AMPLIFY DEVTOOLS PROJECT REVIEW

Summer 2025

Megha Narayanan

Context

Sandbox

- Creates real AWS resources in developer's account
- Provides isolated per-developer cloud environments
- Enables rapid iteration without affecting production
- Local development environment for Amplify Gen 2 backends
- Deploys a CloudFormation stack with resources
- Terminal-based interface for management and monitoring
  - Run with npx ampx sandbox
- Watches local code changes and automatically redeploys

Motivations

Terminal Interface Limitations:

- Dynamic configuration requires sandbox restart (e.g., enabling/disabling function logs)
- Multiplexed function logs make troubleshooting specific resources difficult
- Text-based output limits visualization capabilities and poses accessibility concerns
- Limited log streaming - only Lambda logs visible via terminal

AWS Console Alternative:

- Disjoint experience - developers switch between terminal and console
- Resource names don't align 1:1 with what developers see in code

Motivations: User Feedback

- **45%** of developers struggle to locate AWS resources created by Gen 2 and find their logs
- **68%** resort to adding console logs in Lambda functions and manually checking CloudWatch

From Developer Feedback:

- *“It is a challenge to find the list of resources deployed. Resource names are verbose which makes it difficult to find the correct resource without trial and error."*
- *“Log group naming is ridiculous especially if you have multiple environments and sandboxes. Trying to find logs for the actual instance I'm working with is a nightmare."*

Goals

1. **Unified Debugging Interface:** Centralized dashboard for logs from all Amplify services
2. **Interactive Resource Manager:** Visual hierarchy of resources with direct AWS Console links
3. **Per-Resource Logging:** Dynamic toggling of CloudWatch logs without sandbox restart
4. **Lambda Function Testing:** Clean interface to test functions with custom inputs

Core [P0/P1] Requirements:

- Local server, sandbox lifecycle management, console output streaming
- Resource visualization with logical hierarchy and console linking
- Dynamic log toggling for all resources with organized display
- AWS Console look and feel, Lambda testing with free-form input

Initial Design Process

Becoming a User!

- Created Amplify template app to experience sandbox firsthand (and also familiarize myself with the tool)
- Documented pain points in terminal-based workflow
- Experimented with AWS Console to understand ideal visualization

Proof of Concept

- Built simple console log streaming prototype in browser
- Focused on validating real-time communication approach
- Tested Socket.IO for bidirectional communication
- Validated technical feasibility of very basic architecture
- Also taught myself TypeScript ☺

Big Design Decisions

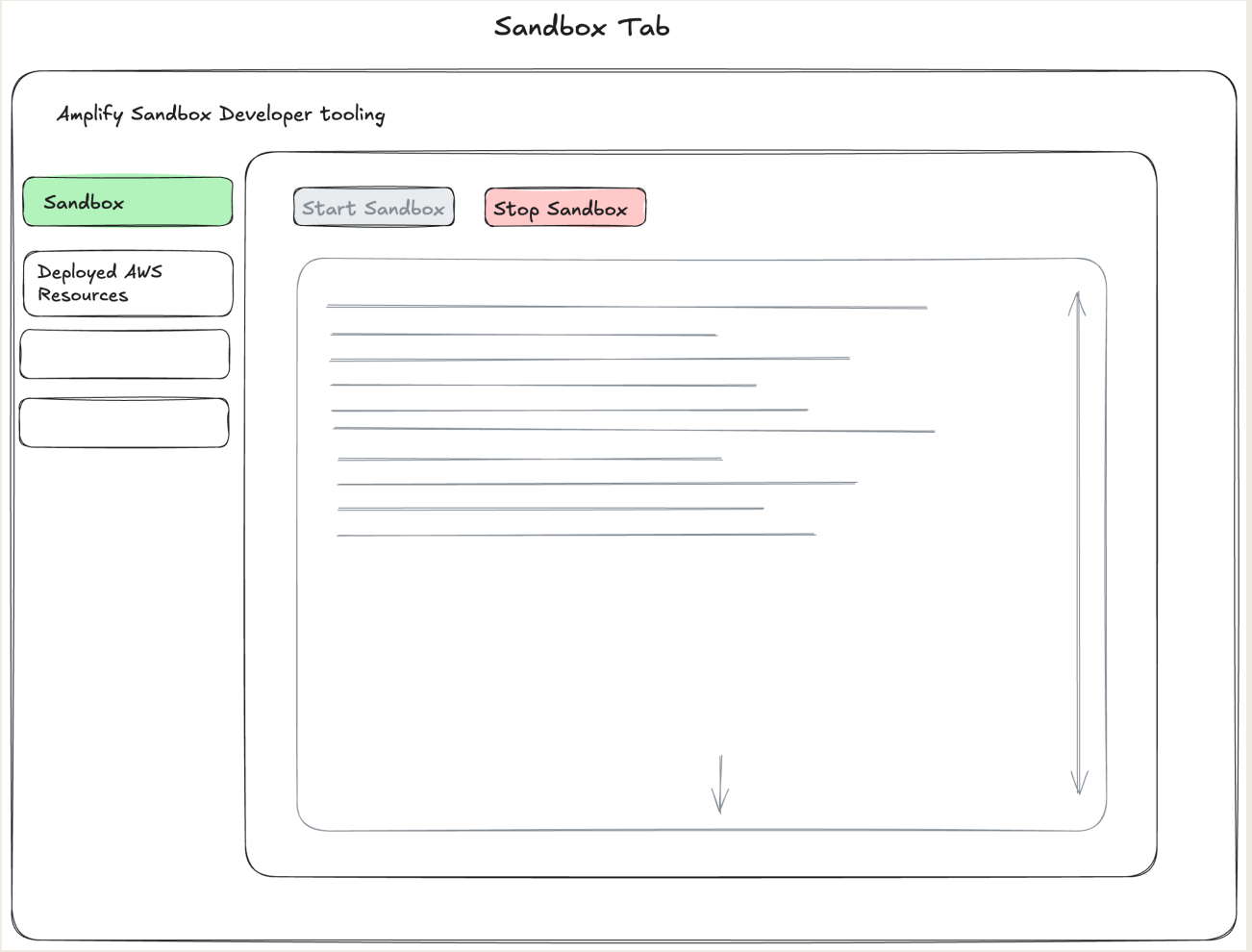
Web-based UI vs. Terminal Enhancement:

- Web UI provides better visualization capabilities
- Browser enables tabs, filtering, and interactive elements
- Local hosting preserves sandbox's per-developer isolation
- No additional cloud infrastructure required

DevTools/Sandbox Relationship:

- Implemented as standalone process that manages sandbox
- Allows DevTools to start/stop sandbox as needed
  - This was up in the air until the very end of my design process – I undertook this functionality, but don't think I fully understood until implementation the consequences of how complex this could be

UX Mockups


Sandbox Tab
Amplify Sandbox Developer tooling
Sandbox
Deployed AWS Resources
Start Sandbox
Stop Sandbox

Overall Design

Frontend Layer:

- React application with CloudScape Design components
- Service clients for backend communication

Communication Layer:

- Socket.IO for bidirectional real-time events
- Info and error propagation strategy across network boundary

Backend Services:

- Resource management service, log streaming service
- Sandbox lifecycle management
- Local storage management

AWS Integration Layer

- CloudWatch Logs integration for resource logs
- CloudFormation monitoring for deployment tracking
- Resource discovery and organization

FEATURES AND DEMOS!

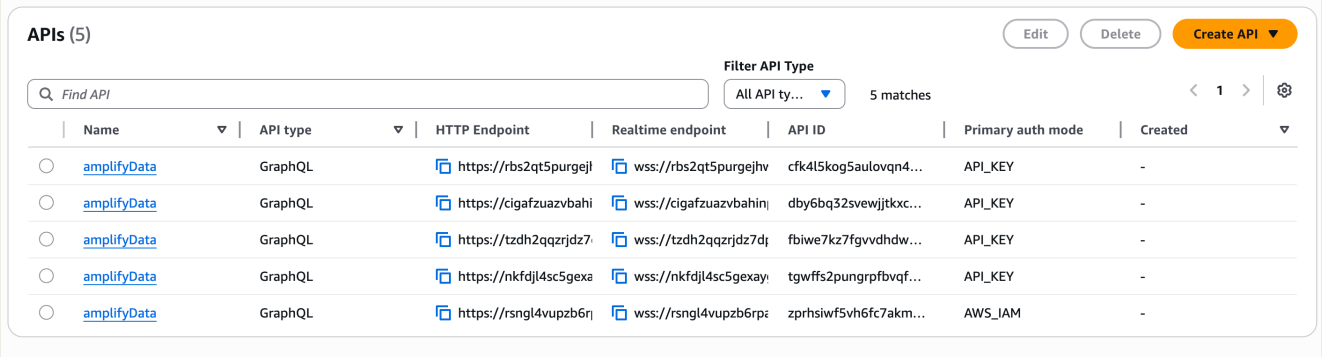
Demo 1

- Opening with npx ampx sandbox devtools
- Brief UI overview
- See streamed console logs, log filtering
- Error handling
- Go to resource console
  - Organization by service
  - Local cache helps minimize API calls
  - Filtering and searching
- Starting the sandbox, seeing state changes
- Making a backend change – see resources update

Demo 2

Resource Console Functions:

- Friendly Names + Custom Friendly Names
- AWS Console Links
- Why custom friendly names (which persist) are useful:



Demo 3

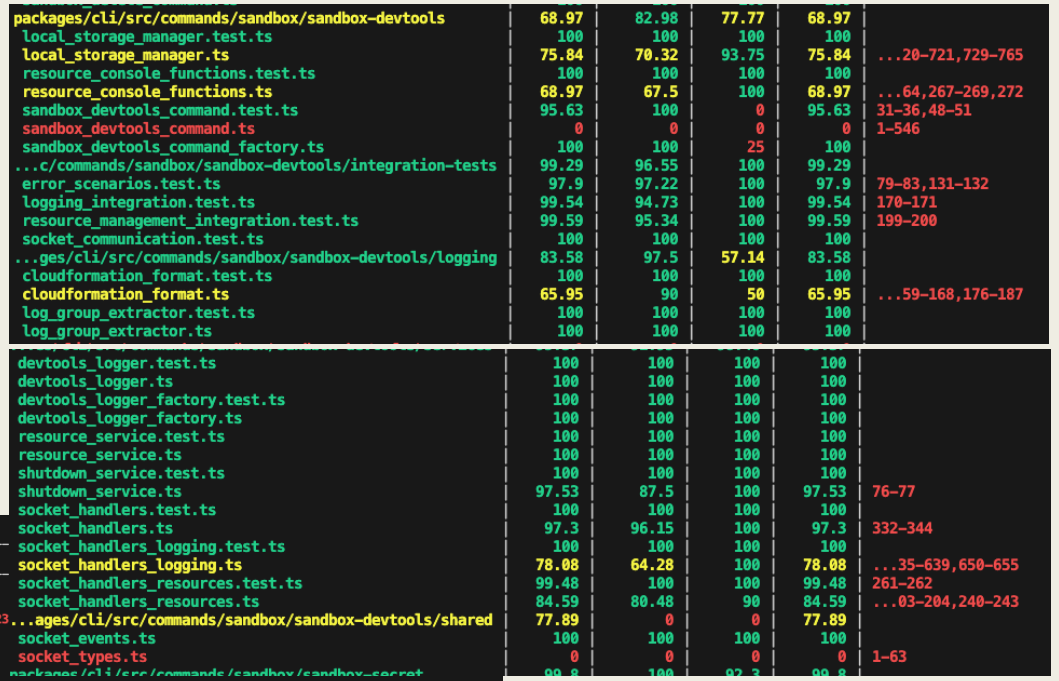
- Logging features:
  - Log settings modal
  - Toggling logs for a resource
  - Dynamic log stream selection
  - Adaptive log polling
  - Using polling vs a subscription
- Testing a Lambda function
  - Can do multiple at once

Demo 4

- Deleting the sandbox
- Seeing deployment streaming
- State update when deletion completes
- Stopping DevTools, see user-friendly disconnect error on UI

Testing

- Unit + integration tests
- Frontend and backend



packages/cli/src/commands/sandbox/sandbox-devtools	68.97	82.98	77.77	68.97	
local_storage_manager.test.ts	100	100	100	100	
local_storage_manager.ts	75.84	70.32	93.75	75.84	...20-721,729-765
resource_console_functions.test.ts	100	100	100	100	
resource_console_functions.ts	68.97	67.5	100	68.97	...64,267-269,272
sandbox_devtools_command.test.ts	95.63	100	0	95.63	31-36,48-51
sandbox_devtools_command.ts	0	0	0	0	1-546
sandbox_devtools_command_factory.ts	100	100	25	100	
...c/commands/sandbox/sandbox-devtools/integration-tests	99.29	96.55	100	99.29	
error_scenarios.test.ts	97.9	97.22	100	97.9	79-83,131-132
logging_integration.test.ts	99.54	94.73	100	99.54	170-171
resource_management_integration.test.ts	99.59	95.34	100	99.59	199-200
socket_communication.test.ts	100	100	100	100	
...ges/cli/src/commands/sandbox/sandbox-devtools/logging	83.58	97.5	57.14	83.58	
cloudformation_format.test.ts	100	100	100	100	
cloudformation_format.ts	65.95	90	50	65.95	...59-168,176-187
log_group_extractor.test.ts	100	100	100	100	
log_group_extractor.ts	100	100	100	100	
devtools_logger.test.ts	100	100	100	100	
devtools_logger.ts	100	100	100	100	
devtools_logger_factory.test.ts	100	100	100	100	
devtools_logger_factory.ts	100	100	100	100	
resource_service.test.ts	100	100	100	100	
resource_service.ts	100	100	100	100	
shutdown_service.test.ts	100	100	100	100	
shutdown_service.ts	97.53	87.5	100	97.53	76-77
socket_handlers.test.ts	100	100	100	100	
socket_handlers.ts	97.3	96.15	100	97.3	332-344
socket_handlers_logging.test.ts	100	100	100	100	
socket_handlers_logging.ts	78.08	64.28	100	78.08	...35-639,650-655
socket_handlers_resources.test.ts	99.48	100	100	99.48	261-262
socket_handlers_resources.ts	84.59	80.48	90	84.59	...03-204,240-243
...ages/cli/src/commands/sandbox/sandbox-devtools/shared	77.89	0	0	77.89	
socket_events.ts	100	100	100	100	
socket_types.ts	0	0	0	0	1-63

```
% Coverage report from v8
-------------------------------|---------|----------|---------|---------|-------------------
File                           | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-------------------------------|---------|----------|---------|---------|-------------------
All files                      |   89.35 |     83.7 |   74.68 |   89.35 |
 src                           |   91.75 |    91.78 |   86.66 |   91.75 |
  App.tsx                      |   93.33 |    93.05 |   92.85 |   93.33 | 131,166-175,199-210,224-225,230-231,236-23
  main.tsx                     |       0 |        0 |       0 |       0 | 1-9
 src/components                |   92.07 |     84.3 |   73.33 |   92.07 |
  ConfirmationModal.tsx        |     100 |      100 |     100 |     100 |
  ConsoleViewer.tsx            |   91.22 |       90 |   72.72 |   91.22 | 56-62,69-77,104-105,128
  DeploymentProgress.tsx       |   96.17 |     90.1 |    90.9 |   96.17 | 191-194,204-206,514-518,523-527
  Header.tsx                   |    94.4 |      100 |      60 |    94.4 | 61-62,65-66,69-71,74-75
  LogSettingsModal.tsx         |     100 |      100 |     100 |     100 |
  ResourceConsole.tsx          |   88.55 |    70.83 |   64.28 |   88.55 | 176-178,189-194,233,246-250,305-307,328-335,356-358,362-364,487,512-528,616-618,663,675,721-737,774-776
  ResourceLogPanel.tsx         |    90.8 |    86.66 |    87.5 |    90.8 | 93-98,106-111,119-124,163,184-193,290-291,381-383
  SandboxOptionsModal.tsx      |   90.84 |    73.33 |      60 |   90.84 | 66-67,70-71,74-83,152,156
 src/contexts                  |   66.66 |    55.55 |   83.33 |   66.66 |
  socket_client_context.tsx    |   66.66 |    55.55 |   83.33 |   66.66 | 64-71,80-83,94-97,108-111,122-125
 src/hooks                     |      88 |    71.42 |   86.66 |      88 |
  useResourceManager.ts        |      88 |    71.42 |   86.66 |      88 | 75-78,81-90,102-103,105-106,215-216,259,263-266
 src/services                  |    73.6 |    84.41 |   65.88 |    73.6 |
  deployment_client_service.ts |     100 |      100 |     100 |     100 |
  logging_client_service.ts    |   42.02 |      100 |   28.57 |   42.02 | 57-66,73-74,81-82,95-96,103-104,113-122,153-155,164-166,186-188,197-199
  resource_client_service.ts   |   77.35 |      100 |   72.72 |   77.35 | 31-32,47-54,61-62
  sandbox_client_service.ts    |   78.57 |      100 |   66.66 |   78.57 | 51-52,65-66,83-84,91-92,159-162
  socket_client_service.ts     |   87.58 |       75 |   71.05 |   87.58 | 23,52-53,173-183,222-224,248-249
  test_helpers.ts              |       0 |      100 |     100 |       0 | 2-23
 src/test                      |    94.2 |    80.39 |    92.3 |    94.2 |
  setup.ts                     |       0 |        0 |       0 |       0 | 1-9
  test-devtools-server.ts      |   96.65 |       82 |      96 |   96.65 | 40-41,332-338
-------------------------------|---------|----------|---------|---------|-------------------

 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

Future Improvements

Enhanced Resource Visualization

- Interactive resource dependency graph visualization
- Resource usage metrics and cost insights

Advanced Developer Productivity

- AI-powered log analysis and error resolution suggestions
- Template generation for common resources
- Integrated documentation viewer
- Schema validation for resource configurations

Data Management & Seeding

- Visual data seeder interface
- Record-and-replay seed operations

More Future Improvements

Architectural Improvements

- Dynamic port selection to support multiple instances
- Multiple sandbox management from single interface
- More flexible UI that allows you to view more things at once

Performance & Scalability

- Pagination for large resource sets (and log sets)
- Optimized rendering for resource-intensive views

Logging and Lambda Testing

- Log group dropdown
- Being able to upload Lamba inputs from files (for large inputs)

Challenges Faced

- Complete lack of familiarity with the WHOLE tech stack at the beginning of the internship
- Battling dependencies and dependency check
  - Had ~30 failed GitHub workflow runs before ever passing the dependency check for my first PR
  - Most of you know this already because I assume you had to turn off your email notifications for them ☺
- This also meant I opened my first PR super late – I was nearing done with my implementation when tests for my first PR passed
  - Wasted a lot of time merging changes
- Wasted lots of time refactoring as I optimized architecture
  - E.g. went through 8 iterations of my printer.log override before getting a CR comment that prompted me to reconsider the whole approach and change it inject a custom logger instead

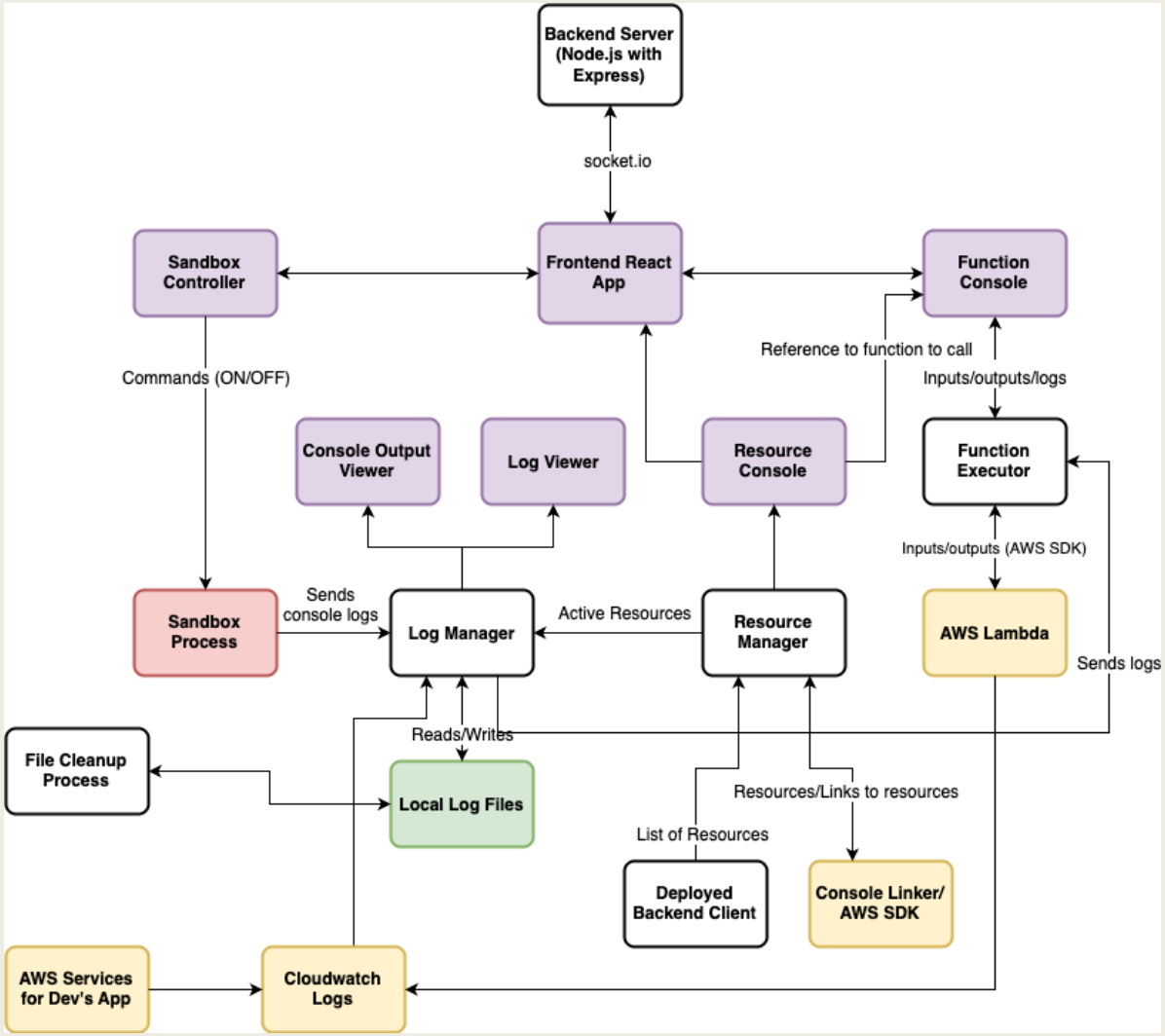
More Challenges Faced

- Open ended project nature – expected behavior was sometimes unclear and I had to do design on the fly
  - E.x. decision on sandbox management
- I severely underestimated how long it would take for:
  - Code reviews: Over my 8 PRs, I submitted over 20,000 lines of actual code– NOT including changes to package_lock.json and other config files.
    - This is a TON of code to read!
    - ~10000 (50%) of these lines were tests
  - Writing unit tests
  - Verifying test coverage
- Project goals were so open ended: it was hard to pick a point where I just had to call it quits on implementation and focus on optimizing.

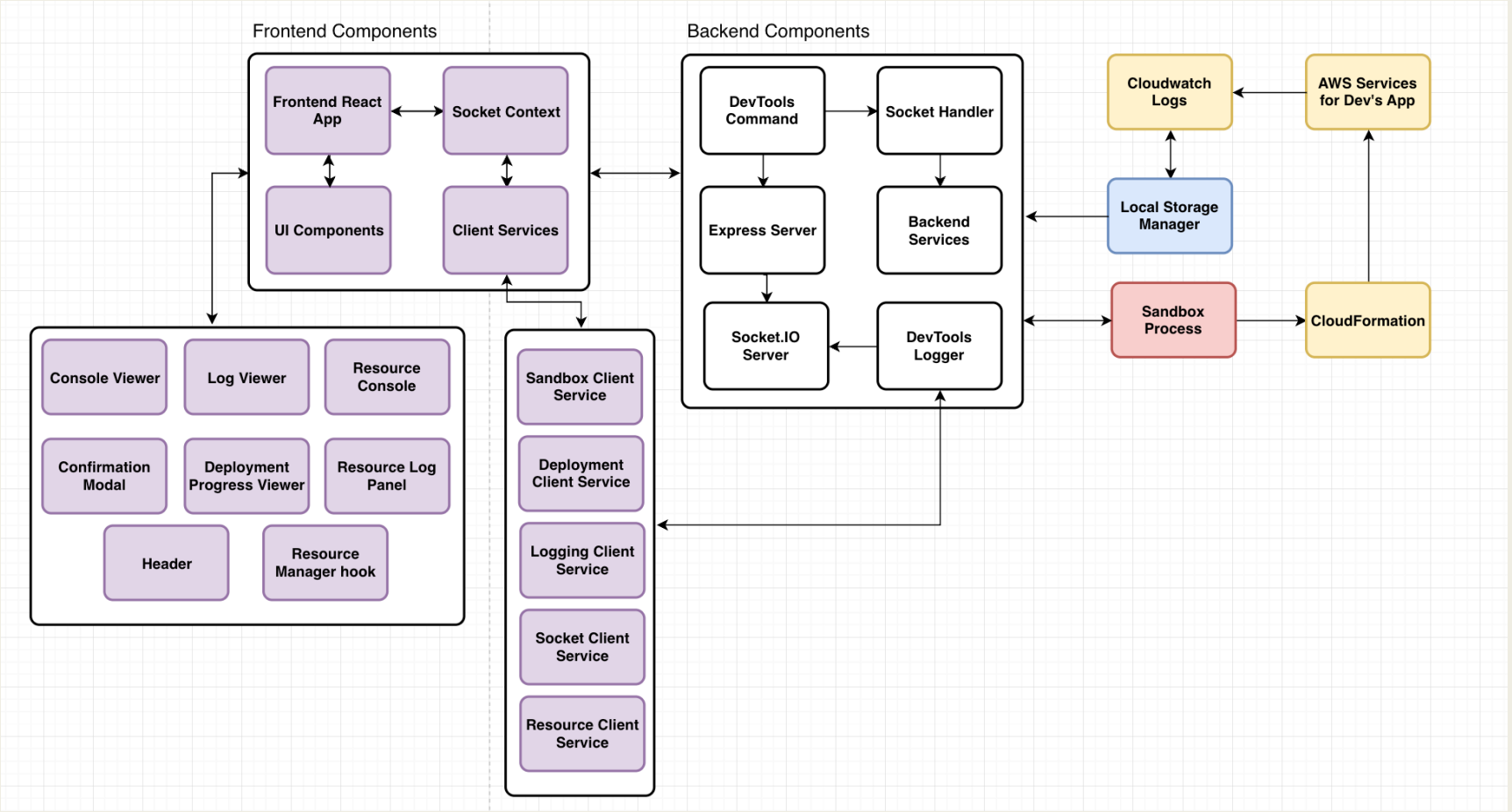
Initial Architecture Plan

Client-server architecture

Shows how little I knew at the beginning– there's nothing connected to my backend process!



Final Architecture



What I Learned

This project pushed me across the full development spectrum:

- UX design and wireframing
- Frontend development with React and CloudScape
- Backend architecture with Node.js and Socket.IO
- AWS service integration and cloud architecture

Testing:

- Building the test pyramid from unit to integration
- Creating specialized test utilities for Socket.IO
- Writing deterministic tests for not-necessarily-deterministic behaviors

What I Learned (cont.)

In addition to my basically 0-to-100 technical growth, I also learned:

- How to handle a truly massive amount of code!
  - And to be less intimidated by starting with a blank file
- What it means to dynamically restructure and optimize architecture as a project grows
- Writing modular, easily readable and testable code
- When to take ownership, and when to ask for help
- Not to get too attached to my code
  - Wrestling with trying to get DevTools to manage an external sandbox process for over a week before just scrapping the whole idea

Knowledge Transfer + Handoff

Pushed to a feature branch, not main

DevTools is a HUGE project, there is a ton of stuff we could add before production

Made a detailed handoff doc: https://quip-amazon.com/vkUlAK8MTTix/Amplify-CLI-Sandbox-DevTools-Project-Offboarding-Knowledge-Transfer

- Comprehensive documentation
- Detailed architecture overview
- Known limitations and workarounds
- Future development roadmap

THANK YOU!