SMBHD
Mission Command: Real-Time Hero Deployment in Salesforce

Technical Architecture and Implementation Demo

Andrew Hood

January 20th, 2026

Justice Alliance Needs

- An increase in global threats necessitates rapid superhero response
- Current process: standard SF record page
- User pain points: too slow for real-time crisis management
- Stakeholder needs: a new, visual, “Mission Command” interface
  - Custom console that should live on a mission record page
  - Must allow mission leaders to visualize available roster of heroes
  - Must allow leaders to quickly deploy heroes to current mission
  - UI must update immediately
  - “Feel native”

Functional Requirements

- Display **hero roster** using visual cards (not a standard data table)
- Display each **hero card** in a responsive grid
- Allow filtering by power type
- Injured superheroes should be excluded from roster
- Visual status should indicate availability
  - If available: show green icon
  - If on mission: show red icon
- Deploy heroes with a single click
- Enforce maximum of 3 heroes per mission
- UI should be immediately responsive – no page refresh*

Technical Requirements

- Should feel like part of the “Native” Lightning Experience
- Apex Trigger on Mission Assignment object
- Scalable code
  - Apex logic is bulkified
  - Design patterns
  - Apex design pattern best practices (e.g. TriggerHandler)
  - Test-Driven Development
- UI should be immediately responsive – no page refresh*
- Parse error messages to make them user-friendly
- Must work in the UI and via the API

Clarifying Questions

- Do missions have a specific start and end date (or dateTime)? e.g. Should missions be able to be scheduled?
- What should happen to mission assignment records when their respective mission is marked as complete? Should we distinguish between mission assignments that are scheduled vs complete for reporting purposes?
- Can a hero be assigned to multiple missions simultaneously (assuming no time conflicts), or is it strictly one "active" mission at a time?
- What determines when a hero transitions from "Injured" back to "Available"? Is this process a manual update, or can I implement a recovery date field?
- What Salesforce license(s) does the Justice Alliance hold?

Answers and Assumptions

• Do missions have a specific start and end date (or dateTime)? e.g. Should missions be able to be scheduled?	• No – keep it simple.
• What should happen to mission assignment records when their respective mission is marked as complete? Should we distinguish between mission assignments that are scheduled vs complete for reporting purposes?	• Let records accumulate; possibly export to data warehouse for completed missions in a future enhancement via Apex batch job
• Can a hero be assigned to multiple missions simultaneously (assuming no time conflicts), or is it strictly one "active" mission at a time?	• Strictly one active mission at a time
• What determines when a hero transitions from "Injured" back to "Available"? Is this process a manual update, or can I implement a recovery date field?	• This process is a manual update.
• What Salesforce license(s) does the Justice Alliance hold?	• Best licenses available.

Architectural Design Considerations

Declarative-first implementation solutions I considered:

- Matching Rules and Duplicate Rules for duplicate prevention
- Roll-up Summary & Validation Rule for team capacity (max 3 heroes)
- Record-triggered flow for status updates
- Master-Detail relationship for the junction object

However, after reviewing these options, I realized some Salesforce limitations that necessitate an Apex-heavy approach.

Limitation #1 – Matching Rules

- **Plan**: Create a Matching Rule on Mission_Assignment__c matching on both Superhero__c **and** Mission__c
- **Problem**: Salesforce limits Custom Object Matching Rules to one lookup field
- **Result**: Cannot deploy Matching Rule or dependent Duplicate Rule
- **Fix**: Move uniqueness validation to Apex: `before insert`

Limitation #2 – Roll-Up Summary Timing

- **Plan**: Create Hero_Count__c field on Mission Assignment object and enforce logic using a Validation Rule

- **Problem**: Order of Execution problem:
  1. Before triggers run
  2. Validation rules run (reading stale count of heroes!)
  3. Record saves
  4. After triggers run
  5. Roll-Up recalculates (too late!!)

- **Result**: Failure scenario
  1. Mission has 3 heroes
  2. Validation Rule reads 3 (passes)
  3. Allows 4th hero to be added
  4. Roll-Up summary updates to 4 after database commit

- **Fix**: Aggregate SOQL query in `before insert` trigger for real-time count

Limitation #3 – Why Not Flow?

- **Plan**: After Apex creates Mission Assignment, an after-save Flow could update hero status to “On Mission”
- **Problem**: Flow runs in a separate execution context
  - When Apex returns values to LWC, Flow may not have committed
  - UI might show stale information until manual refresh
- **Result**: Conflicts with the requirement that UI must update instantly
- **Fix**: Handle this in Apex to **ensure the same execution context** runtime, we can control **when** this data is manipulated

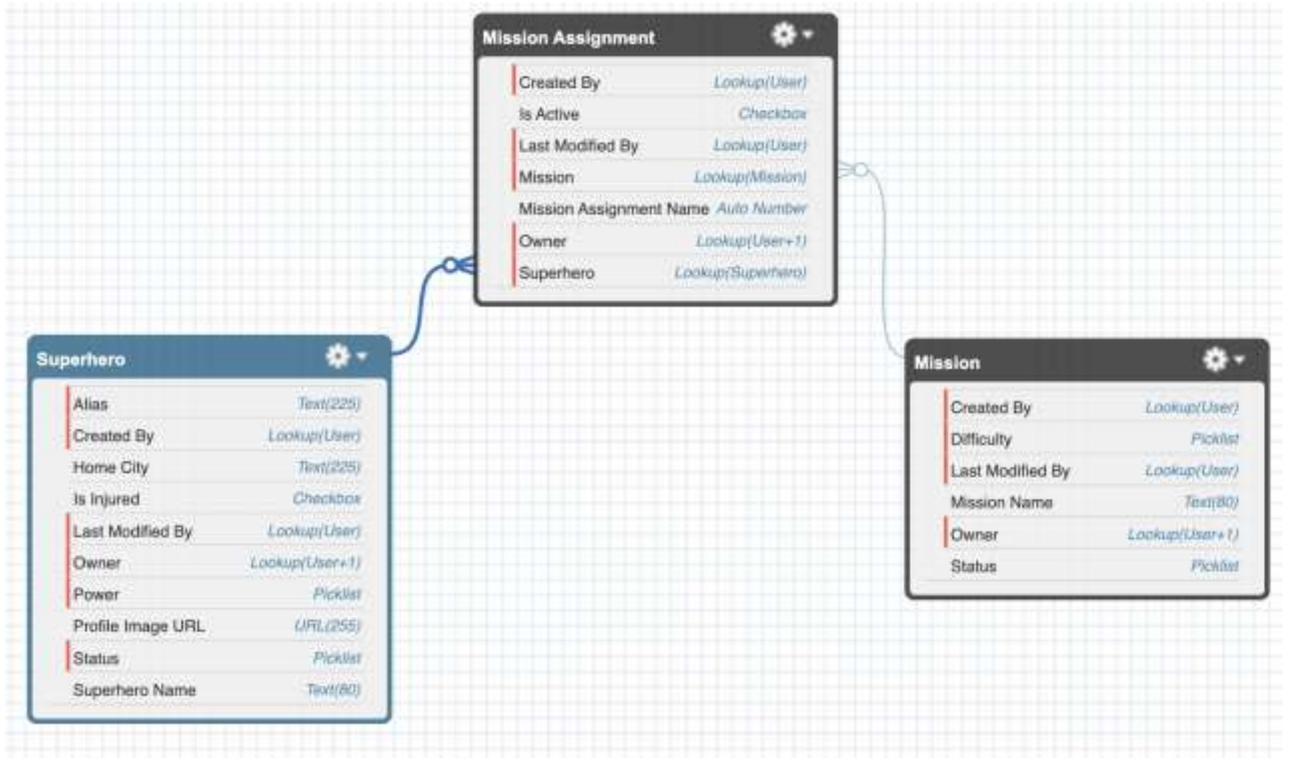
Limitation #4 – Master-Detail Row Locking

- **Plan**: Use Master-Detail relationships for the junction object
- **Problem**: M-D relationships lock the parent record during child DML
  - Two users deploy to same mission simultaneously
  - UNABLE_TO_LOCK_ROW errors
  - Do not want to implement manual locking and unlocking of records
- **Result**: UI failures during peak crisis response, the worst possible time
- **Fix**: Use lookup relationships: no implicit parent locking

Schema Design

Lookups > Master-Detail for the junction object

- No row locking
- Keep historical records
- Independent sharing
- Flexibility in reparenting
- Tradeoffs:
  - Must handle cascade delete
  - Manual sharing



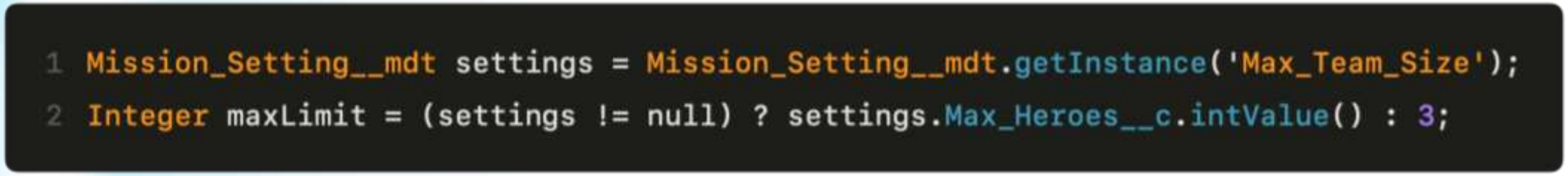
Mission Assignment	
Created By	Lookup(User)
Is Active	Checkbox
Last Modified By	Lookup(User)
Mission	Lookup(Mission)
Mission Assignment Name	Auto Number
Owner	Lookup(User+1)
Superhero	Lookup(Superhero)

Superhero	
Alias	Text(225)
Created By	Lookup(User)
Home City	Text(225)
Is Injured	Checkbox
Last Modified By	Lookup(User)
Owner	Lookup(User+1)
Power	Picklist
Profile Image URL	URL(255)
Status	Picklist
Superhero Name	Text(80)

Mission	
Created By	Lookup(User)
Difficulty	Picklist
Last Modified By	Lookup(User)
Mission Name	Text(80)
Owner	Lookup(User+1)
Status	Picklist

Custom Metadata

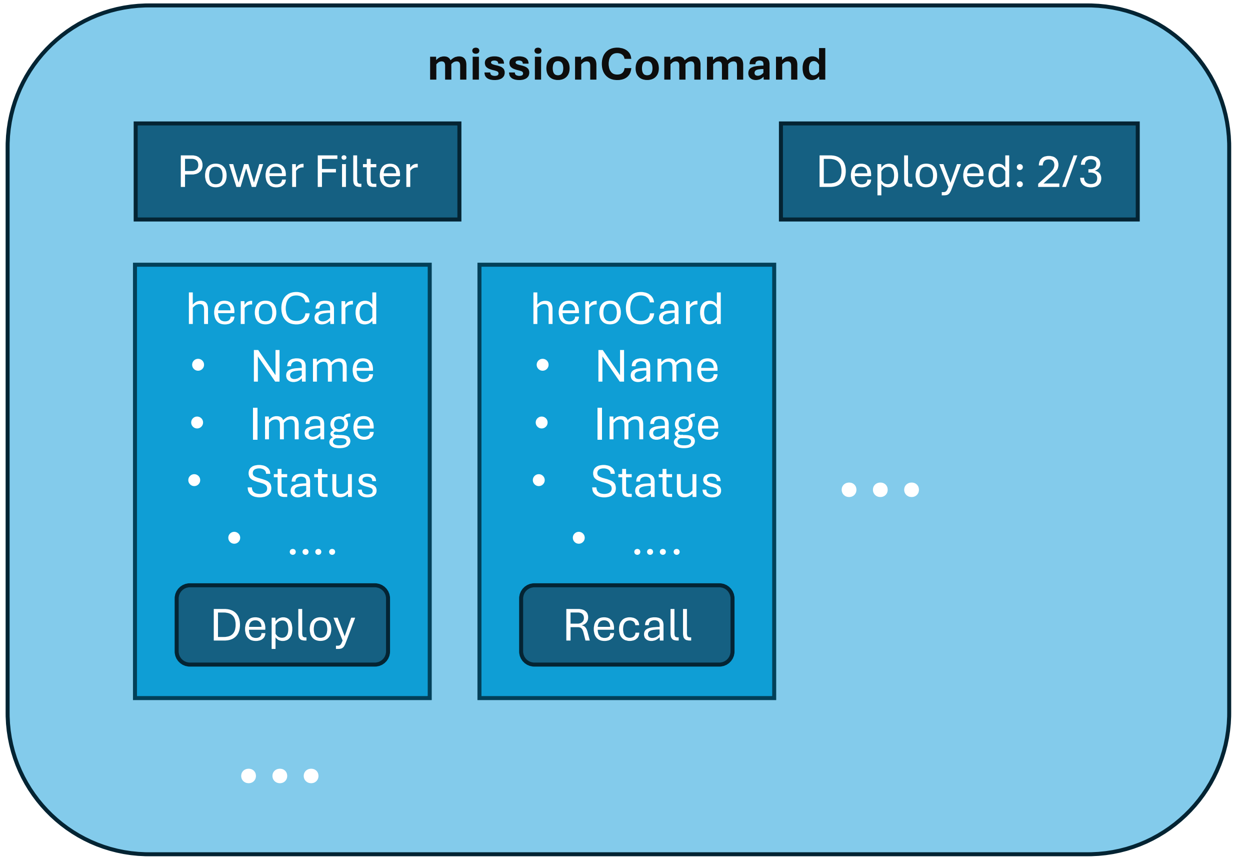
- Hard-coding the maximum hero team size to 3 requires deployment to change
- Easy win - use custom metadata type and custom field to set the team size
- Admins can now adjust the maximum team size without deployment



```
Mission_Setting__mdt settings = Mission_Setting__mdt.getInstance('Max_Team_Size');
Integer maxLimit = (settings != null) ? settings.Max_Heroes__c.intValue() : 3;
```

Lightning Web Component Design

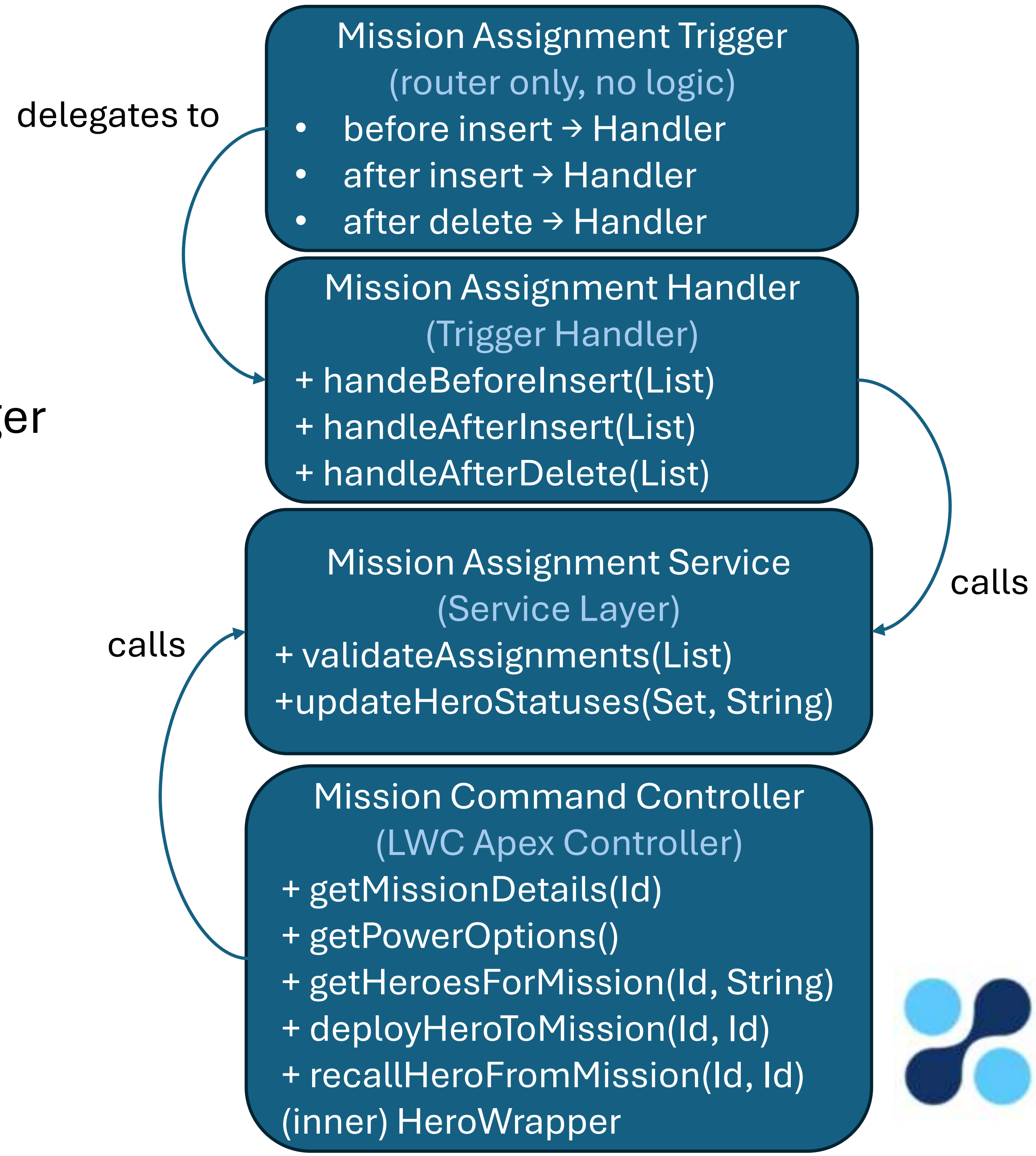
- Parent-Child Architecture:
  - Parent: missionCommand
  - Child: heroCard
- Data flow
  - Parent owns state, passes to child via @api decorator
  - @wire service allows for real-time data binding to:
    - missionId
    - powerFilter
  - refreshApex after hero deploy/recall
- Event flow
  - Children dispatch events
  - parent handles actions



Lightning Web Component Wireframe

Apex Architecture

- Separation of Concerns:
  - **Trigger** – routes context to trigger handler; contains no logic
  - **Handler** – extracts data, orchestrates flow
  - **Service Layer** – Contains all business rules
  - **Controller** – serves the LWC
  - **Wrapper** – inner class in controller helps serve data to LWC in consistent format



Mission Assignment Trigger
(router only, no logic)
- before insert → Handler
- after insert → Handler
- after delete → Handler

Mission Assignment Handler
(Trigger Handler)
+ handeBeforeInsert(List)
+ handleAfterInsert(List)
+ handleAfterDelete(List)

Mission Assignment Service
(Service Layer)
+ validateAssignments(List)
+updateHeroStatuses(Set, String)

Mission Command Controller
(LWC Apex Controller)
+ getMissionDetails(Id)
+ getPowerOptions()
+ getHeroesForMission(Id, String)
+ deployHeroToMission(Id, Id)
+ recallHeroFromMission(Id, Id)
(inner) HeroWrapper

Logic Deep Dive

The following sequence diagrams display:

1. Hero deploy flow
2. Hero recall flow
3. Deploy blocked flow

Live Demo

Demo of current functionality, including MVP of hero deployment and instant UI reaction. All business requirements have been implemented, as well as two big quality-of-life enhancements: hero recall and variable team size, adjustable by an administrator.

Proposed Future Enhancements

- Jests for the LWC
- Mission Completion Flow
  - Implement Record-Triggered Flow
    - When a mission status changes from “In-Progress” to “Complete”
    - Automatically switch all heroes assigned to that mission back to “Available” status
    - Not used in real-time UI, so background process is acceptable
- Injured Hero Recovery Date Flow
  - Implement Schedule-Triggered Flow
    - Runs daily and processes all heroes whose recovery date has passed
    - Automatically switch all heroes from “Injured” to “Available” status
    - Unchecks the “Is_Injured__c” flag
    - Sets the “Recovery_Date__c” field to NULL

Key Takeaways

- Clarifying questions helped to keep the implementation simple
- Identified platform limitations that ruled out most declarative tools
- Chose Lookup over Master-Detail for concurrency safety
- Implemented TriggerHandler design pattern instead of simple Trigger for maximum flexibility and ease of future development
- Developed bulkified Apex with Handler/Service/Wrapper patterns
- Delivered beyond requirements with Recall functionality (serving dual purpose) and variable team sizing with custom metadata
- Comprehensive Apex test coverage (93%), ready to deploy

Thank You

Questions?

Code is available here:
https://github.com/andrewdhood/SMBHD/tree/main/src

Andrew Hood
281-733-5217
andrewdavidhood@gmail.com
https://www.linkedin.com/in/adhood/