ChatGPT

Disaster-Response Demo – Extended Narrative Timeline

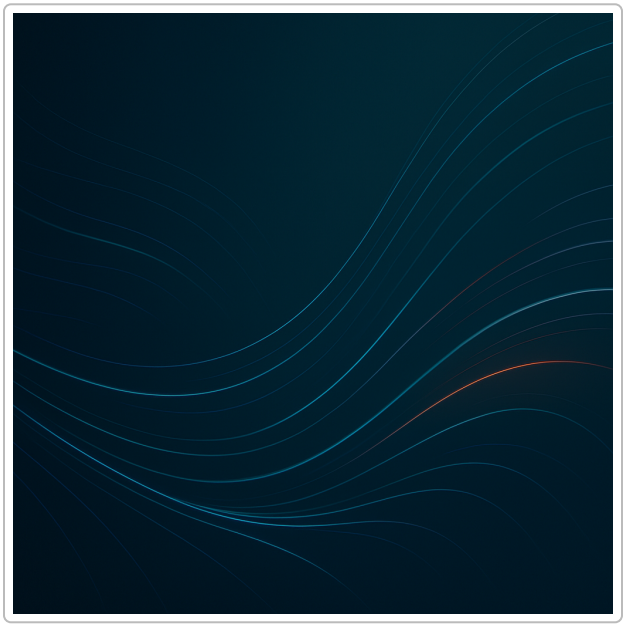
This document lays out an extended **five-plus minute** narrative for your Palantir Disaster Response demo. It follows the same style as earlier timelines, using time-block sections with suggested narration, visuals, and production notes. Feel free to fine-tune the durations slightly, but aim for a total runtime of roughly **5 minutes 40 seconds**. Each section clearly states what the viewer hears, what they see, and how to use the production system's templates and transitions to polish the presentation.

⏱ 0:00–0:30 — Introduction & Problem Context

Suggested dialogue:

> Hello, I'm **Ian Frelinger**, and I built this disaster-response platform for the Palantir Building Challenge. Emergency managers today juggle radios, maps and spreadsheets, which slows them down when minutes matter. Traditional common operating pictures often overload them with data, while front-line responders lack the high-level tools reserved for **Incident Commanders** [1]. My goal is to bring everyone onto the same page, delivering critical information without replacing professional judgment.

Visual: Open with the abstract swirling graphic you generated for the intro. It evokes the complexity of emergency response without showing any real-world scenes:



Production features:

- Apply the `intro_template` from your graphics templates to display the main title (“Disaster Response Platform”) and subtitle (“Palantir Building Challenge Project”) with a fade-in animation. Use the Emergency Response colour palette (info blue and neutral grey) for the text and the semi-transparent box.
- Start with a **fade-in transition** from black (`fade=t=in:st=0:d=1`) to gently introduce the scene. Keep the intro artwork static while the title animates, then crossfade into the next section.

⏱ 0:30–1:00 — User Roles & Needs

Suggested dialogue:

> Who uses this system? **Incident Commanders**, Operations and Planning chiefs, dispatchers and field responders. The **Incident Commander** stays at the top of the chain of command [2], but everyone else gains high-level situational awareness and AI recommendations. By democratising these tools, front-line teams can act faster and safer, while leadership retains clear authority.

Visual: Use two complementary visuals: first, the “User Roles” side panel from the final video, which lists **Incident Commander**, **Operations Planner** and **Dispatcher** down the left-hand side of the dashboard; second, the generic responder illustration. The panel anchors your narrative in the actual UI, while the silhouettes hint at the variety of field personnel:

Production features:

- Transition into this scene with a **slide-left effect** to indicate a new topic. The silhouettes should slide in from the right edge.

- Overlay **label_role** boxes above each persona with the role names (“Incident Commander”, “Planner”, “Responder”) using the Emergency Response theme colours. Each label fades in over 0.5 seconds.
- Optionally add a **lower third** at the bottom left with your name (“Ian Frelinger”) and title (“Developer & Presenter”) using the `lower_third_basic` template.

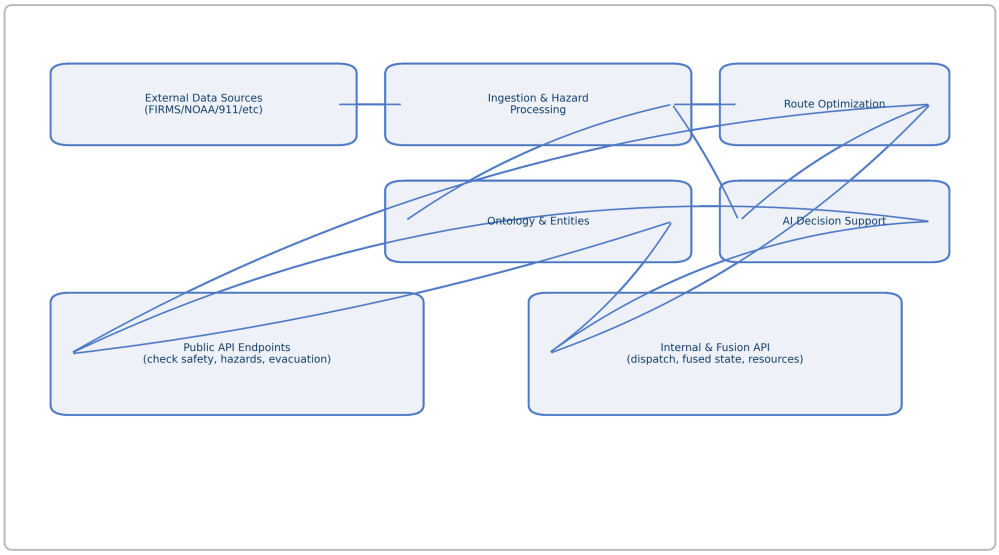
⏱ 1:00–1:40 — Data Flow & Technical Overview

Suggested dialogue:

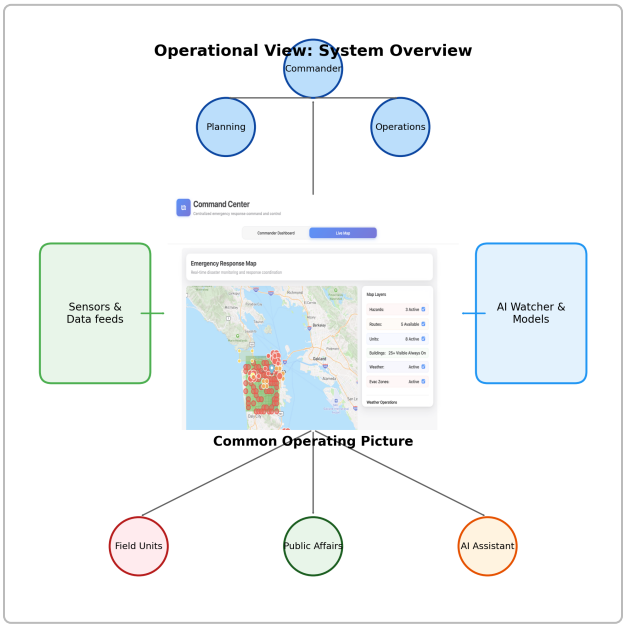
> Let’s look under the hood. A multitude of streams flows into our **ingestion layer**: NASA FIRMS satellite detections (brightness temperatures and confidence scores), NOAA wind and weather vectors, 911 call feeds, census-based population grids, live traffic conditions and GPS locations of response units. These feeds are ingested through **Foundry Inputs** and converted into uniform spatial indexes using **H3** hexagons at ~174-metre resolution. We fuse these data to generate hazard points and enrich them with population impact, wind speed/direction, terrain and infrastructure information. A machine-learning model then predicts how hazards will spread over the next two hours, creating dynamic risk zones.
>
> After ingestion and fusion, we calculate safe routes using a **Route Optimizer** built on an **A Star** pathfinder. It loads road network graphs, applies hazard buffers (default 500 m) and vehicle-specific constraints (weight, height and width) and returns the optimal path between origin and destination. By combining ML-powered risk scores and hazard-aware routing, we transform raw data into actionable insights in real time thanks to Palantir Foundry’s pipelines and ontology.
>
> The API layer then exposes these fused datasets through a clear set of **RESTful endpoints**, each implemented as a `@function` backed by **Foundry Functions**. Core endpoints include:
>
> - `GET /api/hazards` — summarises hazards, risk distribution, bounding boxes and populations at risk.
> - `GET /api/hazard_zones` — returns GeoJSON for map visualisation with timestamps serialised to ISO strings.
> - `POST /api/routes` — calculates a hazard-aware path using **A Star** with vehicle constraints and returns route geometry, distance, time and safety scores.
> - `GET /api/risk` — assesses risk around a coordinate buffer, listing nearby hazards with scores and distances.
> - `GET /api/evacuations` — retrieves evacuation status by zone, population at risk, progress percentages and nearest shelters.
> - `GET /api/units` and `PUT /api/units` — track emergency resources, update their status and report availability and assignments.
> - `GET /api/public_safety` — provides multi-language safety advisories, hazard proximity, recommended actions and evacuation routes for the public.
>
> Each endpoint loads processed datasets via `Input` decorators, invokes the hazard processor or route optimiser as needed, and writes results via `Output` decorators. This pattern allows dashboards, mobile apps and the AI assistant to consume the same unified

data without touching the processing pipelines. As you design diagrams, highlight only the relevant components to avoid clutter [3] while still conveying that the API is a gateway to the entire data fusion and decision engine.

Visual: Start with the technical architecture slide from the final demo. This dark-themed diagram divides the system into **Data Ingestion** (satellite, 911 and weather feeds), **Processing** (Foundry data fusion, hazard detection, risk assessment, real-time updates) and **Delivery** (React frontend, Mapbox 3D, WebSocket API). Overlay the API data-flow diagram afterwards to unpack the ingestion → processing → routing → AI pipeline.



Also consider adding the operational overview diagram showing the Incident Commander above Planning and Operations to reinforce the chain-of-command concept:



After the data-flow diagram, overlay callouts that label the core REST endpoints—**hazards**, **routes**, **evacuations** and **units**—on the pipeline. Use the `callout_info` template to position each label near the corresponding stage (processing, routing, ontology) so viewers can see how the API surfaces data for external applications.

Production features:

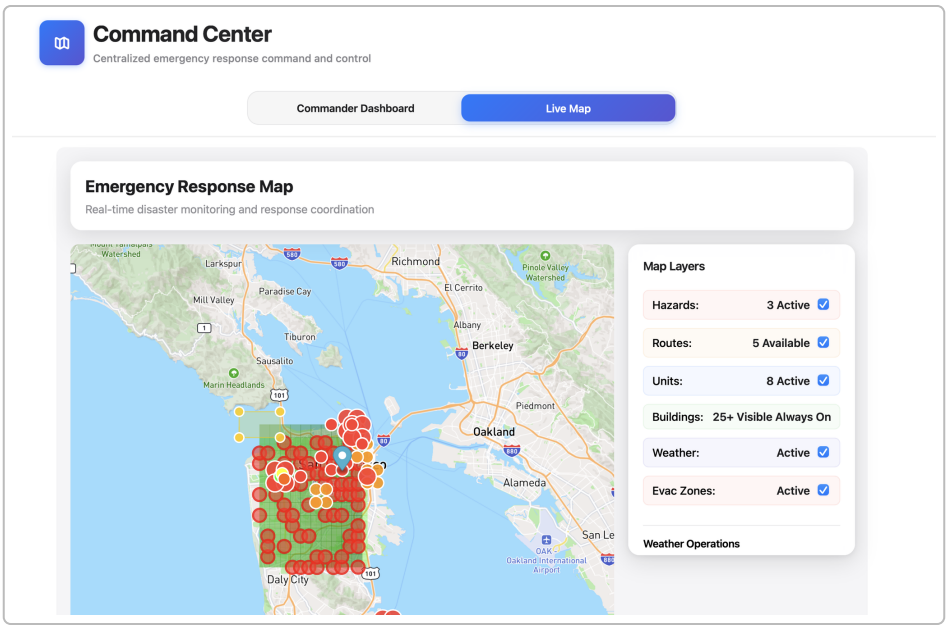
- Use a **slide-up transition** from the previous scene, with the diagram sliding up onto the screen while the personas slide down and disappear.
- Highlight each stage of the diagram as you discuss it by animating a glowing outline or subtle colour change around the relevant component. You can implement this using callouts or labels with the `label_component` template.
- Keep the diagram on screen for at least 10 seconds so viewers can digest the flow. Use a **graphics_fade** transition to gently fade the diagram out when moving to the next scene.

⏱ 1:40–2:20 — Hazard Detection & Triage

Suggested dialogue:

> On the Live Map, we visualise hazards in real time. Each red circle represents a fire detection or other hazard. We can toggle layers like hazards, routes and evacuation zones. Clicking a hazard shows its intensity and the population at risk. This is where triage happens: based on risk scores, we decide whether to monitor the area, shelter in place or evacuate.

Visual: Show the Live Map with hazard clusters, toggling hazards on and off. Include the “Active Hazard Detected” overlay from the final demo, which pops up when a fire is discovered. This card lists the hazard type, location, severity, risk score and number of buildings affected, grounding your conceptual description in the existing UI. Demonstrate clicking a hazard marker to view its details and turning layers like Routes and Units on or off:



Production features:

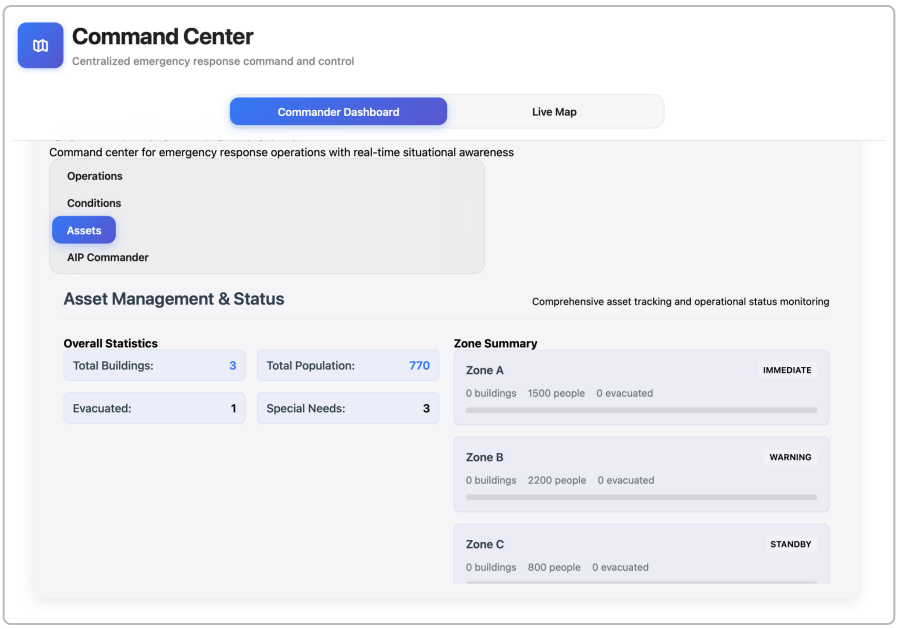
- Transition into this scene with a **crossfade**, merging the API diagram with the map view. As the map fades in, animate a **callout_alert** (“New Hazard Detected”) at the centre, bouncing briefly before fading.
- Use the **bounce** animation to draw attention to the hazard when you click on it. If possible, add a tiny pulsating glow around the selected marker while its details appear.
- Overlay a **status indicator** (from `label_status`) at the bottom to show the current risk level (e.g., “Risk: High”) with a slide-up animation.

⏱ 2:20–3:00 — Zone & Building Management

Suggested dialogue:

> Back on the dashboard, each card represents an evacuation zone. Here’s Zone A with 770 people and three buildings. The progress bars show how many residents have been evacuated versus those still in place. Clicking a zone or building lets you update statuses, see special needs and track refusals. Even though our current UI doesn’t draw zones on the map, the concept is clear: prioritise areas based on risk and population, and update building statuses as responders report back.

Visual: Use two views: the commander dashboard showing zone cards and building status counts, and the “Zone Management” legend overlay from the final video. The legend summarises the four evacuation statuses—Immediate, Warning, Standby and All Clear—connecting your narrative about prioritisation to something viewers see on screen:



Production features:

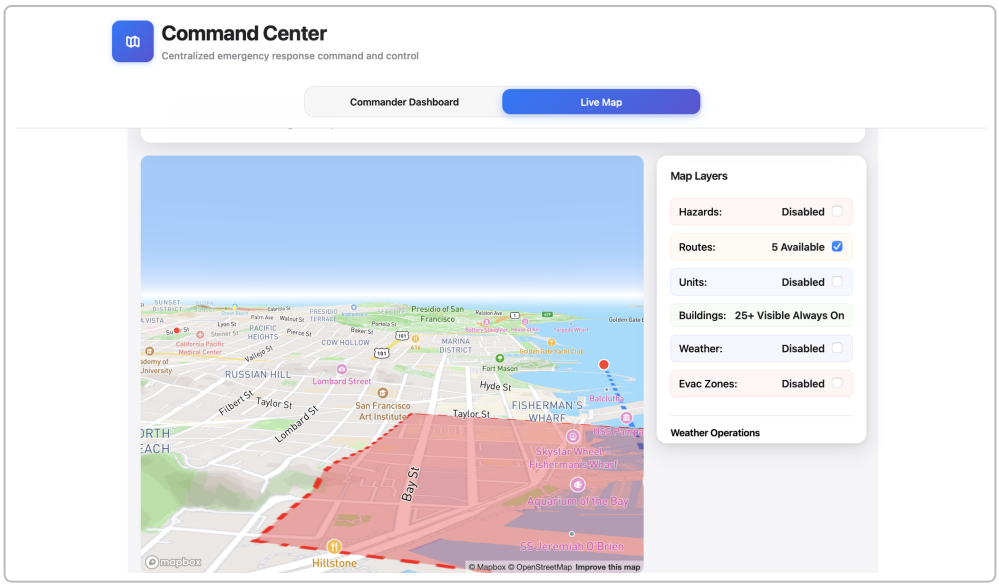
- Slide into this scene using a **slide-down** transition. The map view moves up and off screen while the dashboard scrolls into view.
- Highlight the progress bars with subtle fill animations so viewers can see them "fill up" as you describe evacuees leaving. Use the `label_status` template near each bar to show counts like "Evacuated: 1 / In Progress: 2".
- If you want to hint at the unimplemented zone-drawing feature, add a small **callout_info** ("Dynamic zone editing coming soon") to manage expectations.

⏱ 3:00–3:40 — Route Profiles & Dispatch Strategy

Suggested dialogue:

> Getting people out safely depends on the route. We calculate hazard-aware paths using the **A Star** algorithm. Different profiles balance safety and speed: **Civilian evacuation** maximises safety, avoiding hazards with big buffers; **EMS response** takes calculated risks; **Fire tactical** goes directly at hazards; **Police escort** secures the route. Even though the UI doesn't yet let us choose profiles, we can overlay the concept on the map. Notice how the blue path curves around the red hazard zones.

Visual: Superimpose a conceptual hazard-aware route over the map. Since route selection isn't available in the current UI, draw a blue line that curves around the red hazard zones to represent the **A Star** path. Use the route planning screenshot as a visual reference:



Production features:

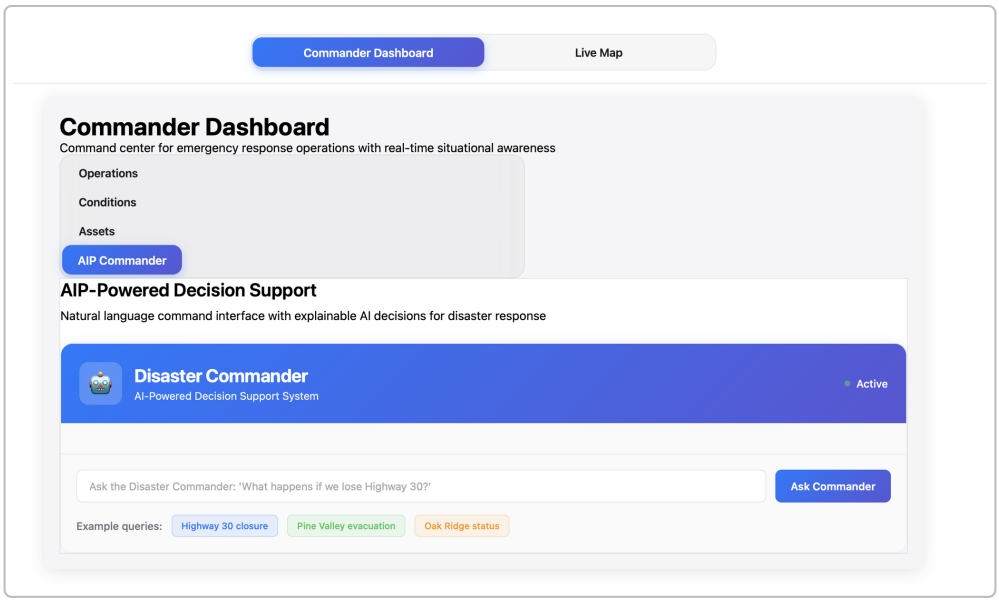
- Transition into this scene with a **wipe effect** from left to right, revealing the map with the route overlay.
- Use a **callout_info** panel on the right to summarise the four profiles. The panel can slide in from the right edge, with each profile appearing one after another (typewriter animation optional).
- Animate the route drawing using a **wipe animation** or reveal, so viewers can see the line draw from origin to destination, hugging the hazard buffers.

⏱ 3:40–4:10 — AI Decision Support & Replanning

Suggested dialogue:

> When conditions change, the system helps you decide what to do next. In the **AIP Commander** tab, you can ask natural-language questions like "What if we lose Highway 30?" The AI analyses hazards, unit availability and traffic, and recommends actions. You can accept a recommendation and replan on the fly. This keeps the Incident Commander in control but adds cognitive horsepower when the situation is fluid.

Visual: Show the AIP Commander interface. Type a sample query into the input field and imagine a recommendation card appearing:



Production features:

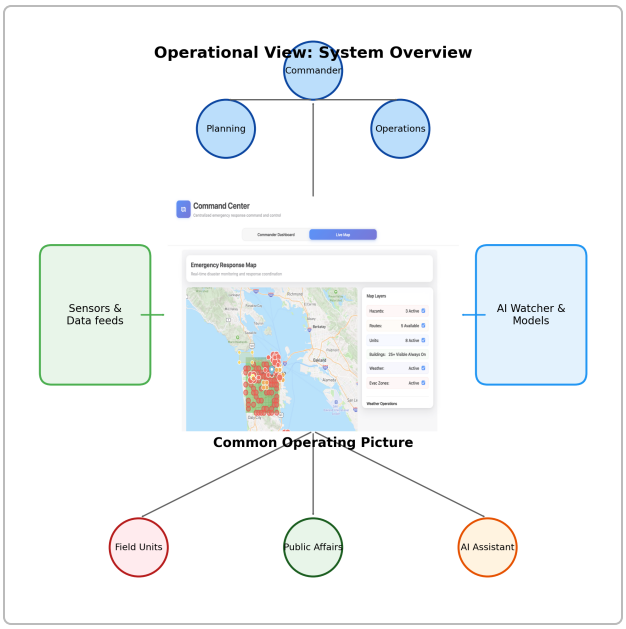
- Use a **slide-up** transition to reveal the AI tab. The map slides down and the AI interface slides up.
- Animate the typing with a **typewriter effect** so each character appears sequentially. After the question is typed, fade in a callout showing the hypothetical recommendation ("Evacuate via Route 3; stage EMS at Oak Ridge").
- Finish this scene with a **crossfade** into the next deep-dive section.

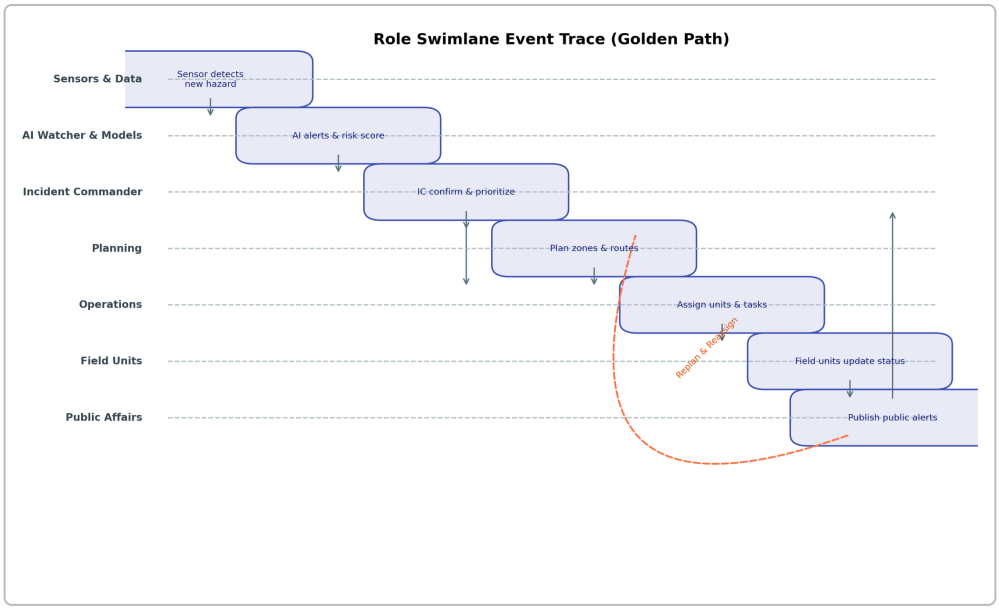
⏱ 4:10–4:50 — Technical Deep Dive & Foundry Integration

Suggested dialogue:

> Let's go deeper on the tech. Each hazard detection becomes a point on an **H3** grid. We enrich it with weather data, population impact and terrain. A machine-learning model predicts spread probabilities, and we update the ontology with risk levels and relationships. Foundry's pipelines make this possible: they ingest data, run transforms and manage objects like Hazard Zones, Routes and Units. When you issue an evacuation order, the ontology automatically links zones, routes and units, and events propagate across the system. This ensures everyone has a consistent picture.

Visual: Revisit the API data-flow diagram, but zoom into the hazard processing section and annotate it. You can also briefly show the role swimlane diagram to illustrate how different actors interact over time:





Production features:

- Use a **graphics_scale** transition to zoom in on the hazard processing box in the API diagram. Add small **labels** for H3 indexing, ML prediction and ontology updates using the `label_component` template.
- Layer the swimlane diagram on top of the diagram with a **dissolve** or **fade** effect, and highlight the row corresponding to the Incident Commander to show how orders propagate.
- Keep the diagrams on screen for 15 seconds while you describe the tech; then crossfade back to the dashboard for the next segment.

⏱ 4:50–5:20 — Impact & Value Proposition

Suggested dialogue:

> By fusing real-time data and AI guidance into one platform, we cut decision times, reduce manual coordination and improve compliance. Imagine evacuation completion increasing from 65 % to 90 % and response coordination time dropping from hours to minutes. Lower-level responders get tools previously reserved for senior leaders, and the system scales across hazards. Palantir Foundry enables this by managing data relationships and stream processing [1].

Visual: Show the green “Impact & Value Proposition” slide from the final demo. It lists measurable gains such as **Time Savings 40 %** and **Coordination 60 %**, plus bullet points for unified data sources, automated processing, scalable architecture, real-time updates and preserved human judgment. You can supplement this with a simple bar chart (evacuation completion rate vs. response time) overlaid on the asset dashboard.

Production features:

- Use a **fade-through-black** or **transition_smooth** effect to move into this scene. Present a bar chart that animates upward, using the `callout_success` template to call out key metrics.
- Maintain the Emergency Response colour scheme and professional typography. Conclude the visuals by emphasising the key numbers with bold text or coloured accents.

⏱ 5:20–5:40 — Conclusion & Next Steps

Suggested dialogue:

> In summary, this platform addresses a critical need in emergency response—integrating data, AI and intuitive interfaces so teams can act faster and safer. We’ve shown how hazards are detected, triaged, and managed; how routes could be planned; and how AI can guide decisions. With Palantir Foundry, this system is ready to scale. Thank you for watching—I’d love to discuss how we can pilot this solution with your team.

Visual: End with the hopeful sunrise artwork you generated. It’s a calm, optimistic image that symbolises new beginnings after a disaster:

Production features:

- Fade out the previous scene and fade in the conclusion image. Use the **fade-in** effect for the text (“Thank you” and a brief call-to-action). Keep the final image on screen for at least 5 seconds to let the message sink in.
- Add a **lower third** with your contact details or a link to continue the conversation. The lower third should gently slide up from the bottom and then fade out.

This extended timeline weaves together real interactions, conceptual explanations and diagrams. It preserves the narrative requested by your recruiter—why you chose this problem, who the users are, how the system works, and what value it delivers—while fitting within a longer video. You can adjust durations or swap visuals as needed, but following this structure will help you deliver a comprehensive, professional demo.

---

1 2 emilms.fema.gov
https://emilms.fema.gov/is_0200c/groups/450.html

3 API Flow Diagram: Best Practices & Examples | Multiplayer
https://www.multiplayer.app/distributed-systems-architecture/api-flow-diagram/