NYC311 Infrastructure Hotspots — what this thing actually does

We start with a firehose of 311 tickets and end with:

- a **ranked list of work areas** (polygons on the map), and
- a **4-bullet brief** for each area (what's wrong, likely fix, urgency, crew).

Turning individual complaints into **actionable "repair zones"** for crews.

The 60-second mental model

1. **Find tickets that talk about the same kind of problem** (e.g., potholes) using semantic search, not just keywords
2. **Group those tickets on the map** so we get real-world hotspots, not scattered pins
3. **Score and rank** those hotspots by recency, volume, and how severe the category is
4. **Summarize** each hotspot into 4 tight bullets for ops/dispatch

Everything runs **inside BigQuery** with a couple of calls out to **Vertex AI** models through BigQuery ML. It's end-to-end SQL (plus a small Python cell upfront to create the dataset).

Why these choices (and not those)?

Semantic retrieval (embeddings + vector search) vs. keyword filters

- **We chose embeddings** because "pothole" can show up as "road surface hole," "asphalt depression," or just "big dip in the road." Embeddings place text into a vector space where *meaning* matters, not exact words.
- **We didn't rely on plain keywords** because they miss synonyms and slang, and they create brittle rules that break when phrasing changes.

Compact "seed" taxonomy vs. open-ended clustering of everything

- **We chose a small, explicit list of categories** (pothole, sidewalk damage, etc.) so retrieval is traceable and auditable ("this ticket matched the 'Pothole' seed").
- **We didn't do unsupervised topic discovery** because it's harder to explain to operations, and it can mix issues that crews handle with different tools.

DBSCAN for geospatial clustering vs. k-means

- **We chose DBSCAN** because it can discover clusters of arbitrary shape, ignore isolated noise points, and it works naturally in meters. That matters for streets.
- **We didn't pick k-means** because it assumes spherical clusters and forces us to pick "k" ahead of time. Street issues rarely form neat circles.

Convex hull polygons vs. "just show the pins"

- **We chose convex hulls** to draw clean polygons that wrap each cluster so a crew chief sees an **area** to work, not random dots.
- **We didn't keep point clouds** because they're not "dispatch-ready". We also buffer zero-area hulls so they're always valid polygons.

Snapshot-consistent time decay vs. "latest day only"

- **We chose a half-life decay** (recent reports count more, old reports fade but aren't forgotten).
- **We didn't only use the last week** because we'd miss persistent problems that flare up again.

Allow-list/QA gating vs. blind trust

- **We chose a precision floor** per category using weak labels (regex hints) to estimate a lower bound on retrieval quality. Only categories that meet the bar make it to the final, QC-filtered output.
- **We didn't ship everything** because demo safety and ops trust matter more than squeezing every category in.

How the pieces fit together (GCP tools)

BigQuery

- Stores raw NYC 311 data and all our project tables.
- Runs **SQL with geospatial functions** (e.g., ST_CLUSTERDBSCAN, ST_CONVEXHULL).
- Hosts **vector search** (finds nearest tickets to each seed embedding).
- Calls Vertex AI models via **BigQuery ML** (ML.GENERATE_EMBEDDING and ML.GENERATE_TEXT).
- Optional **vector index (IVF)** speeds up search when coverage is healthy.

Vertex AI

- **text-embedding-005**: turns text into vectors so "asphalt hole" ≈ "pothole".

- **gemini-2.0-flash-001**: writes the 4-bullet brief per hotspot.
- Both are used **remotely** through a BigQuery connection—no keys in code, and the data stays US-regional.

BigQuery Geo Viz / Looker Studio

- **Geo Viz**: quick map—paste the GeoJSON query (Section 28), color by priority_score, and we've got a live, ranked map.
- **Looker Studio** (optional): point to ops_worklist_with_briefs for an ops dashboard with polygons + briefs.

The pipeline

1. **Make a clean, recent slice** (Section 6): last ~180 days, anchored to the source table's own max date so everyone sees a consistent window.
2. **Embed tickets** (Sections 8–9): only a PII-safe ticket_text goes to the embedding model; lat/lon stays in BigQuery for clustering.
3. **Seed terms & seed embeddings** (Sections 11–12): our small, explainable taxonomy.
4. **Vector search** (Section 13): for each seed, pull the top similar tickets. If the ANN index is healthy, we use it for speed.
5. **Weak labels + threshold tuning** (13a–13b): regex hints estimate precision; we auto-pick conservative cosine cutoffs per seed and keep the best seed per ticket.
6. **Geo clustering per category** (Section 16): DBSCAN finds "where it's happening" on the map.
7. **Scoring & ranking** (Section 17): recency-weighted volume × category severity → priority_score.
8. **Polygons & boroughs** (Section 18): convex hulls, buffered if needed, with majority borough.
9. **Join into the final hotspots** (Section 19): everything we need for maps and briefs.
10. **Write the 4-bullet briefs** (Sections 20–22): low temperature, strict prompt, no street hallucinations.
11. **Evaluate & gate** (Sections 23–25): precision/cohesion checks → **allow-list** of production-safe categories → QC-filtered outputs.
12. **Publish ops view** (Section 26): one view to rule them all—ops_worklist_with_briefs.

What actually open to "see it"

- **Map**: run through Section 28 and paste into **BigQuery Geo Viz** (color by priority_score).
- **Worklist + briefs**: open the view **infra_prioritizer_us.ops_worklist_with_briefs** and read the top rows.

- **Dashboard** (optional): point Looker Studio to ops_worklist_with_briefs.

Guardrails we put in (so reviewers & ops can trust it)

- **US-region pinning**: every SQL cell starts with SET @@location='US';
- **Privacy**: only the ticket_text goes to the model; precise addresses don't.
- **Cost control**: bounded time window, top_k, and only enable ANN if the index has good coverage.
- **Reproducibility**: explicit table names and deterministic model temps.
- **Quality gates**: categories must meet a precision floor and minimum sample size to ship.

Knobs you can tweak (without breaking the story)

- **Window length** (tickets_base): 60–180 days, depending on budget and seasonality.
- **DBSCAN** (eps_m, min_pts in config): tighten/loosen how "close" points must be to form a hotspot.
- **Precision floor** (p_floor): higher = stricter allow-list, lower = more coverage with more risk.
- **Category weights**: bump safety-critical issues (e.g., sinkholes) higher or lower.
- **Max output** (max_clusters): how many polygons we want on the map.

Here's the mental map

- **BigQuery** is our database + analytics engine. We use it for SQL, GIS, vector search, and calling ML models.
- **Vertex AI** provides the models (embeddings and Gemini). BigQuery talks to Vertex **through a connection** we set once (IAM allows it), so SQL can say: "generate embeddings/text with that model."
- **Geo Viz** is the no-code map viewer for BigQuery results; **Looker Studio** is a simple BI/dashboard layer.
- **IAM and region** are the safety rails: we pin everything to **US**, and the service account has only the permissions it needs.

What comes out

- **Tables:**
  prioritized_hotspots_v2_qc (ranked polygons + scores)
  map_hotspots_v2 (GeoJSON-ready)
  hotspot_briefs_qc (4 bullets per hotspot)
- **View:**
  ops_worklist_with_briefs (joins polygons, scores, and briefs)
- **Models:**
  text_embedding_005 (remote) and gemini_text (remote)
- **(Optional) Vector index:**
  tickets_ix if we want faster ANN lookups

TL;DR

"We take 311 tickets, use AI to understand *what* each one is about, group nearby tickets into real-world hotspots on a map, score them by how active and serious they are, and then write a short brief per hotspot for the crew. It all runs in BigQuery with Vertex AI models plugged in—fast to run, easy to audit, and safe on cost and privacy. The final output is a ranked map layer and a dispatch-ready worklist."