Automated Receiving Process Business Plan

Prepared for: The Hashery Management & IT Team
Prepared by: Diego Arriola, Data Analyst
Date: February 25, 2026
Project: GCP-Based Receiving Automation Pipeline

Executive Summary

This proposal outlines an automated receiving workflow to replace manual data entry for product receiving across The Hashery's current and future store locations. By implementing OCR, API integration with METRC NJ, and automated data pipelines on Google Cloud Platform (GCP), we can reduce receiving time by 80-90%, eliminate manual transcription errors, and scale efficiently to three stores by early 2027.

The solution leverages existing technology investments (Dutchie ERP, METRC API access, Slack) and builds a cloud-native pipeline that processes vendor invoices and state manifests automatically. Initial implementation targets Store 1 with rapid expansion to new locations as they open.

Key Benefits:

- Reduce receiving processing time from 1-2 hours to 10-15 minutes per delivery
- Eliminate manual transcription of expiration dates, costs, and prices
- Create audit trail for compliance and inventory accuracy
- Scale seamlessly to multiple store locations
- Enable supervisor approval workflow without physical document handling

Current Process Overview

Today's receiving workflow involves 12 manual steps spanning multiple systems and stakeholders:

1. Buyer places verbal or written purchase order with vendor
2. Vendor processes and ships order
3. Store receives physical delivery
4. Store staff confirms quantities received and checks expiration dates on physical product batches, then manually writes these dates on the METRC manifest
5. Store staff manually updates cost/price on vendor invoice (handwritten)
6. Staff scans invoice and manifest to PDF (no centralized storage)
7. Slack message with manifest photo confirms delivery
8. Data analyst manually inputs each line item into Dutchie Receive Inventory UI
9. New products require manual catalog creation
10. Supervisor validates random items against physical inventory
11. Invoice posted to Slack channel with vendor name and invoice number
12. Product supervisor reviews and approves (replies "Approved")
13. Second receive operation in Dutchie triggers label printing

Pain Points:

- Manual data entry takes 1-2 hours per delivery depending on line item count
- Handwritten expiration dates from physical product inspection create transcription errors
- Manual quantity confirmation is time-consuming and error-prone
- No centralized storage for scanned documents (compliance risk)
- Double data entry (once for validation, again for final receive)
- Process does not scale beyond one location without adding full-time staff
- Approval workflow relies on manual Slack monitoring

Proposed Automated Solution

Architecture Overview

The automated system uses Google Cloud Platform services to create an event-driven pipeline that processes documents, enriches data via METRC API, and outputs Dutchie-ready files.

Component	Technology	Purpose
Storage	Google Cloud Storage	Centralized document repository
OCR	Cloud Vision API	Extract text from PDF invoices/manifests
Enrichment	Cloud Run + METRC API	Retrieve expiration dates and package data
Automation	Cloud Functions Gen2	Event-driven processing pipeline
Notifications	Slack Bot (Cloud Run)	Approval workflow automation
Audit	BigQuery	Compliance logging and analytics
Integration	Dutchie Bulk Upload	Automated inventory receiving

Table 1: System components and technologies

Automated Workflow

1. **Document Upload:** Store staff uploads scanned invoice + METRC manifest PDFs to designated GCS folder: `store-1/2026/02/25/vendor-name/`
2. **Automatic Processing:** GCS upload triggers Cloud Function that:
    - Runs Cloud Vision OCR on both PDFs
    - Extracts product names, quantities, costs, prices, package IDs, and handwritten expiration dates from annotated manifest
    - Calls METRC NJ API to retrieve package details and validate quantities
    - Joins data and generates Dutchie-compatible CSV

    - Writes CSV to GCS processed folder
    - Logs transaction to BigQuery for audit
3. **Slack Notification:** Slack bot posts message to #hashery-nj-receiving: "VendorX Invoice 123 for Store 1 processed, CSV ready [link]"
4. **Supervisor Review:** Product supervisor reviews CSV (optional download) and replies "Approved" or requests corrections
5. **Approval Detection:** Slack bot detects approval, marks CSV as ready in Firestore/BigQuery
6. **Dutchie Integration:** Staff uploads approved CSV to Dutchie bulk receive (future: automated API push)
7. **Label Printing:** Dutchie triggers label printing at store; products move to floor

Technical Implementation Details

Google Cloud Storage Folder Structure:

```
hashery-receiving/
├── store-1/
│   └── 2026/
│   └── 02/
│   └── 25/
│   └── vendor-curaleaf/
│   ├── invoice-12345.pdf
│   ├── manifest-1A4000.pdf
│   └── processed/
│   └── receiving-12345.csv
├── store-2/
│   └── [same structure]
└── store-3/
└── [same structure]
```

Cloud Vision OCR Function:

- Trigger: GCS object finalize event for *.pdf files
- Runtime: Python 3.11 with Cloud Vision client library
- Output: Structured JSON with line items (product, qty, cost, price, package_id)
- Error handling: Logs failures to BigQuery for manual review

METRC Enrichment Service (Cloud Run):

- Accepts package IDs via REST API
- Calls METRC NJ API (https://api-nj.metrc.com) with stored credentials (Secret Manager)
- Returns: production dates, source facility, expected quantities for validation
- Validates OCR-extracted expiration dates against expected product shelf life
- Caches responses in Firestore to minimize API calls

Slack Bot (Cloud Run):

- Built with Slack Bolt SDK (Python)
- Handles Events API for message replies
- Posts formatted messages with CSV links (signed GCS URLs, 24-hour expiry)
- Updates approval status in Firestore when supervisor responds

Dutchie CSV Format:
Matches Dutchie POS bulk upload template with required columns:

- catalog_product: Product name/SKU matched to existing catalog
- package_id: METRC package identifier (e.g., 1A4000...)
- quantity: Number of units
- expiration_date: YYYY-MM-DD format from METRC
- cost_per_unit: Wholesale cost from invoice
- price_per_unit: Retail price from invoice or catalog
- room: Always "Receiving Room"

Business Impact Analysis

Time Savings

Task	Current (min)	Automated (min)	Savings
Manual data entry	60-120	5	92%
Expiration transcription	15-30	0	100%
Document scanning/storage	10	2	80%
Supervisor approval	15	5	67%
Second Dutchie receive	20	3	85%
Total per delivery	**120-195**	**15**	**87-92%**

Table 2: Time savings per delivery

Monthly Impact (Store 1):

- Average deliveries per month: 40-50
- Current labor: 80-160 hours/month
- Automated labor: 10-12.5 hours/month
- Labor savings: 70-147.5 hours/month

Three-Store Projection (Q1 2027):

- Total deliveries: 120-150/month
- Current labor requirement: 240-480 hours/month (3-6 FTE)
- Automated labor: 30-37.5 hours/month (0.5 FTE)
- Labor savings: 210-442.5 hours/month

Error Reduction

Current Error Sources:

- Handwritten expiration dates (written by store staff from physical products): 5-10% transcription error rate
- Manual quantity verification: 3-5% error rate
- Manual price/cost entry: 3-5% error rate
- Wrong product selection in Dutchie: 2-3% error rate
- Missing METRC package ID linkage: 8-12% of items

Automated Accuracy:

- OCR accuracy on handwritten dates: 85-92% (Cloud Vision on handwriting; manual review for low-confidence extractions)
- OCR accuracy on printed invoices: 95-98%
- METRC API data: 100% accurate (direct from state system for validation)
- Catalog matching: 90-95% with fuzzy matching (manual review for new products)
- Package ID linkage: 100% (automated join)

Quality Impact:

- Reduces inventory discrepancies
- Improves METRC compliance reporting accuracy
- Enables better margin analysis (accurate cost data)

Scalability

The automated system scales linearly with store count without proportional staffing increases:

Stores	Deliveries/Month	Manual FTE	Automated FTE
1	40-50	1.0-2.0	0.15
2	80-100	2.0-4.0	0.30
3	120-150	3.0-6.0	0.45

Table 3: Staffing requirements by store count

Cloud Cost Estimates:

- GCS storage: $0.02/GB/month (~$5/month for 250GB documents)
- Cloud Vision OCR: $1.50 per 1,000 pages (~$75-100/month for 50-65k pages)
- Cloud Run/Functions: $0.40/million requests (~$20/month)
- BigQuery: $5/TB scanned (~$10/month for audit queries)
- METRC API: No usage fees (state-provided)
- **Total GCP monthly cost: $110-135 (all stores)**

Compliance Benefits

- **Audit Trail:** BigQuery logs every document processed with timestamps, user IDs, and data changes
- **METRC Accuracy:** Direct API integration eliminates manual transcription errors in state reporting
- **Document Retention:** Centralized GCS storage with 7-year retention policy for compliance
- **Version Control:** All CSV outputs stored with processing metadata for reconstruction

Implementation Plan

Phase 1: Pilot (Weeks 1-4)

Objectives:

- Validate OCR accuracy on sample invoices/manifests from top 3 vendors
- Build METRC enrichment service and test with real package IDs
- Create GCS folder structure and access controls
- Develop basic Slack notification (no approval workflow yet)

Deliverables:

- Working OCR pipeline for 3 vendors
- METRC API integration
- Sample Dutchie CSV outputs
- Documentation for store staff

Resources:

- 1 data analyst (Diego) – 20 hours/week
- IT support for GCP project setup and IAM – 5 hours total
- 1 store manager for testing – 2 hours/week

Phase 2: Approval Workflow (Weeks 5-6)

Objectives:

- Build Slack bot with approval detection
- Implement Firestore approval status tracking

- Add BigQuery audit logging
- Train product supervisor on new workflow

Deliverables:

- Slack bot deployed on Cloud Run
- Approval workflow documentation
- BigQuery dashboard for processing metrics

Resources:

- 1 data analyst – 15 hours/week
- Product supervisor training – 3 hours

Phase 3: Store 1 Production (Weeks 7-8)

Objectives:

- Deploy to Store 1 for all vendors
- Run parallel with manual process for validation
- Collect accuracy metrics
- Refine catalog matching rules

Deliverables:

- Production deployment checklist
- Store staff training materials
- Error handling procedures
- Week 1 accuracy report

Resources:

- 1 data analyst – 10 hours/week
- Store 1 staff – 5 hours training

Phase 4: Dutchie Integration Enhancement (Weeks 9-12)

Objectives:

- Request and obtain Dutchie API access
- Build direct API integration (bypass manual CSV upload)
- Implement automatic label printing trigger
- Expand OCR to support 10+ vendors

Deliverables:

- Dutchie API integration service (Cloud Run)
- Automated end-to-end workflow (upload → labels)
- Vendor coverage expansion to 95%+

Resources:

- 1 data analyst – 15 hours/week
- Dutchie support coordination – 5 hours

Phase 5: Multi-Store Rollout (Ongoing)

Objectives:

- Deploy to Store 2 and Store 3 as they open
- Configure store-specific folder structures
- Train new store staff
- Monitor and optimize costs

Deliverables:

- Store 2/3 deployment runbooks
- Per-store dashboards
- Cost optimization report

Resources:

- 1 data analyst – 5 hours per new store
- Store managers – 2 hours training each

Risk Assessment and Mitigation

Risk	Likelihood	Impact	Mitigation
OCR accuracy on handwritten notes	Medium	High	Phase out handwritten annotations; require typed invoices from vendors
METRC API downtime	Low	Medium	Cache recent package data; fallback to manual entry with flag
Dutchie bulk upload format changes	Low	Medium	Version-control CSV templates; monitor Dutchie release notes
GCP cost overruns	Low	Low	Set budget alerts at $150/month; optimize Vision API batch calls
Staff resistance to new workflow	Medium	Medium	Emphasize time savings; provide hands-on training; run parallel validation
New product catalog matching failures	High	Low	Human-in-loop for new products; build fuzzy matching library over time

Table 4: Risk assessment matrix

Success Metrics

Primary KPIs

- **Processing Time:** Target <15 minutes per delivery (current: 120+ minutes)
- **Error Rate:** <2% on expiration dates, costs, prices (current: 5-10%)
- **Staff Satisfaction:** >80% positive feedback on time savings
- **Coverage:** >95% of deliveries processed via automation by Month 3

Secondary Metrics

- **GCP Cost per Delivery:** <$3.00 target
- **METRC API Success Rate:** >98% successful enrichment calls
- **Supervisor Approval Time:** <10 minutes median (current: 15 minutes)
- **Catalog Match Rate:** >90% automatic (new products excluded)

Monitoring and Reporting

- BigQuery dashboard with daily processing counts, error rates, and timing
- Weekly email report to management with key metrics
- Monthly cost analysis and optimization recommendations
- Quarterly business review with ROI calculation

Budget Summary

One-Time Costs

Item	Cost
GCP project setup (IT labor)	$500
Initial development (4 weeks)	$6,000
Staff training (all locations)	$1,200
Documentation and runbooks	$800
Total One-Time	**$8,500**

Table 5: One-time implementation costs

Recurring Monthly Costs (3 Stores)

Item	Cost
GCP services (Storage, Vision, Functions, Run)	$110-135
Maintenance and support (5 hours/month)	$500
Slack Enterprise (existing, no incremental cost)	$0
Dutchie (existing, no incremental cost)	$0
METRC API (state-provided, no cost)	$0
Total Monthly	**$610-635**

Table 6: Recurring monthly costs at full scale

ROI Calculation

Labor Savings (3 Stores at Full Volume):

- Hours saved: 210-442.5 hours/month
- Average labor cost: $25/hour (blended rate)
- Monthly savings: $5,250-11,062.50

Net Monthly Benefit:

- Savings: $5,250-11,062.50
- Costs: $610-635
- **Net: $4,615-10,427.50/month**

Payback Period:

- One-time investment: $8,500
- Monthly net benefit: $4,615-10,427.50
- **Payback: 0.8-1.8 months**

Annual ROI (3 Stores):

- Annual net benefit: $55,380-125,130
- One-time investment: $8,500
- **ROI: 551-1,372%**

Recommendations

Immediate Actions (Week 1)

1. Approve GCP project creation and assign IT resource for IAM setup
2. Grant data analyst access to METRC NJ API credentials (already available)
3. Identify 3 high-volume vendors for pilot testing
4. Schedule kickoff meeting with store manager, product supervisor, and IT

Short-Term (Weeks 2-8)

1. Execute Phase 1-3 implementation plan
2. Run parallel validation with manual process
3. Train Store 1 staff on new workflow
4. Request Dutchie API access for Phase 4

Long-Term (Months 3-12)

1. Deploy to Store 2 and Store 3 as locations open
2. Expand automation to purchase order creation (upstream)
3. Integrate with inventory forecasting and reorder point calculations
4. Explore ML models for demand prediction and optimal order quantities

Conclusion

Automating the receiving process represents a high-impact, low-risk investment that directly addresses operational bottlenecks and enables scalable growth to three locations. The proposed GCP-based architecture leverages existing technology investments (Dutchie, METRC, Slack) and proven cloud services to deliver 87-92% time savings with a payback period under 2 months.

The phased implementation approach minimizes disruption, allows for validation at each stage, and builds internal expertise in cloud automation. By Month 3, The Hashery will have a production-ready

system processing 95%+ of deliveries automatically, freeing staff to focus on customer service, compliance, and strategic initiatives.

Next Step: Approve Phase 1 pilot to validate OCR accuracy and METRC integration with real invoices and manifests from our top 3 vendors.

Appendices

Appendix A: GCS Folder Structure

```
hashery-receiving/
├── store-1/
│   └── YYYY/
│   └── MM/
│   └── DD/
│   └── vendor-name/
│   ├── invoice-{number}.pdf
│   ├── manifest-{number}.pdf
│   └── processed/
│   ├── receiving-{number}.csv
│   └── metadata.json
├── store-2/
│   └── [same structure]
├── store-3/
│   └── [same structure]
└── templates/
├── dutchie-bulk-upload-template.csv
└── vendor-catalog-mappings.json
```

Appendix B: Dutchie CSV Column Mapping

CSV Column	Data Source	Example Value
catalog_product	Invoice + Catalog Match	"Blue Dream 3.5g Flower"
package_id	METRC Manifest	"1A40000000000001000012345"
quantity	Invoice Line Item	"24"
expiration_date	METRC API	"2026-12-31"
cost_per_unit	Invoice Unit Cost	"15.50"
price_per_unit	Invoice or Catalog	"45.00"
room	Static Value	"Receiving Room"
batch_number	METRC Manifest	"BATCH-2026-0225"
received_date	Processing Timestamp	"2026-02-25"
vendor	Folder Name	"Curaleaf"
invoice_number	PDF Filename	"INV-12345"

Table 7: Dutchie CSV column definitions

Appendix C: METRC API Endpoints Used

- `GET /packages/v1/{id}` – Retrieve package details including expiration
- `GET /packages/v1/active` – List active packages for inventory reconciliation
- `GET /transfers/v1/incoming` – Retrieve pending incoming transfers (future enhancement)

Appendix D: Technology Alternatives Considered

Alternative	Pros	Cons
Microsoft Azure	Familiar to some teams	Higher cost; not current stack
AWS	Mature OCR (Textract)	Learning curve; cost
Manual process	No tech investment	Does not scale; high error rate
Third-party receiving software	Turnkey solution	High monthly cost; vendor lock-in

Table 8: Technology alternatives analysis

Selected approach (GCP) aligns with current infrastructure, provides best cost-to-performance ratio, and offers flexibility for future enhancements.

References

[1] Google Cloud. (2026). Cloud Vision API Documentation. https://cloud.google.com/vision/docs

[2] Google Cloud. (2026). Cloud Run functions Overview. https://cloud.google.com/functions

[3] Dutchie Support. (2023). Receive inventory in Dutchie POS via bulk upload. https://support.dutchie.com/hc/en-us/articles/13551370860819

[4] State of New Jersey. (2024). METRC API Documentation. https://api-nj.metrc.com/documentation

[5] Slack API. (2026). Building bots with Bolt. https://api.slack.com/tools/bolt

[6] Google Cloud. (2026). Optical Character Recognition Tutorial. https://cloud.google.com/functions/docs/tutorials/ocr