SAND2019-11226C

Sandia National Laboratories

Hellhound: A Modern Infrasound Tool Suite

Challenge: A toolset for a new ***infrasound*** mission that must be modern, extensible, and delivered to a ***high security environment***...

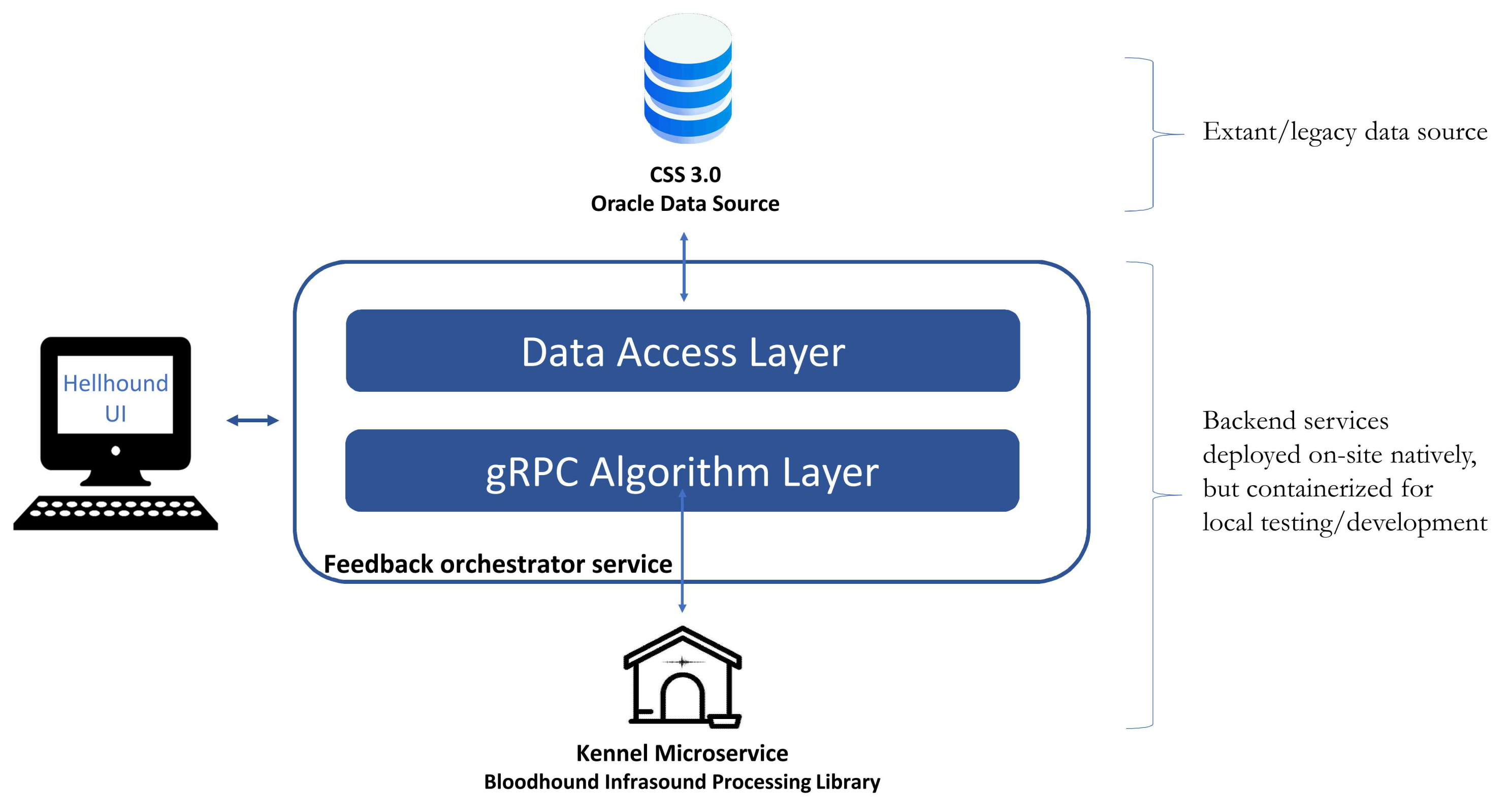
Introduction

A familiar story

The customer has a new mission but only legacy tools that were neither designed for modern systems nor the new mission. The geopolitical climate, however, is such that they *need to begin work on the mission immediately*.

The needs:

- Visualize, process, and analyze infrasound data
- Powerful enough for a scientist, accessible enough for an analyst
- Extensible: an architecture that allows arbitrary data types and algorithms, in any language to be plugged in
- Deployed to high security deployment environment
- **Time between initial demand to minimum viable product: 3 months**

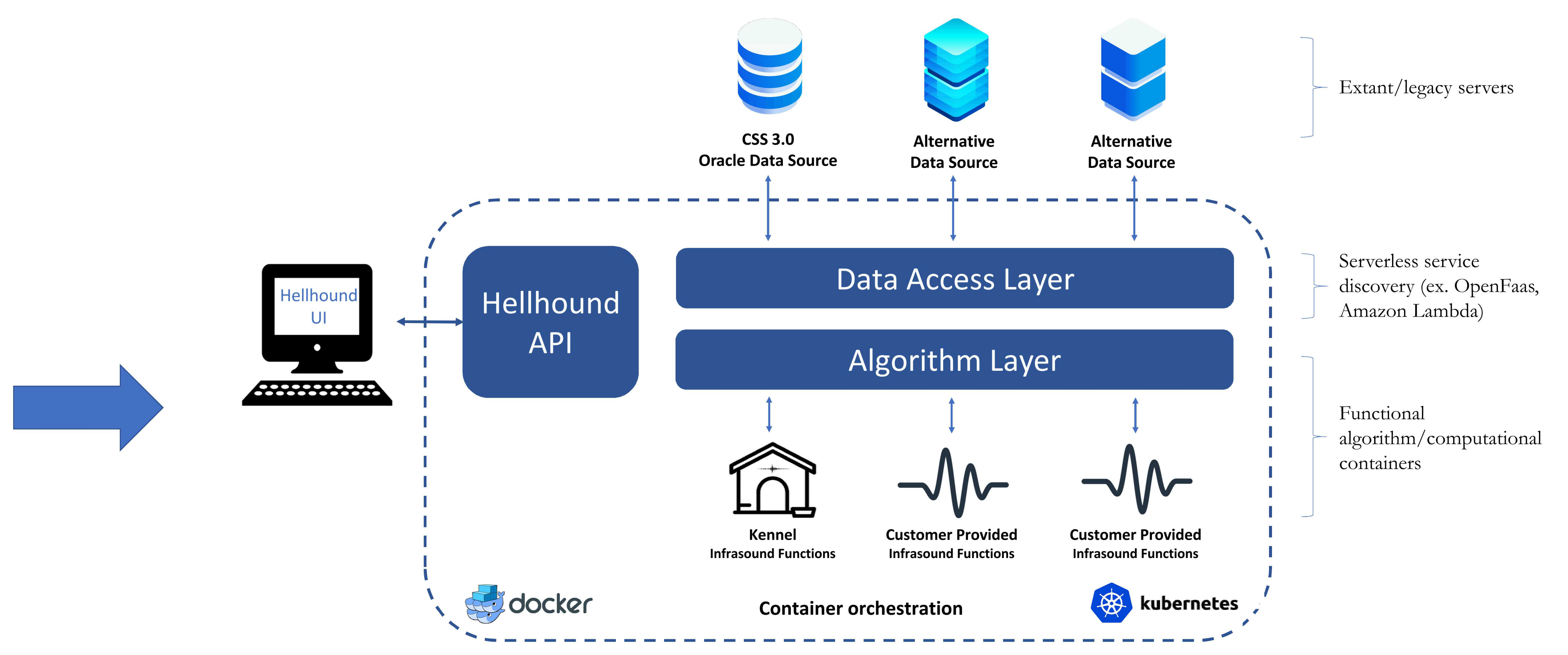

CSS 3.0
Oracle Data Source
Extant/legacy data source
Hellhound UI
Data Access Layer
gRPC Algorithm Layer
Feedback orchestrator service
Backend services deployed on-site natively, but containerized for local testing/development
Kennel Microservice
Bloodhound Infrasound Processing Library

Initial Architecture: Simple, such that any analyst could deploy the entire stack, but also designed for the future: modular pieces that are Dockerized for local testing/deployment

Challenge:
We need a **modern technology stack**

Solution:

- Browser based UI implemented using React and TypeScript
- Clojure backend API to interact with microservices: the support of Java with a functional and easy Lisp dialect
- Python wrappers to interact with pre-existing scientist code
- GitLab Continuous integration to test and deploy code


CSS 3.0
Oracle Data Source
Alternative Data Source
Alternative Data Source
Extant/legacy servers
Hellhound UI
Hellhound API
Data Access Layer
Algorithm Layer
Serverless service discovery (ex. OpenFaas, Amazon Lambda)
Kennel Infrasound Functions
Customer Provided Infrasound Functions
Customer Provided Infrasound Functions
Functional algorithm/computational containers
docker
Container orchestration
kubernetes

Eventual Architecture: Break apart the already modular components of the original architecture and fully embrace modern technologies and cloud-native deployment once the deploy environment supports it

Challenge:
The **architecture needs to be flexible and extensible**, ready for the developers on site to be able to insert arbitrary algorithms and data types into the system

Solution:

- Carefully defined interface layers with opportunities to dynamically plug-in extensions
- GraphQL for the client request layer: the API can by dynamically composed of different schemas
- gRPC+protobuf: High-performing, streaming gRPC microservices can be plugged into the algorithm API access layer
- When transmitting large amounts of data, use encoding optimizations such as msgpack and transit

Challenge:
We need to deliver robust, high-quality software, **quickly**

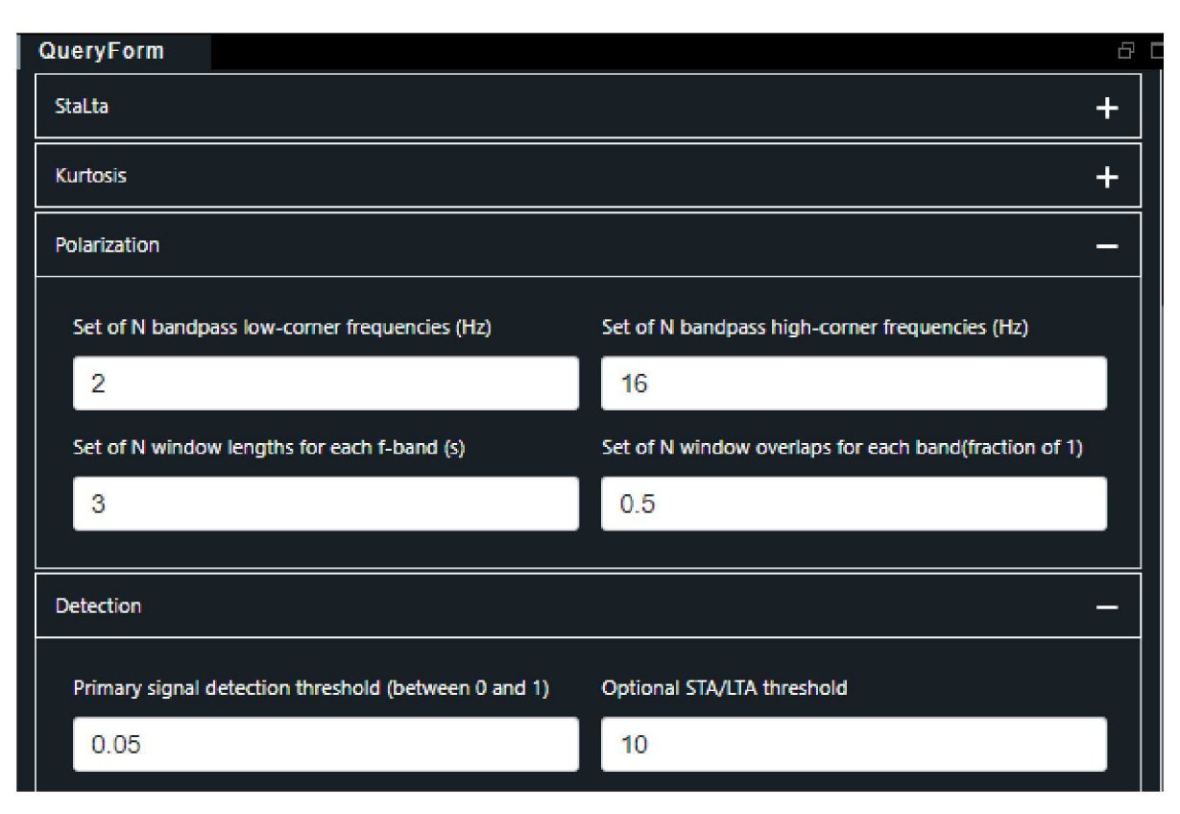
Solution:

- Agile development following best practices, merge requests, continuous integration, and testing
- Be flexible in software design and ensure that all developers are comfortable with the design and technologies from the start
- If a particular strategy becomes a pain point, be ready to try an alternative
- Collaborate and cultivate open-source and inner-source technology

Challenge:
We want to use **modern deployment technologies in a high security environment** such a Docker, Electron, and continuous integration/deployment. However, at the on-prem deployment environment, these technologies are either not supported or not approved.

Solution:

- Compromise accordingly: deliver **value now** but design for **value later**
- Initial design of code optimizes for easy on-prem deployment, with relatively few dependencies and executables
- Architect code in such a way where it can be modularized and scaled
- Develop a browser based application that can be transitioned into Electron
- Internally Dockerize services for CI, testing, and sandbox deployment



QueryForm

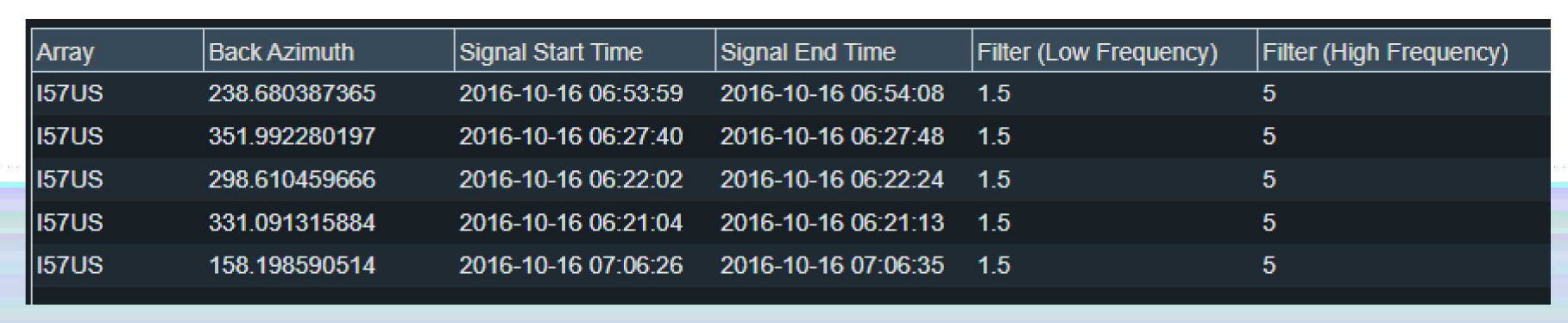
StaLta	+
Kurtosis	+
Polarization	−

Set of N bandpass low-corner frequencies (Hz)	2
Set of N bandpass high-corner frequencies (Hz)	16
Set of N window lengths for each f-band (s)	3
Set of N window overlaps for each band(fraction of 1)	0.5

Detection −

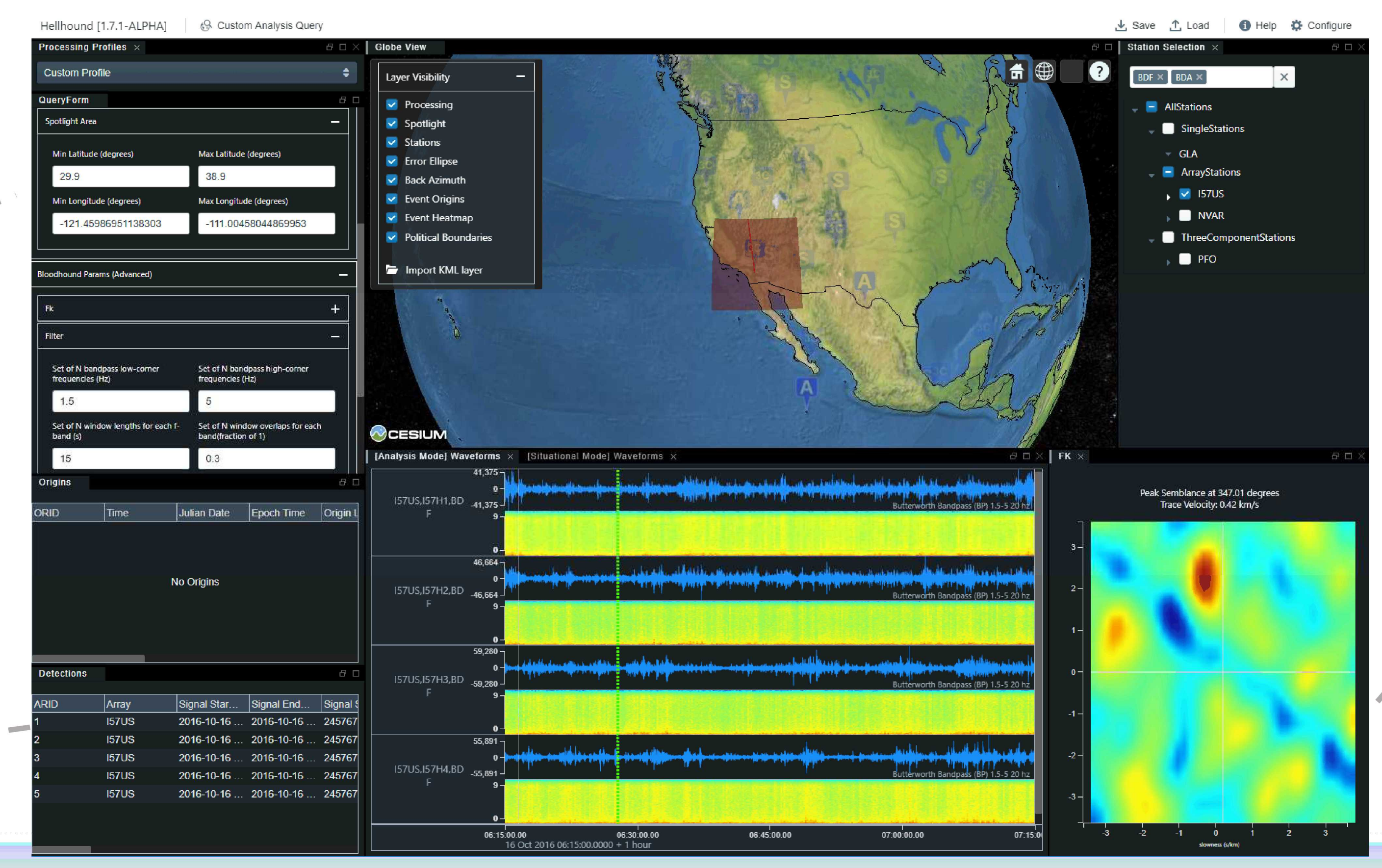
Primary signal detection threshold (between 0 and 1)	0.05
Optional STA/LTA threshold	10

Configure the underlying algorithm library to find origins and detections using different parameters

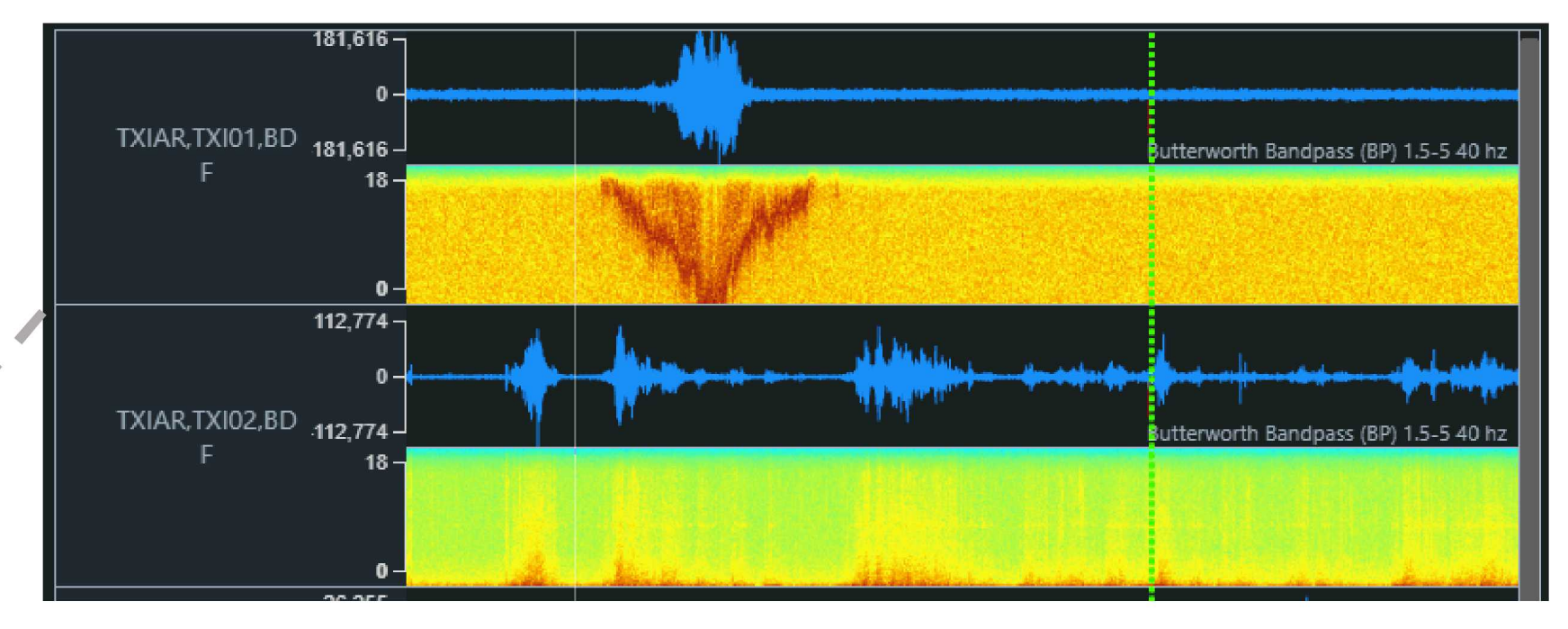


Array	Back Azimuth	Signal Start Time	Signal End Time	Filter (Low Frequency)	Filter (High Frequency)
I57US	238.680387365	2016-10-16 06:53:59	2016-10-16 06:54:08	1.5	5
I57US	351.992280197	2016-10-16 06:27:40	2016-10-16 06:27:48	1.5	5
I57US	298.610459666	2016-10-16 06:22:02	2016-10-16 06:22:24	1.5	5
I57US	331.091315884	2016-10-16 06:21:04	2016-10-16 06:21:13	1.5	5
I57US	158.198590514	2016-10-16 07:06:26	2016-10-16 07:06:35	1.5	5

Origins and detections produced by the underlying algorithm library. Analysts can compare their own observations of raw waveform data against the results produced by the algorithm library.



An analyst uses the interactive 3D globe (Cesium.js) to visualize and select infrasound stations that are of interest



Visualize the results of the processing. This includes waveforms, spectrograms, Fk plots, and more as development continues. Due to the subtleness of infrasound perturbations, the spectrograms are especially important for analysts in this mission.

Conclusion and Lessons Learned

Replacing legacy systems, keeping legacy data

- Hellhound needed to be able to interface with legacy data that was exposed in a pre-existing environment
- Subtle differences in the deploy system and our test environment yielded unexpected results
- **Get as much information about the deployment environment as possible and replicate for CI testing using automated provisioning (ex. Ansible, Terraform, Docker, etc.)**
- **Aim to discover problems locally rather than on-site**

Feature Request Cycle

- Recognize that the customer may be focused on the features of the old system, rather than the potential of the new system
- Delivering a feature often leads to the customer realizing what else is possible and revectoring. This can lead to development challenges if too locked into the initial design
- **Agile means both the developers AND the architecture and design of a system should be flexible**
- **Ensure time for regular re-evaluation of the system design to minimize technical debt accrued**

Value now, value later?

- Make no initial assumptions about what the target system is
- Iterate on the design early, repeatedly, to understand the target environment
- **Come to a compromise regarding technologies when you can, iterate and re-plan when you cannot**

...Solution: Hellhound!

U.S. DEPARTMENT OF ENERGY
NNSA National Nuclear Security Administration
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC,
a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.
SAND No. ____________

Sandia National Laboratories