 API

url, db = "https://erp.example.com", "staging_rehearsal_2"
uid = xmlrpc.client.ServerProxy(f"{url}/xmlrpc/2/common").authenticate(db, USER, API_KEY, {})
odoo = xmlrpc.client.ServerProxy(f"{url}/xmlrpc/2/object")
call = lambda model, method, *args, **kw: odoo.execute_kw(db, uid, API_KEY, model, method, list(args), kw)

def upsert(model, xmlid, vals):
    """Re-running a rehearsal updates records instead of duplicating them."""
    ref = call("ir.model.data", "search_read",
               [("module", "=", "legacy"), ("name", "=", xmlid)], fields=["res_id"], limit=1)
    if ref:
        call(model, "write", [ref[0]["res_id"]], vals)
        return ref[0]["res_id"]
    rid = call(model, "create", vals)
    call("ir.model.data", "create", {"module": "legacy", "name": xmlid, "model": model, "res_id": rid})
    return rid

for row in read_staging("clean_customers"):          # already deduped & validated
    upsert("res.partner", f"cust_{row['legacy_code']}", {
        "name": row["name"], "vat": row["tax_id"], "email": row["email"],
        "customer_rank": 1, "ref": row["legacy_code"],          # traceability back to source
    })
```

Which load method?

Odoo's import tool (CSV/XLSX, with an External ID column) for small, simple master data · **External API scripts** for volume, transformations and repeatable rehearsals · **Server-side ORM scripts** for very large loads on self-hosted databases.

Non-negotiables

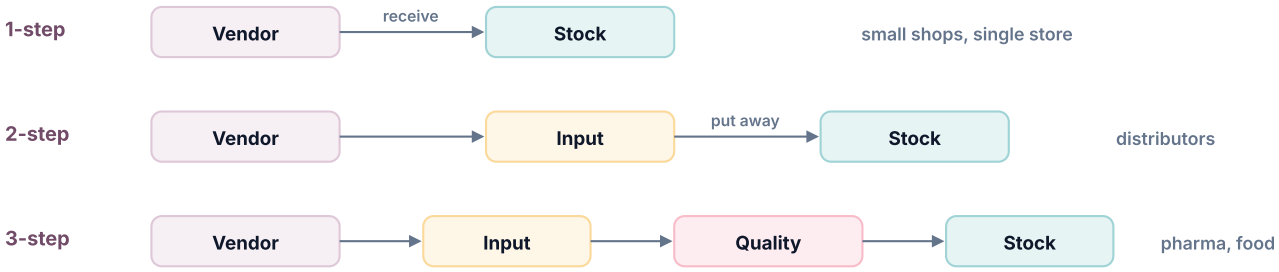
- Every record gets a stable external ID
- Loads are re-runnable end-to-end
- Rejects go back to a named data owner

06 · INVENTORY MODULE

Stock you can finally trust

Odoo Inventory uses double-entry stock moves: every unit moves *from* one location *to* another, so nothing appears or disappears without a trace. Design the structure before loading a single quantity.

Warehouse flows — pick the complexity you actually need



Deliveries mirror this: ship directly, pick → ship, or pick → pack → ship.

Valuation decisions (per product category)

Standard price	Fixed cost, variances tracked — manufacturing
Average cost (AVCO)	Most trading businesses
FIFO	Perishables, price-volatile goods
Automated valuation	Stock moves post to accounting in real time — the integration you migrated for

*Agree the method with your auditor **before** loading opening stock.*

Loading opening stock right

- Physical count at cutover, signed by the warehouse lead.
- Load quantities per **location**, with **lot/serial** and expiry where tracked.
- Load **unit cost** per product so stock value = legacy closing stock value.
- Inventory value report must tie to the stock account in the trial balance.

AUTOMATION YOU SWITCH ON AFTER GO-LIVE

Reordering rules
Min/max per product creates purchase or manufacturing orders automatically.

Barcode
Scan receipts, picks and counts on a phone or scanner.

Cycle counts
Scheduled counts by location replace the annual stock-take shutdown.

Traceability
Upstream/downstream report for any lot — recalls in minutes.

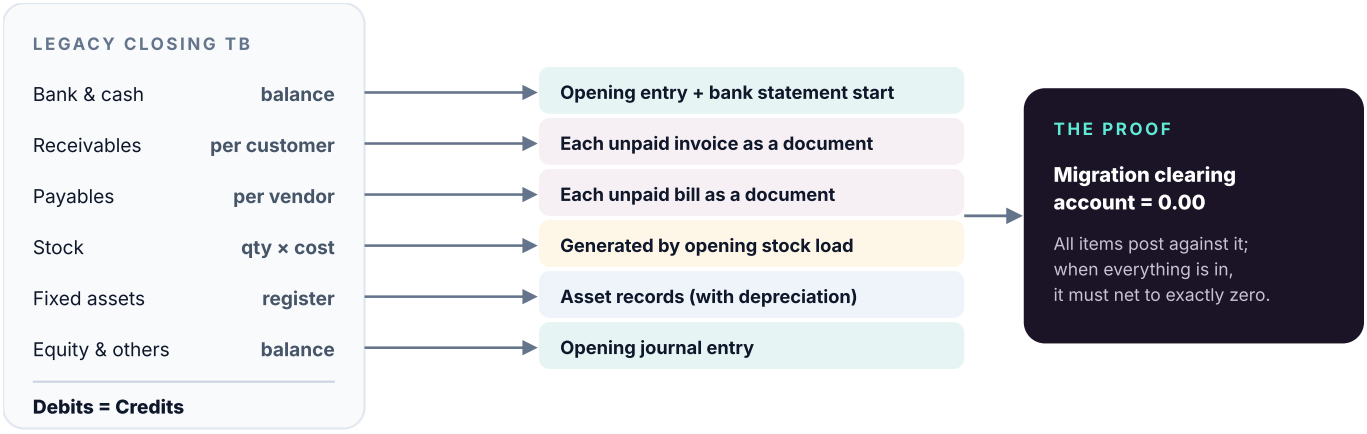
INVENTORY SIGN-OFF

- [ ] Every physical location exists in Odoo
- [ ] Count sheets match loaded quantities
- [ ] Lot / serial numbers unique and complete
- [ ] Units of measure and conversions tested
- [ ] Stock value equals legacy closing value
- [ ] Test receipt, transfer and delivery run end to end

07 · ACCOUNTING MODULE

Opening balances that tie out to zero

Pick a clean cutover date — ideally the first day of a fiscal year or month — and prove that the books on day one in Odoo equal the closing books of the old system, line by line.



Why open items, not totals?

Loading **each unpaid invoice and bill** (rather than one receivable total) lets customers' payments after go-live reconcile against the right document, keeps ageing reports accurate, and lets follow-up reminders start on day one.

Chart of accounts & taxes

- Start from Odoo's **fiscal localisation** package (e.g. India GST) — it brings compliant taxes, reports and tax grids.
- Map old ledgers onto it; add accounts only where reporting needs them.
- Set fiscal positions for exports, SEZ and inter-state rules.

FINANCE SIGN-OFF CHECKLIST

- [ ] Trial balance matches legacy to the paisa / cent
- [ ] AR and AP ageing match per partner
- [ ] Bank balances match statements on cutover date
- [ ] Stock valuation = stock account balance
- [ ] Clearing account is zero, then archived
- [ ] Auditor / CA has reviewed the opening entry

CHOOSING THE CUTOVER DATE

OPTION	PROS	CONS
Start of fiscal year	Cleanest books; only balance-sheet balances to load	Busy season for finance; waits for audit
Start of a month / quarter	Most common; monthly reporting stays intact	Year-to-date P&L must be loaded as monthly totals
Mid-month	Fits urgent deadlines	Split-month reporting and tax returns — avoid if possible

08 · SUPPLY CHAIN AUTOMATION

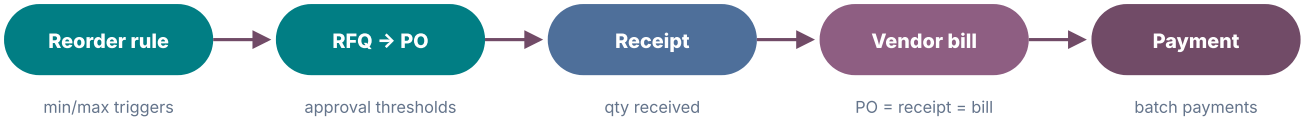
One flow from order to cash

This is the payoff of consolidation: Sales, Purchase, Inventory and Accounting share one database, so each step creates the next document automatically instead of someone re-typing it.

Order-to-cash



Procure-to-pay, with 3-way match



BEFORE VS. AFTER CONSOLIDATION

TASK	FRAGMENTED (BEFORE)	ODOO (AFTER)
Know what to reorder	Weekly stock sheet, gut feel	Reordering rules propose POs daily
Invoice a shipment	Copy dispatch note into accounting	Invoice created from delivered quantities
Check a vendor bill	Hunt for PO and goods-receipt register	Bill matched to PO and receipt automatically
Promise a delivery date	Call the warehouse	Forecasted stock on the sales order line
Month-end stock value	Spreadsheet valuation, journal by hand	Live, posted with every stock move

Make-to-order & dropship
Sales orders trigger purchases or vendor-direct shipping without manual steps.

Manufacturing (MRP)
Bills of materials consume components and value finished goods automatically.

Automation rules
Notify, assign or update records on any condition — no code for most cases.

WHAT CHANGES FOR EACH TEAM

↓ **Finance**
Fewer manual journals; faster month-end close

↑ **Warehouse**
Real-time stock accuracy by location and lot

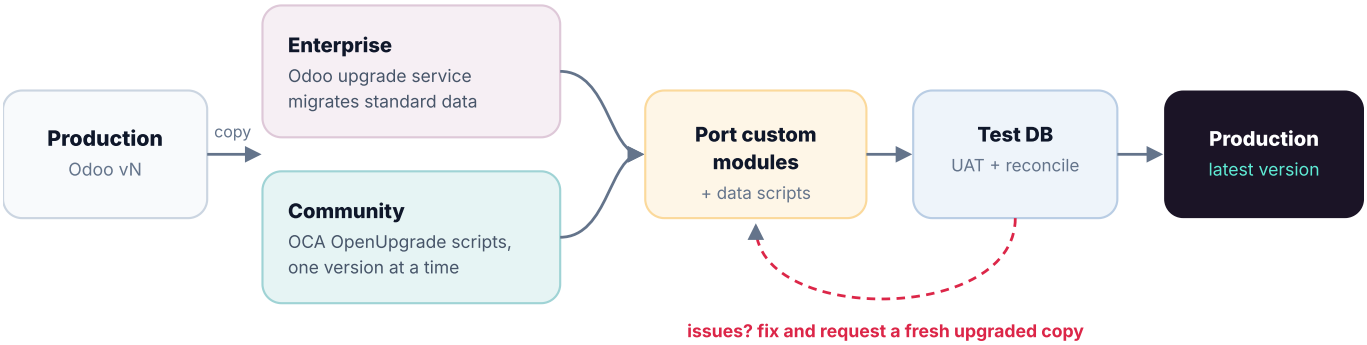
↓ **Purchasing**
Stock-outs and emergency buys

↑ **Sales**
Reliable delivery promises and faster invoicing

09 · VERSION UPGRADE PLAYBOOK

Upgrading Odoo without breaking it

Odoo ships a new major version every year and maintains only the most recent few. Staying more than two versions behind means lost security fixes, missing features and a harder jump later.



Porting custom modules — what usually breaks

- Renamed or removed fields and models between versions
- View inheritance (XPaths) pointing at elements that moved
- Front-end widgets — rewrite legacy JS for the OWL framework
- Report templates (QWeb) and Studio customisations
- Third-party apps without a release for the target version
- Python and PostgreSQL minimum versions on self-hosted servers

Make every future upgrade cheaper

- Inherit — never edit — Odoo's core code
- Keep custom modules small, focused and in Git
- Write automated tests for every custom flow
- Uninstall unused apps and modules before upgrading
- Upgrade every 1–2 versions instead of every 5
- Record all Studio changes in your fit-gap register

UPGRADE EFFORT ESTIMATOR

YOUR SITUATION	TYPICAL EFFORT	MAIN DRIVER
Standard apps, no custom code	1–2 weeks	Testing and retraining on UI changes
A few custom modules, 1 version jump	2–4 weeks	Porting and regression testing
Heavy customisation, 3+ versions behind	6–12 weeks	Multi-step data upgrade, re-design of old customisations

Pre-upgrade audit — do this before requesting the upgraded copy

- List every installed module and who uses it
- Export Studio customisations and automation rules
- Check third-party apps for target-version releases
- Clean failed scheduled actions and orphaned records
- Confirm server meets target Python / PostgreSQL versions
- Freeze new customisation until the upgrade is live

10 · TESTING & CUTOVER

The go-live weekend, hour by hour

DATA

Reconciliation

Record counts, totals, TB, ageing and stock value vs source

PROCESS

End-to-end UAT

Key users run real scenarios: order → delivery → invoice → payment

TECHNICAL

Custom & integrations

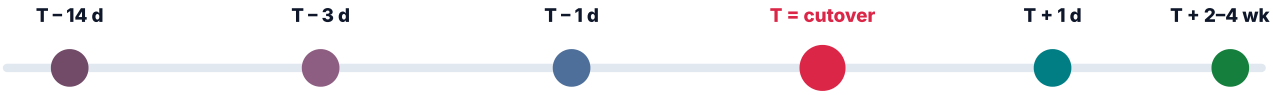
Automated tests, e-commerce, banks, e-invoicing portals

LOAD

Performance

Month-end reports and peak order volumes on production-sized data

Cutover runbook



Freeze master-data & config changes; rehearsal 3 signed

Final prep users trained; go/no-go criteria agreed

Close legacy last transactions; stock count; final extract

Load & verify run pipeline; finance & stock sign-off; go/no-go

Go live legacy read-only; war room open

Hypercare daily triage, first close in Odoo

Rollback plan: if go/no-go fails, the legacy system reopens unchanged — which is only possible because it was set read-only, not shut down.

Go / no-go criteria

- Trial balance and stock value differences = 0
- All critical UAT scenarios passed
- No open severity-1 defects
- Integrations tested end-to-end in production
- Each department lead has signed

Hypercare essentials

- One ticket channel; triage twice daily for week 1
- Floor-walkers in warehouse and accounts
- Daily stock and bank reconciliation for 2 weeks
- First month-end close run together with the partner
- Handover to support with a known-issues log

RECONCILIATION TEST MATRIX

CHECK	LEGACY REPORT	ODOO REPORT	TOLERANCE
Record counts	Master lists per entity	List views with filters	Exact
Trial balance	Closing TB at cutover	Trial Balance report	0.00
Receivables / payables	Party-wise outstanding	Aged Receivable / Payable	0.00 per partner
Stock quantity & value	Stock summary	Inventory valuation	Exact qty, 0.00 value
Open orders	Pending SO / PO register	Sales / Purchase to deliver	Exact

11 · GO-LIVE READINESS

The 40-point migration checklist

Data — MUST

- [ ] Data owners named for every entity
- [ ] Duplicates merged; dormant records archived
- [ ] Mapping workbook signed by key users
- [ ] Every record has a stable external ID
- [ ] Three full rehearsals completed and reconciled
- [ ] Rejected rows resolved, not ignored
- [ ] Attachments / documents migrated or archived
- [ ] Legacy system set read-only, retained per law
- [ ] Backups of source and Odoo before cutover
- [ ] Load timings fit inside the cutover window

Finance — MUST

- [ ] Localisation, taxes and fiscal positions verified
- [ ] Chart of accounts mapping approved by CA
- [ ] Opening TB equals legacy closing TB
- [ ] AR / AP ageing match per partner
- [ ] Bank journals start at statement balance
- [ ] Stock valuation ties to stock accounts
- [ ] Migration clearing account = 0
- [ ] Invoice / e-invoice numbering sequences set
- [ ] Payment terms and follow-up levels loaded
- [ ] User access rights follow segregation of duties

Operations — SHOULD

- [ ] Warehouses, locations and routes mirror the floor
- [ ] Physical count done and signed at cutover
- [ ] Lots / serials and expiry dates loaded
- [ ] Reordering rules reviewed with buyers
- [ ] Open SOs and POs loaded with correct status
- [ ] Vendor price lists and lead times in place
- [ ] Delivery, invoice and PO templates approved
- [ ] Barcode labels printed for locations / products
- [ ] Each role has a one-page quick guide
- [ ] Key users can demo their flow unaided

Technical — SHOULD

- [ ] Custom modules in Git, with automated tests
- [ ] No modifications to Odoo core code
- [ ] Staging environment mirrors production
- [ ] Integrations tested with real credentials
- [ ] Scheduled actions and e-mail servers checked
- [ ] Automated daily backups with a tested restore
- [ ] HTTPS, 2FA and strong admin passwords
- [ ] Server sized for peak users and month-end
- [ ] Monitoring and error alerts enabled
- [ ] Upgrade owner and next-version plan named

The 6 mistakes that sink Odoo projects

1. Recreating old workarounds in custom code
2. Loading dirty data "to clean later"
3. A single migration attempt, on go-live day
4. Key users too busy to test
5. Migrating all history instead of open items
6. No owner for upgrades after go-live

Steigern Tech

STEIGERNTECH.COM

MIGRATE WITH US

Moving to Odoo — or stuck on an old version?
Let's migrate it right, once.

Steigern Tech plans and delivers Odoo migrations, upgrades and custom modules — with rehearsed data loads and books that reconcile on day one.

Migration assessment

Process and data audit, fit-gap register, and a fixed plan with timeline before you commit.

Data migration & cutover

ETL pipelines, three rehearsals, reconciliation reports and a managed go-live weekend.

Version upgrades

Community and Enterprise upgrades, custom-module porting and regression testing.

Custom modules & integrations

Industry-specific modules, e-commerce, logistics, payment and government-portal integrations.

WHATSAPP

+91 94275 72571

EMAIL

info@steigerntech.com

WEBSITE

steigerntech.com

© 2026 Steigern Tech LLP. Timelines and percentages are typical ranges — your plan depends on scope and data.

Odoo is a trademark of Odoo S.A. Steigern Tech is an independent service provider.