IQEngine Project Update

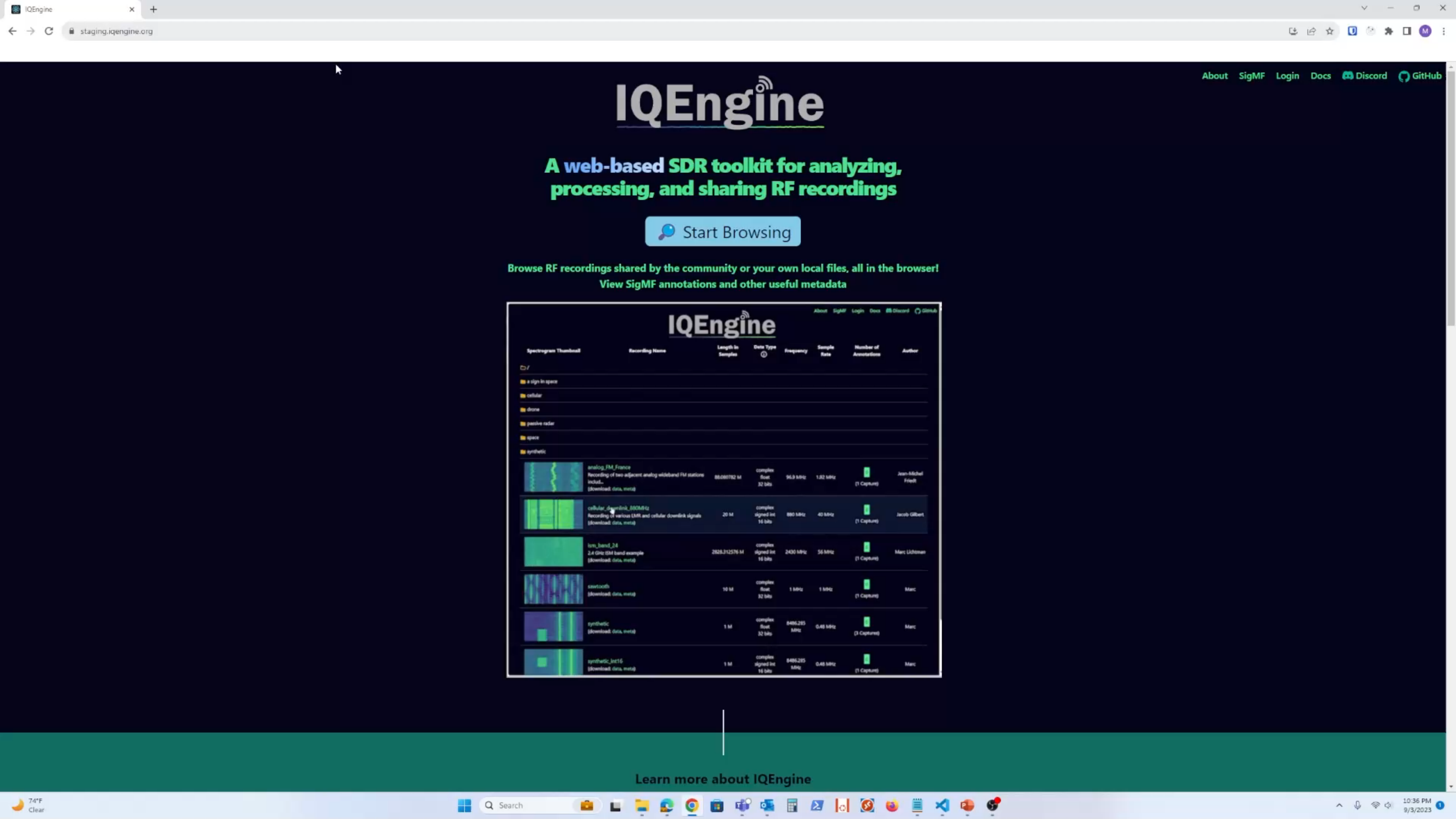
Marc Lichtman, Gabriel Nepomuceno, Denis Sutherland,
and all contributors of code, recordings, and plugins

IQEngine.org

What is it?

RF recording management
analysis
processing
sharing

... all in your browser



IQEngine

A web-based SDR toolkit for analyzing, processing, and sharing RF recordings

🔎 Start Browsing

Browse RF recordings shared by the community or your own local files, all in the browser!
View SigMF annotations and other useful metadata

IQEngine

Spectrogram Thumbnail	Recording Name	Length in Samples	Data Type ⓘ	Frequency	Sample Rate	Number of Annotations	Author
/							
a sign in space							
cellular							
drone							
passive radar							
space							
synthetic							
	analog_FM_France Recording of two adjacent analog wideband FM stations includ... (download: data, meta)	88.080782 M	complex float 32 bits	96.9 MHz	1.92 MHz	0 (1 Capture)	Jean-Michel Friedt
	cellular_downlink_880MHz Recording of various LMR and cellular downlink signals (download: data, meta)	20 M	complex signed int 16 bits	880 MHz	40 MHz	0 (1 Capture)	Jacob Gilbert
	ism_band_24 2.4 GHz ISM band example (download: data, meta)	2828.312576 M	complex signed int 16 bits	2430 MHz	56 MHz	0 (1 Capture)	Marc Lichtman
	sawtooth (download: data, meta)	10 M	complex float 32 bits	1 MHz	1 MHz	0 (1 Capture)	Marc
	synthetic (download: data, meta)	1 M	complex float 32 bits	8486.285 MHz	0.48 MHz	0 (3 Captures)	Marc
	synthetic_int16 (download: data, meta)	1 M	complex signed int 16 bits	8486.285 MHz	0.48 MHz	0 (1 Capture)	Marc

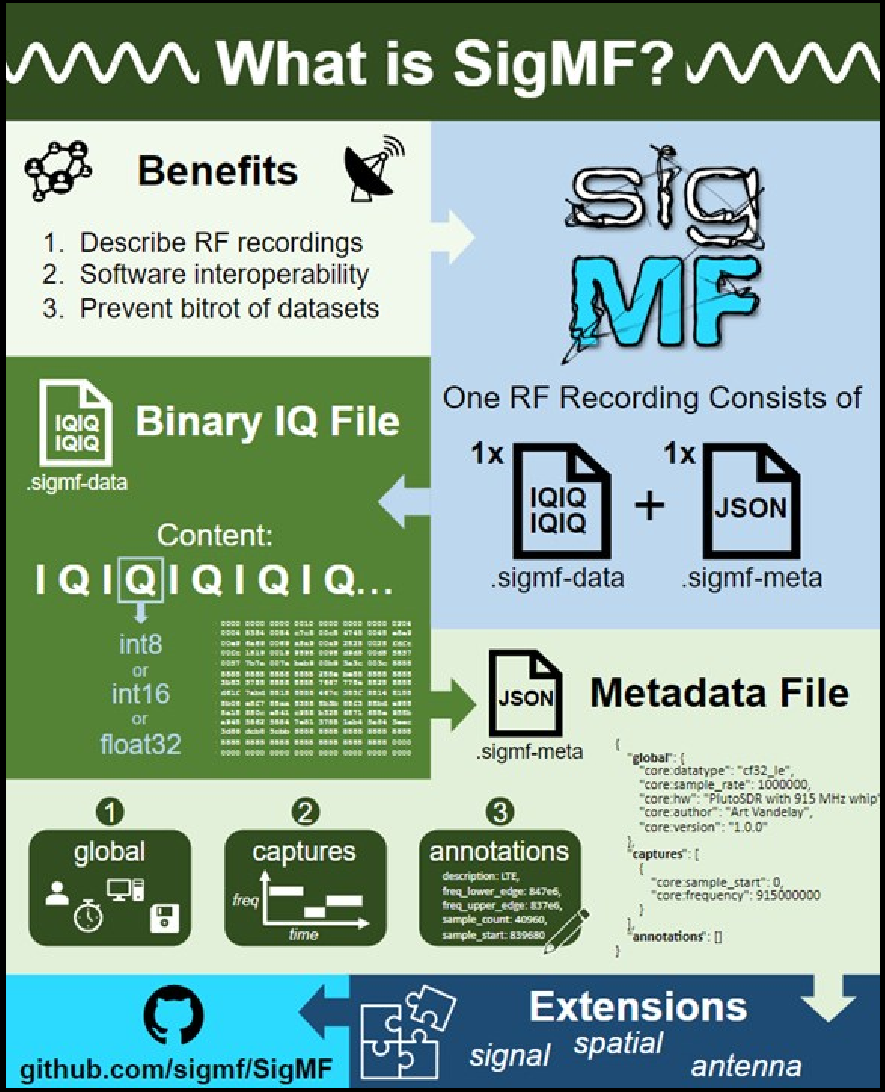
Learn more about IQEngine

Deployment Options

- The public instance at IQEngine.org is used to publicly share recordings and plugins
  - Recordings/plugins hosted either by IQEngine, GNU Radio, or 3rd party
- You can also open local files with IQEngine.org
- Only reason to run your own instance is to share recordings privately within your org, or for analyzing extremely sensitive data
- For all the above, use deployment via Docker images
- Developers of the frontend/backend will want to run a local instance from source

Built on Top of

- An open standard for saving RF recordings
- It’s a binary IQ file + a JSON file
- SigMF specifies how to write the JSON
- At a minimum, store
  1. Sample rate
  2. Center frequency
  3. Datatype of IQ
- Avoid data bitrot



What is SigMF?

Benefits

1. Describe RF recordings
2. Software interoperability
3. Prevent bitrot of datasets

SigMF

One RF Recording Consists of

1x .sigmf-data + 1x .sigmf-meta

Binary IQ File

.sigmf-data

Content:

I Q I Q I Q I Q I Q...

int8 or int16 or float32

Metadata File

.sigmf-meta

```
{
  "global": {
    "core:datatype": "cf32_le",
    "core:sample_rate": 1000000,
    "core:hw": "PlutoSDR with 915 MHz whip",
    "core:author": "Art Vandelay",
    "core:version": "1.0.0"
  },
  "captures": [
    {
      "core:sample_start": 0,
      "core:frequency": 915000000
    }
  ],
  "annotations": []
}
```

1. global
2. captures
3. annotations

description: LTE,
freq_lower_edge: 847e6,
freq_upper_edge: 837e6,
sample_count: 40960,
sample_start: 839680

Extensions

signal spatial antenna

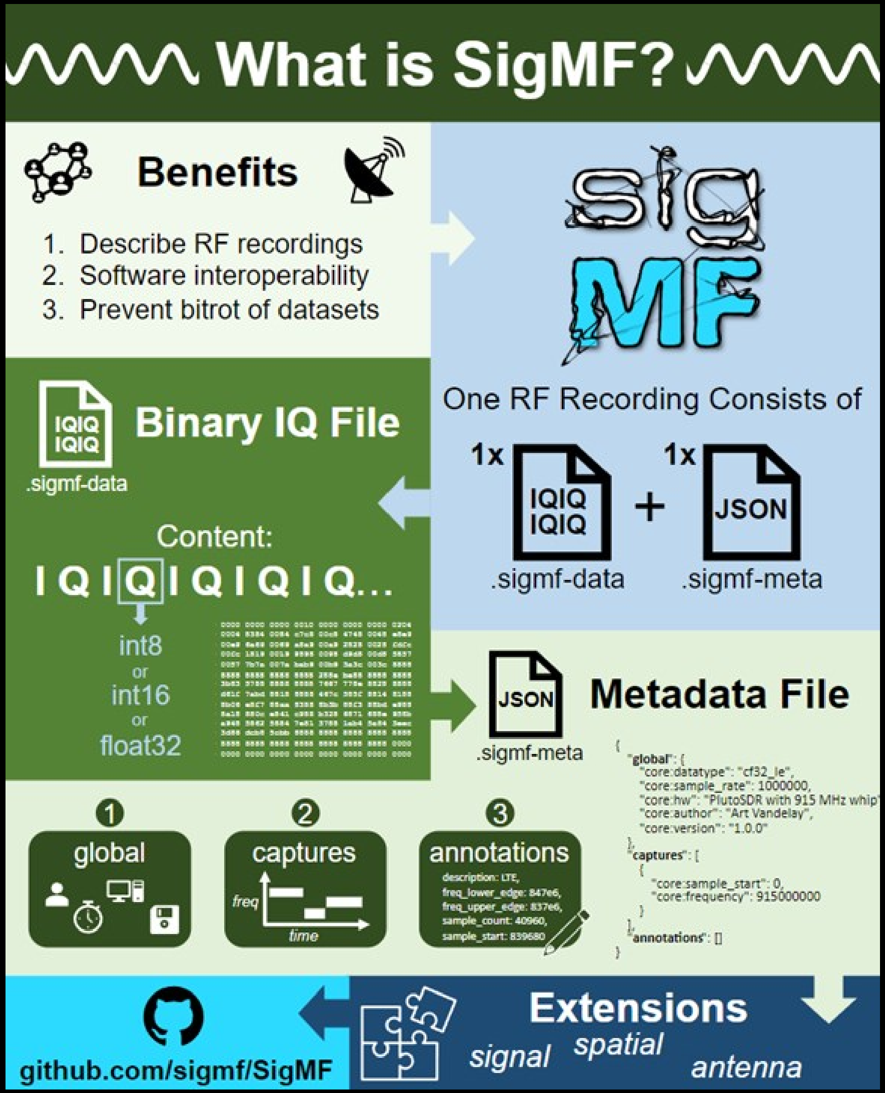
github.com/sigmf/SigMF

Built on Top of

- An open standard for saving RF recordings
- It's a binary IQ file + a JSON file
- SigMF specifies how to write the JSON
- At a minimum, store
    1. Sample rate
    2. Center frequency
    3. Datatype of IQ
- Avoid data bitrot

Learn more about SigMF

+ SigMF Workshop



1. Describe RF recordings
2. Software interoperability
3. Prevent bitrot of datasets

description: LTE,
freq_lower_edge: 847e6,
freq_upper_edge: 837e6,
sample_count: 40960,
sample_start: 839680

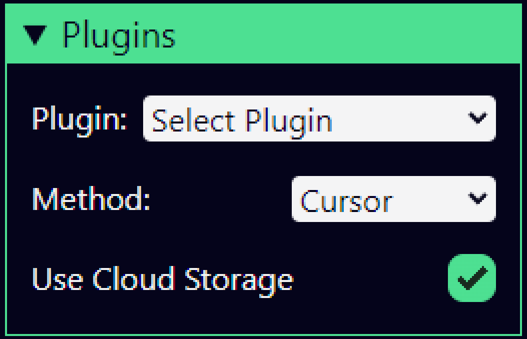
```
{
  "global": {
    "core:datatype": "cf32_le",
    "core:sample_rate": 1000000,
    "core:hw": "PlutoSDR with 915 MHz whip"
    "core:author": "Art Vandelay",
    "core:version": "1.0.0"
  },
  "captures": [
    {
      "core:sample_start": 0,
      "core:frequency": 915000000
    }
  ],
  "annotations": []
}
```

Target Users

- Just like GNU Radio, IQEngine is meant for a variety of users:
  - Students
  - Hobbyists - hams, CTFs
  - Orgs – research labs, companies, gov
- For research or alongside production systems

Plugins

- RF signal processing on the backend, triggered from browser
- The plugins backend server is separate, and there can be multiple
  - In theory, plugins don't have to be open-source, if the 3rd party runs the server
- REST-based API defined in our OpenAPI spec
- Allows for plugins to be written in any language
- We have examples/templates for Python and GNU Radio

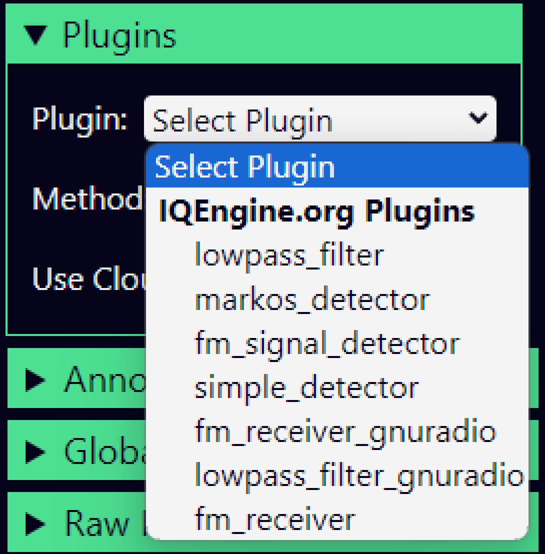


▼ Plugins

Plugin: Select Plugin

Method: Cursor

Use Cloud Storage ☑



▼ Plugins

Plugin: Select Plugin

Method

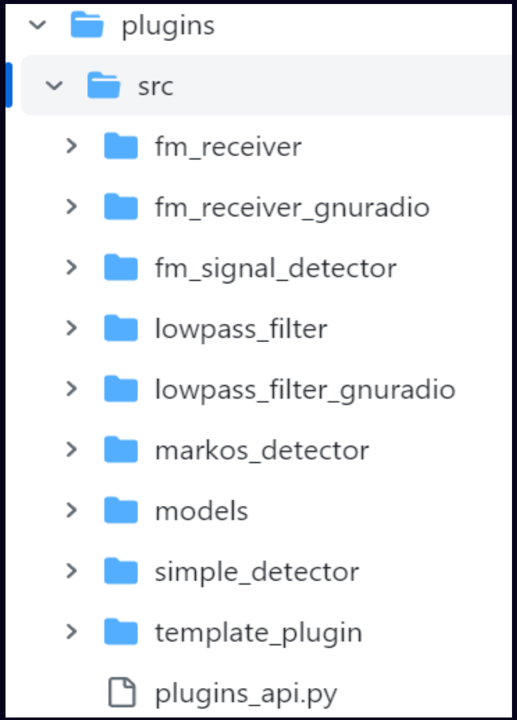
Use Clou

- Select Plugin
- **IQEngine.org Plugins**
  - lowpass_filter
  - markos_detector
  - fm_signal_detector
  - simple_detector
  - fm_receiver_gnuradio
  - lowpass_filter_gnuradio
  - fm_receiver

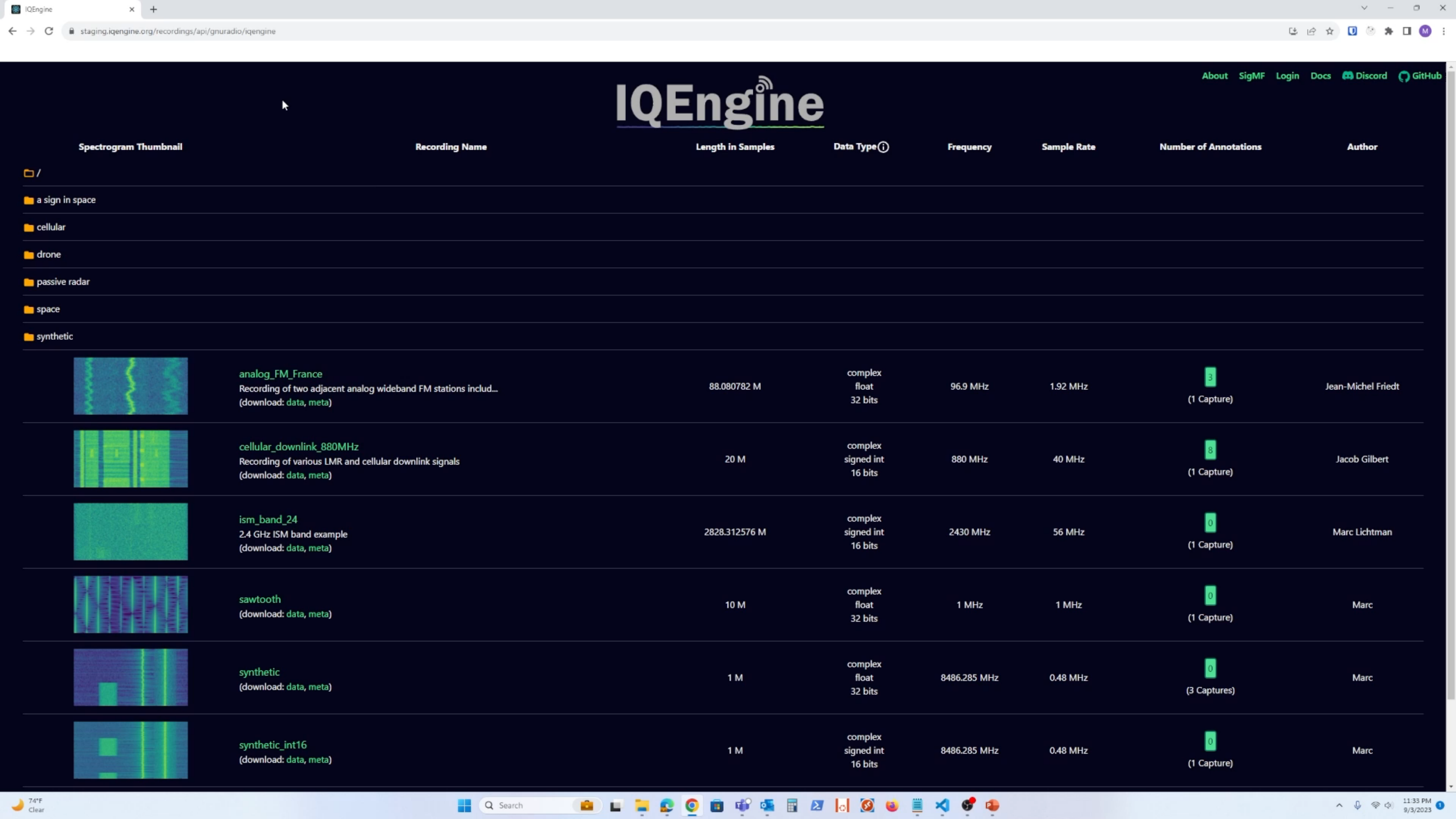
▶ Anno

▶ Glob

▶ Raw I



```
plugins
  src
    fm_receiver
    fm_receiver_gnuradio
    fm_signal_detector
    lowpass_filter
    lowpass_filter_gnuradio
    markos_detector
    models
    simple_detector
    template_plugin
    plugins_api.py
```

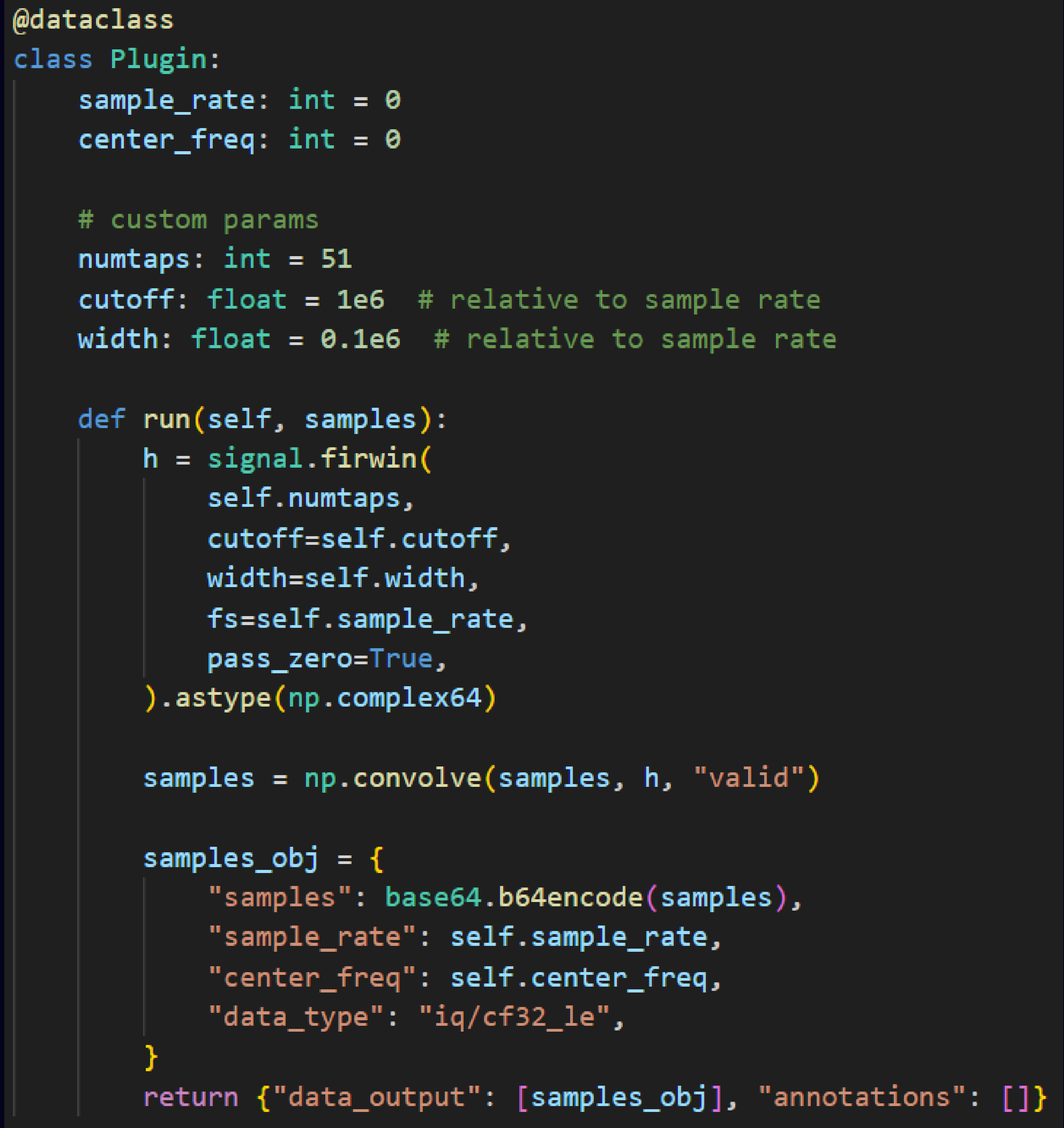


IQEngine

Spectrogram Thumbnail	Recording Name	Length in Samples	Data Type	Frequency	Sample Rate	Number of Annotations	Author
/							
a sign in space							
cellular							
drone							
passive radar							
space							
synthetic							
	analog_FM_France Recording of two adjacent analog wideband FM stations includ... (download: data, meta)	88.080782 M	complex float 32 bits	96.9 MHz	1.92 MHz	3 (1 Capture)	Jean-Michel Friedt
	cellular_downlink_880MHz Recording of various LMR and cellular downlink signals (download: data, meta)	20 M	complex signed int 16 bits	880 MHz	40 MHz	8 (1 Capture)	Jacob Gilbert
	ism_band_24 2.4 GHz ISM band example (download: data, meta)	2828.312576 M	complex signed int 16 bits	2430 MHz	56 MHz	0 (1 Capture)	Marc Lichtman
	sawtooth (download: data, meta)	10 M	complex float 32 bits	1 MHz	1 MHz	0 (1 Capture)	Marc
	synthetic (download: data, meta)	1 M	complex float 32 bits	8486.285 MHz	0.48 MHz	0 (3 Captures)	Marc
	synthetic_int16 (download: data, meta)	1 M	complex signed int 16 bits	8486.285 MHz	0.48 MHz	0 (1 Capture)	Marc

Python Plugin Example

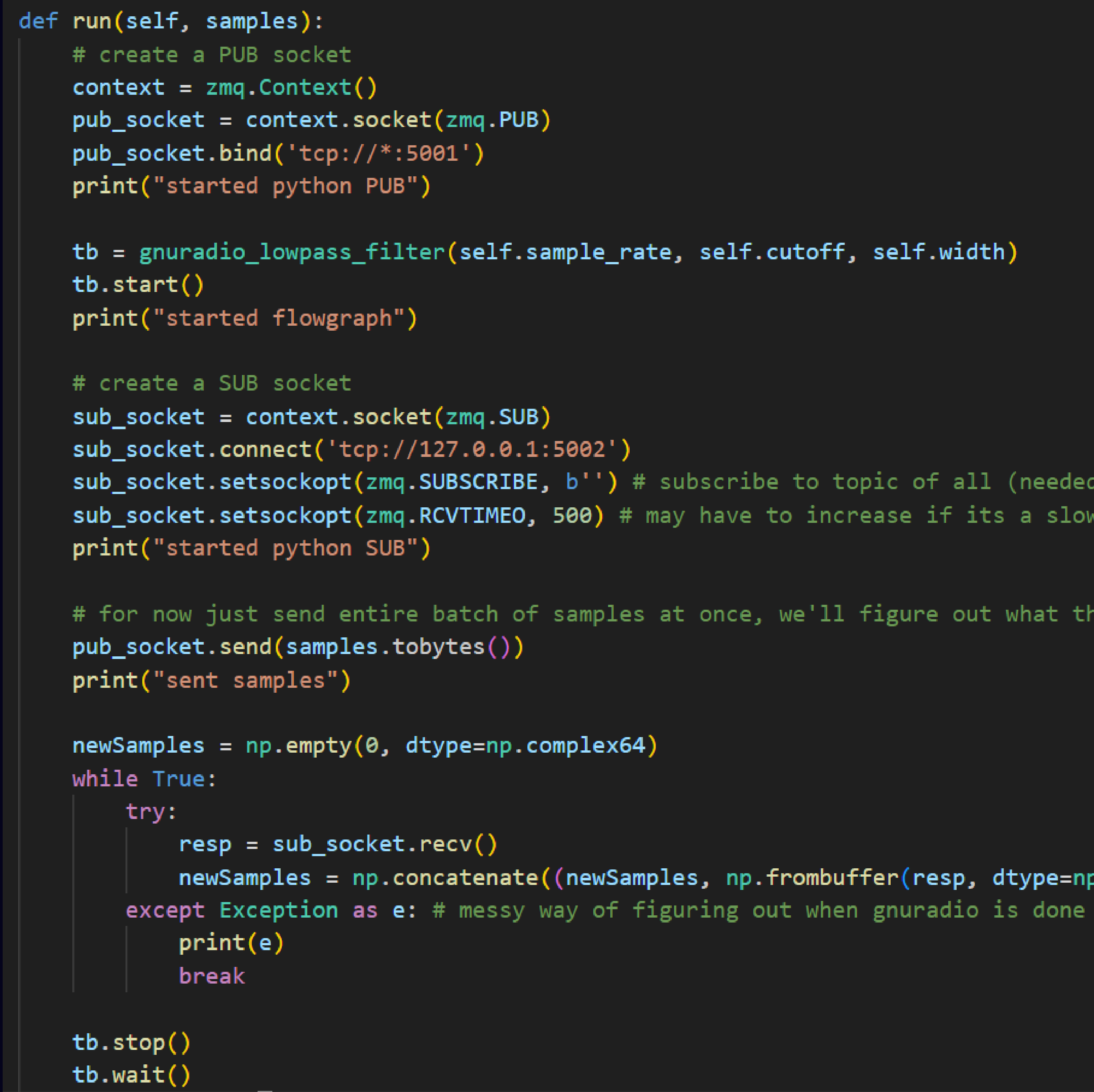
- Must specify custom params
- Must have a run() function that takes in sample
- OpenAPI spec defines interface



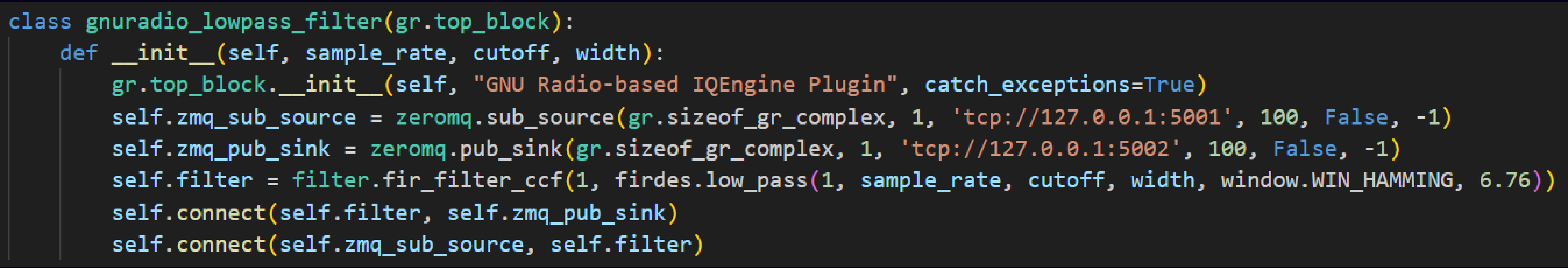
```python
@dataclass
class Plugin:
    sample_rate: int = 0
    center_freq: int = 0

    # custom params
    numtaps: int = 51
    cutoff: float = 1e6  # relative to sample rate
    width: float = 0.1e6  # relative to sample rate

    def run(self, samples):
        h = signal.firwin(
            self.numtaps,
            cutoff=self.cutoff,
            width=self.width,
            fs=self.sample_rate,
            pass_zero=True,
        ).astype(np.complex64)

        samples = np.convolve(samples, h, "valid")

        samples_obj = {
            "samples": base64.b64encode(samples),
            "sample_rate": self.sample_rate,
            "center_freq": self.center_freq,
            "data_type": "iq/cf32_le",
        }
        return {"data_output": [samples_obj], "annotations": []}
```

GNU Radio Plugin Example

- Little hacky but works for now with existing blocks
- Define Python flowgraph using zeromq’s sub_source & pub_sink (next slide)
- run() function has its own pub/sub for feeding in samples and getting the output
- Come to the workshop for a hands-on tutorial



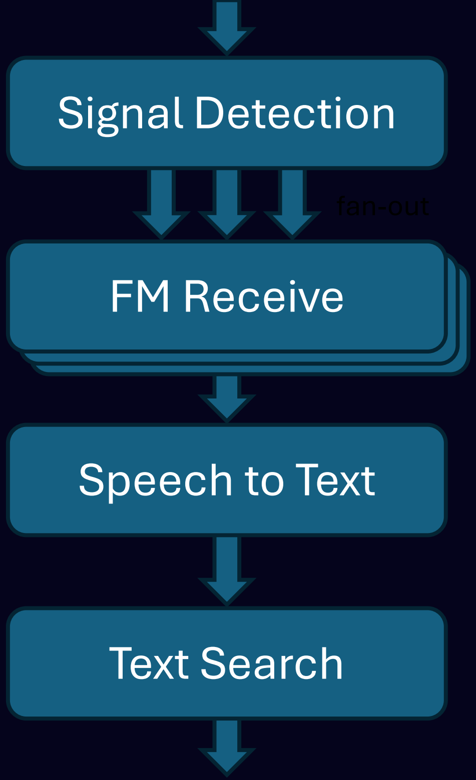
```python
def run(self, samples):
    # create a PUB socket
    context = zmq.Context()
    pub_socket = context.socket(zmq.PUB)
    pub_socket.bind('tcp://*:5001')
    print("started python PUB")

    tb = gnuradio_lowpass_filter(self.sample_rate, self.cutoff, self.width)
    tb.start()
    print("started flowgraph")

    # create a SUB socket
    sub_socket = context.socket(zmq.SUB)
    sub_socket.connect('tcp://127.0.0.1:5002')
    sub_socket.setsockopt(zmq.SUBSCRIBE, b'') # subscribe to topic of all (neede
    sub_socket.setsockopt(zmq.RCVTIMEO, 500) # may have to increase if its a slow
    print("started python SUB")

    # for now just send entire batch of samples at once, we'll figure out what th
    pub_socket.send(samples.tobytes())
    print("sent samples")

    newSamples = np.empty(0, dtype=np.complex64)
    while True:
        try:
            resp = sub_socket.recv()
            newSamples = np.concatenate((newSamples, np.frombuffer(resp, dtype=np
        except Exception as e: # messy way of figuring out when gnuradio is done
            print(e)
            break

    tb.stop()
    tb.wait()
```



```
class gnuradio_lowpass_filter(gr.top_block):
    def __init__(self, sample_rate, cutoff, width):
        gr.top_block.__init__(self, "GNU Radio-based IQEngine Plugin", catch_exceptions=True)
        self.zmq_sub_source = zeromq.sub_source(gr.sizeof_gr_complex, 1, 'tcp://127.0.0.1:5001', 100, False, -1)
        self.zmq_pub_sink = zeromq.pub_sink(gr.sizeof_gr_complex, 1, 'tcp://127.0.0.1:5002', 100, False, -1)
        self.filter = filter.fir_filter_ccf(1, firdes.low_pass(1, sample_rate, cutoff, width, window.WIN_HAMMING, 6.76))
        self.connect(self.filter, self.zmq_pub_sink)
        self.connect(self.zmq_sub_source, self.filter)
```

Pipelines

- IQEngine's browser interface lets you evaluate and tweak params of plugins on a variety of RF recordings
- But what if you know you want to run a series of plugins on all recordings being captured by a receiver?
- The same Plugins API can be used to call plugins in a chain
- We would like to use an existing format and design software for creating the chain of plugins, ideally with a web-based interface (web GRC?)
- The pipeline would then run on a Kubernetes cluster, listening for new recordings to appear in storage or provided via REST
- Obviously more tailored towards orgs than individuals



Signal Detection

fan-out

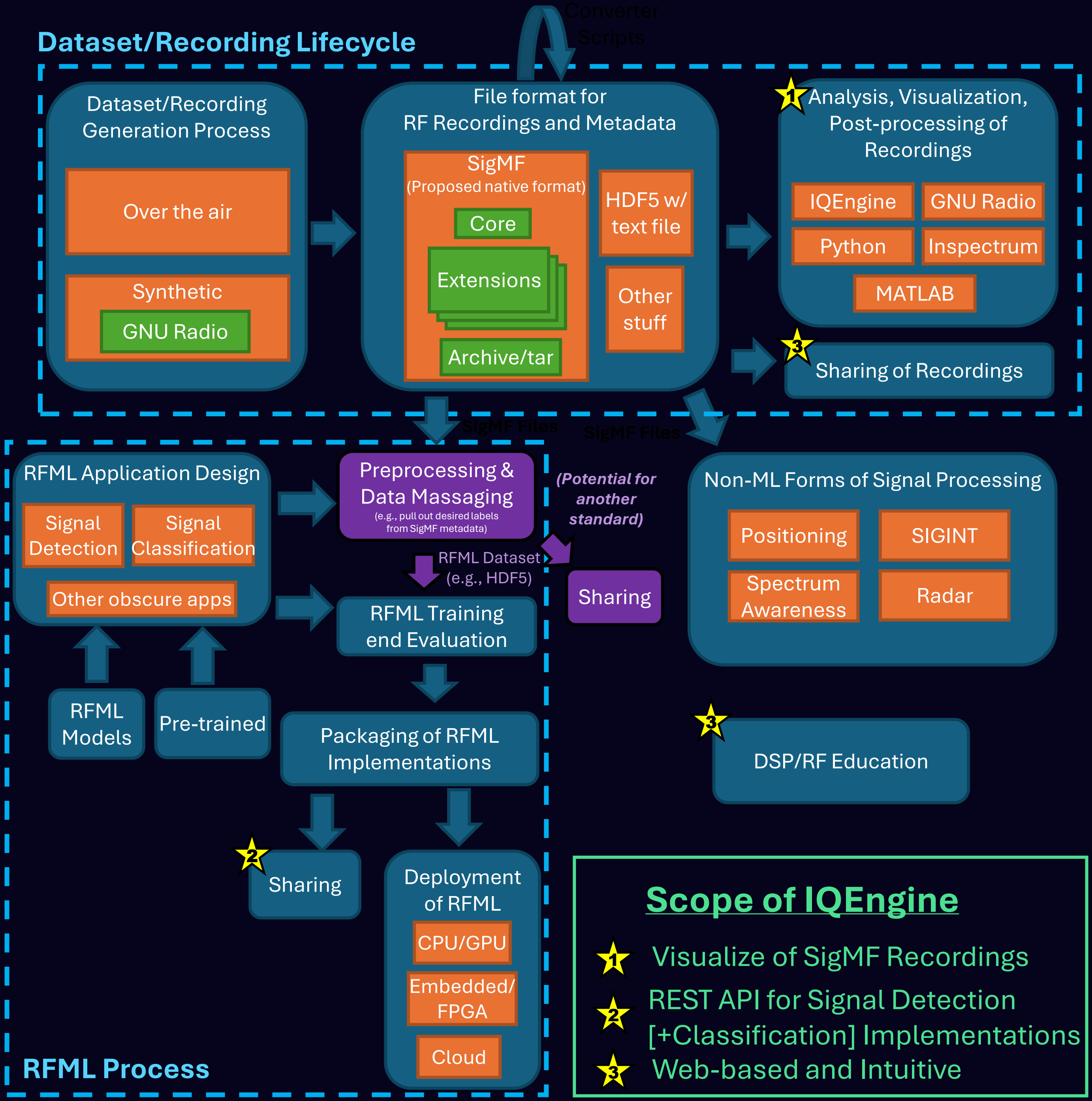
FM Receive

Speech to Text

Text Search

For You RFML Folks

- Use it at the beginning & end of the RFML workflow
- Manage IQ recordings used to form your dataset
- Share your detection and classification implementation (inference) with the world, as plugins



Dataset/Recording Lifecycle

Converter Scripts

Dataset/Recording Generation Process

Over the air

Synthetic

GNU Radio

File format for RF Recordings and Metadata

SigMF (Proposed native format)

Core

Extensions

Archive/tar

HDF5 w/ text file

Other stuff

1 Analysis, Visualization, Post-processing of Recordings

IQEngine

GNU Radio

Python

Inspectrum

MATLAB

3 Sharing of Recordings

SigMF Files

SigMF Files

RFML Application Design

Signal Detection

Signal Classification

Other obscure apps

RFML Models

Pre-trained

Preprocessing & Data Massaging (e.g., pull out desired labels from SigMF metadata)

RFML Dataset (e.g., HDF5)

(Potential for another standard)

Sharing

RFML Training end Evaluation

Packaging of RFML Implementations

2 Sharing

Deployment of RFML

CPU/GPU

Embedded/ FPGA

Cloud

RFML Process

Non-ML Forms of Signal Processing

Positioning

SIGINT

Spectrum Awareness

Radar

3 DSP/RF Education

Scope of IQEngine

1 Visualize of SigMF Recordings

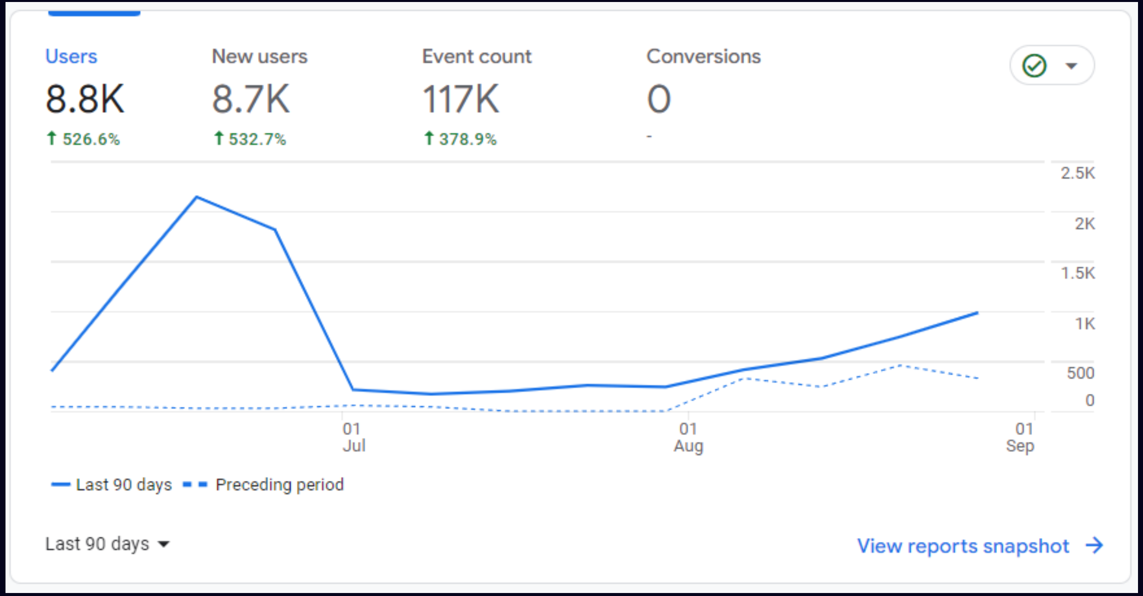
2 REST API for Signal Detection [+Classification] Implementations

3 Web-based and Intuitive

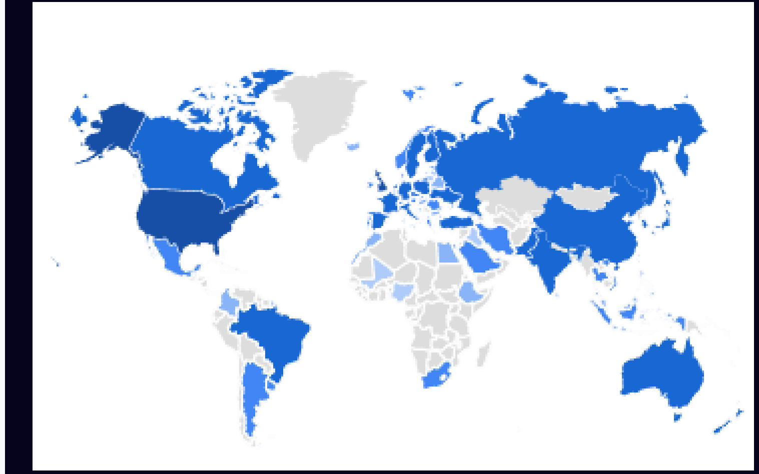
Link to Discord

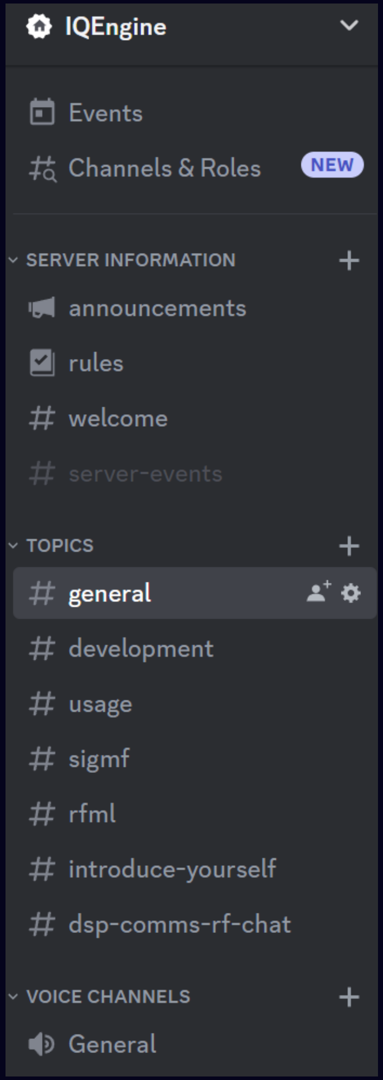
Community

- Discord
  - Thanks to Jumbotron for moderating and helping configure
- GitHub Issues/PRs
  - Issues can also be used for feature requests or ideas
- Google Analytics shows 8.8k unique visitors over last 90 days



Users	New users	Event count	Conversions
8.8K	8.7K	117K	0
↑ 526.6%	↑ 532.7%	↑ 378.9%	-





IQEngine

Events

Channels & Roles NEW

SERVER INFORMATION

- announcements
- rules
- welcome
- server-events

TOPICS

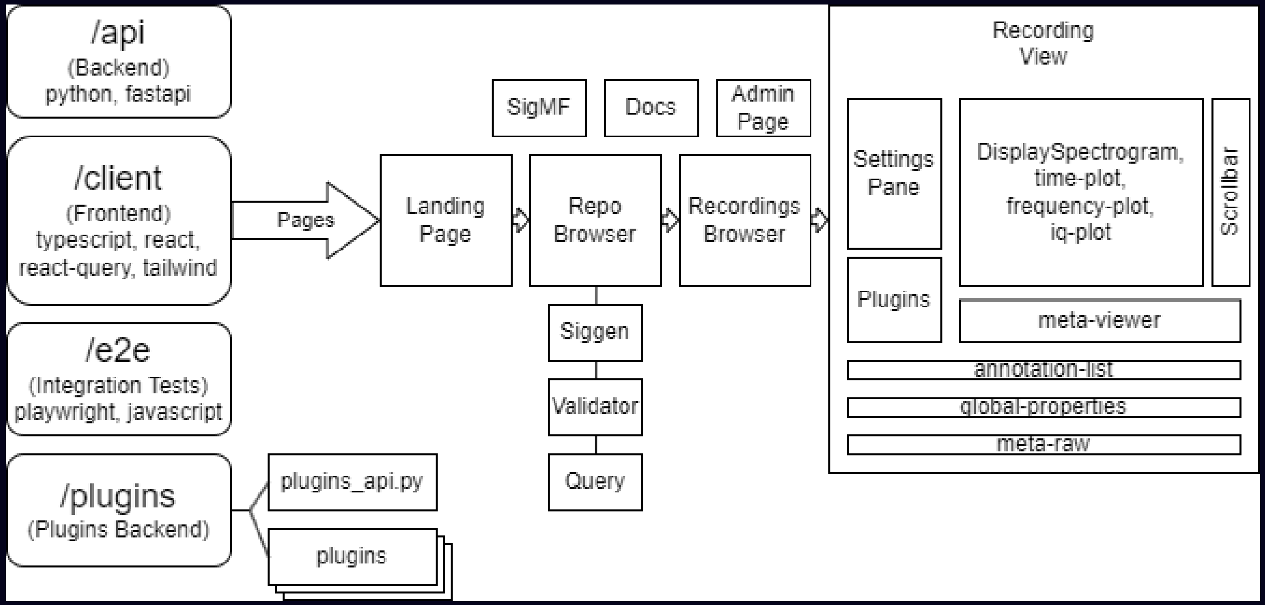
- general
- development
- usage
- sigmf
- rfml
- introduce-yourself
- dsp-comms-rf-chat

VOICE CHANNELS

- General

Code Organization

- Mono-repo
- Frontend and backend are built into the same Docker image
- Plugins image includes GNU Radio and, in the future, any other open-source software that will be wrapped into plugins
- Docs live as .mdx files and render into https://iqengine.org/docs
- Frontend uses React, backend is in Python/fastapi, tailwind for css/styling



CI/CD

- Deployment through Docker images
- https://staging.iqengine.org is always running the latest “main” branch
  - No dev branch, but major releases are periodically tagged
- All PRs must pass
  1. Frontend unit tests (vitest)
  2. Backend unit tests (pytest)
  3. Integration tests (playwright)
  4. CodeQL
  5. GitHub’s dependency-review
  6. Mega-Linter (optional)
- Nightly integration tests of staging and prod for good measure
- Weekly Dependabot for version bumping
- OpenSSF Scorecard analysis on pushes to main

Upcoming Features

- Near-term
  - Using a server’s local storage or NAS to host recordings
  - Wrapping SatDump into a plugin
  - Web Assembly-based client-side FFTs
  - UX improvements (a big thanks to Bernard)
  - Indicator that client is waiting on the plugin to finish
- Long-term
  - Better time/freq/IQ interactive plots, e.g., ability to display alongside spectrogram
  - Progress bar for plugins
  - Plugins pipeline designer and cluster-based executor
  - Include more SigMF-specific functionality
  - Cyclostationary processing in place of the FFT

Ways to Contribute

- Contribute:
  1. RF recordings
  2. Open-source signal processing implementations via the plugins system
  3. Python transmitter code via the siggen tool (education-oriented)
- We can also use help curating recordings
- Code contributors are also nice!
- We are looking for universities and companies/orgs/labs to engage with
- Reach out on Discord or email iqengine@vt.edu

One Last Thing...

(If laptop has internet access)

Questions?

Show your support by starring the GitHub Repo! ⭐

https://github.com/iqengine/iqengine