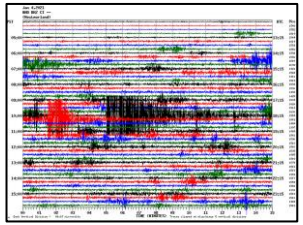
But it works for me! How To Share Research Code

Aparna Bhaskaran, Prabha Acharya and Ryan Tam
Southern California Seismic Network
Brown Bag Seminar
9-3-25

Postprocessing SoA: How To Share Research Code

Ryan Tam

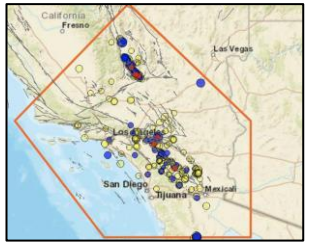
Earthquake Monitoring in a Regional Seismic Network



Continuous seismic data

?

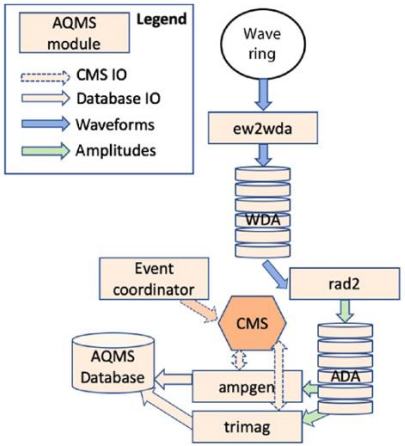
Earthquake catalog



Source of images: Clara Yoon, USGS

Earthquake Monitoring: Realtime

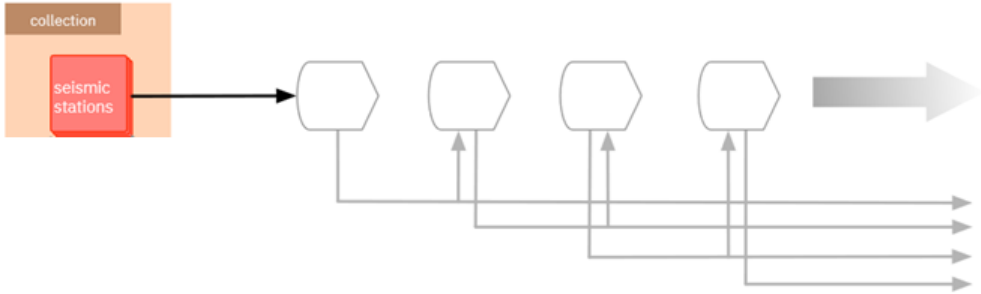
- Processing seismic data as it is being recorded.
- Rapidly detect and characterize earthquakes, and provide immediate information
- Data has not yet been labeled
- Backlog not tolerable - any signals missed will not be processed



J. Renate Hartog, Paul A. Friberg, Victor C. Kress, Paul Bodin, Rayomand Bhadha; Open-Source ANSS Quake Monitoring System Software. *Seismological Research Letters* 2019;; 91 (2A): 677–686.
doi: https://doi.org/10.1785/0220190219

Earthquake Monitoring: Post-processing

- Events have already been identified by AQMS
  - Event has an ID and is now labeled
- Can be anytime after the event, even milliseconds after an event
- Refine the results using machine learning.
- Backlog can form, but not desirable.



Our motivation for moving to a Service-Oriented Architecture (SoA)

- **Development of new machine learning algorithms**
  - Hard to fit algorithms into existing pipeline without additional development
- **Make processes more scalable**
  - Have the freedom to host services either on-prem or in the cloud

- **So why use microservices**?
  - Think of it as a URL, like your gmail service
  - Each service is fine-grained, does one thing well
  - Can be re-used in different applications
  - Loosely coupled: just need to know how to communicate with each other
  - Interoperable: services can run on different platforms/written in different languages
  - Basis of cloud and modern web apps

Distributing Processes, Not Just Data

- **Our services communicate with a URL (endpoint)**
  - Callable by any program - a website, your phone, a Java application can all hit it as long as they are part of the same network
  - As the user, you only need to know how to call the URL, not how it is implemented.
  - As developers, we can secure the URL with authorization and authentication as well.
- We utilize containerization
  - Allows the API endpoint to work on your machine
  - Sets up a distributed architecture of multiple APIs that you can access
  - As developers, containerization naturally fits into our CI/CD pipelines
- **Use our API, or bring your own algorithm. We welcome your contribution or software development in any form using our services.**

Project Goals

Key Services Offered

- Retrieving windows from events or triggers
- Retrieving waveforms from windows
- Picker
- Pick Filter
- Pick Associator

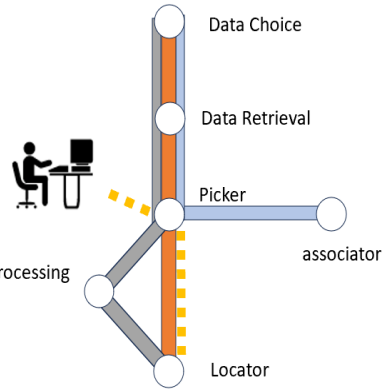
Expectations

- Utilize a service-oriented architecture
- Utilize JSON or NEIC-based format
- Resilient and fast. Each service should be fast and resilient enough to minimize backlog during periods of large earthquakes/swarms.
- Each service's outputs can be processed into its downstream service's inputs.
- Each service should be adaptable and easy to use by users.

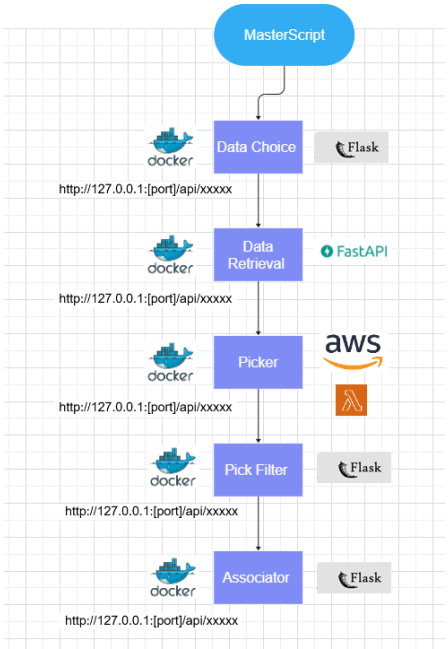
SoA: Services Offered

- Data Choice: Given an event/trigger ID, retrieve windows
- Data Retrieval: Retrieve waveforms from windows
- Picker: Find picks from waveforms (Phasenet currently)
- Pick Filter: Filter picks:
  - Outside a window of their expected arrival time
  - From stations beyond a given distance from the event
  - Based on phase score
  - Based on incorrect channel/phase matches
- Associator: Find events from filtered picks (GaMMA)
- Locator: TBD
- More new services in the future, will accommodate

In the future, with seismic processing available as services, users can create their own workflows.



SoA: Our Services



Machine Learning-Informed Catalog

ID	EXT_ID	VER	OWHO	ST	DATETIME	TF	MAG	MTYP	MWHO	HF	LAT	LON	Z
39515370	628333	0	---	H	2019-07-06 03:16:30.847	0	4.07	Mw	---	0	35.592	-117.344	21.00
39515354	627980	0	---	H	2019-07-06 03:16:32.496	0	4.58	Mw	---	0	35.716	-117.564	16.73
39515362	627981	0	---	H	2019-07-06 03:17:13.801	0	4.05	Mw	---	0	35.733	-117.572	8.95
39515378	628192	0	---	H	2019-07-06 03:18:38.074	0	2.67	Mw	---	0	35.744	-117.564	11.71
39515394	628377	0	---	H	2019-07-06 03:19:51.645	0	5.29	Mw	---	0	35.623	-117.450	21.00
39515410	628669	0	---	H	2019-07-06 03:22:47.127	0	4.99	Mw	---	0	35.872	-117.748	21.00
39515426	629117	0	---	H	2019-07-06 03:23:51.383	0	4.91	Mw	---	0	35.784	-117.656	0.35
39515434	629175	0	---	H	2019-07-06 03:24:05.007	0	5.14	Mw	---	0	35.020	-117.423	21.00
39515450	629233	0	---	H	2019-07-06 03:25:27.977	0	4.74	Mw	---	0	35.857	-117.686	11.01
39515474	629528	0	---	H	2019-07-06 03:27:16.774	0	4.60	Mw	---	0	35.294	-117.851	21.00

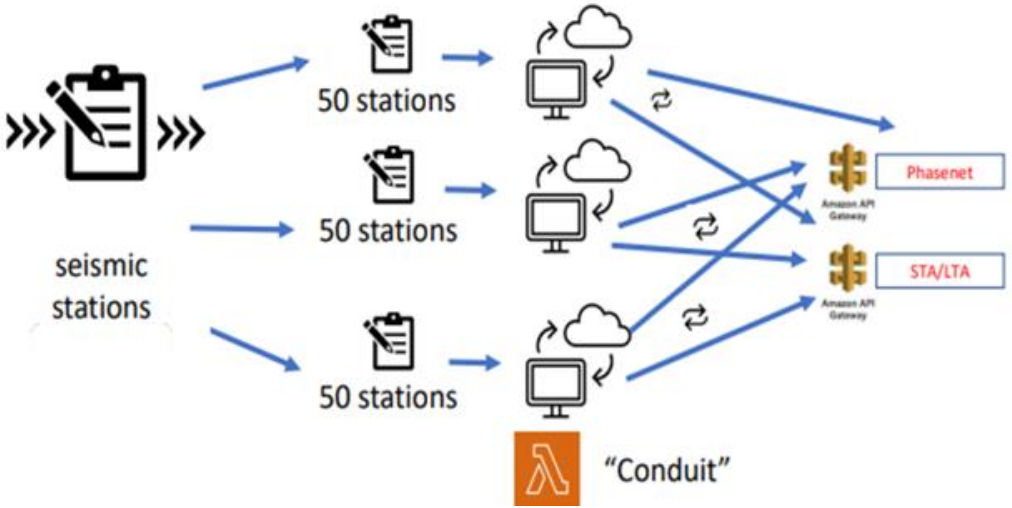
Waveforms

SoA: Services used in Operational Postprocessing

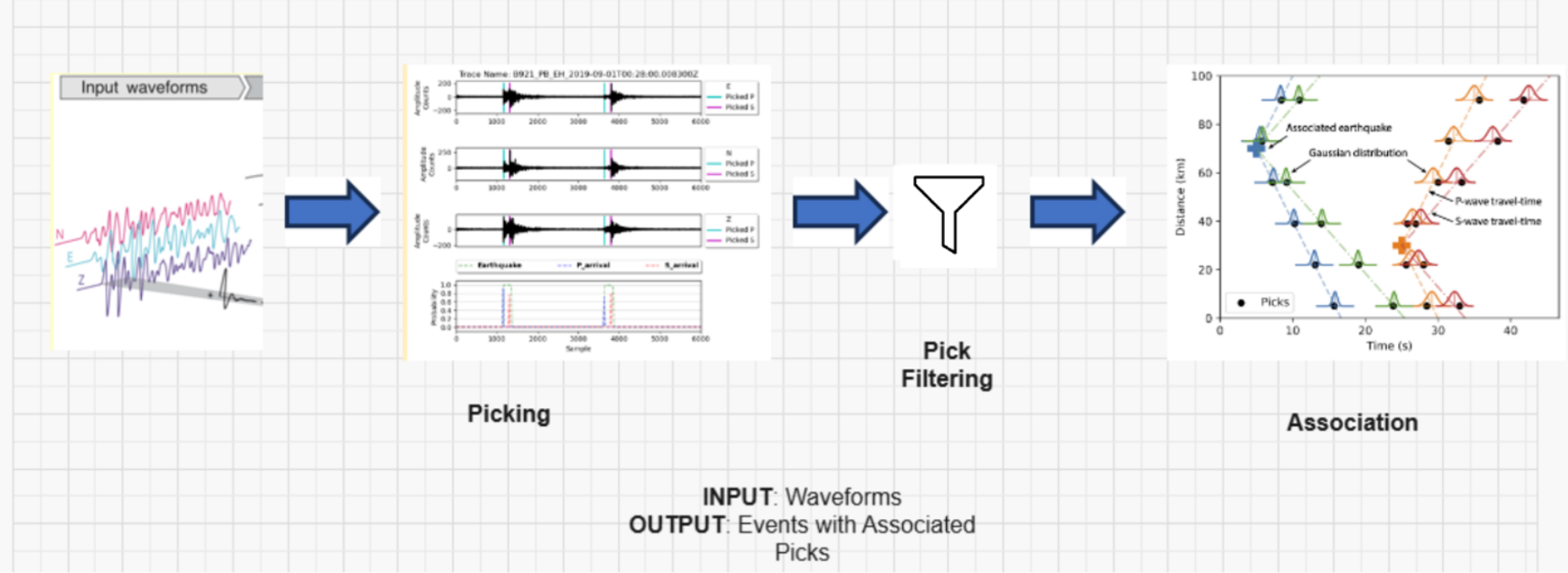
- **Our services are used as part of the SCSN's near real-time operational post-processing pipelines:**
  - HypoPN - event post-processing
    - Utilizes machine learning picker service to refine an event's real-time origin
  - STProc - automatic post-processing pipeline for sub-network triggers
    - Subnet triggers are collections of nearby phase picks that the real-time system could not associate into an event
    - Utilizes machine learning picker and machine learning associator services to identify new events

The Phasenet Picker

- **Leverages AWS**
  - Cloud scalable, can run many waveforms in parallel
  - Decreases picking time significantly compared to prior efforts, such as using a single server
  - Fixed quality issues
    - Kept picks found past 2 seconds from the start of the waveform
  - Endpointed and containerized
- Research code implemented as a service
  - Yours could be next!
- Phasenet easier to incorporate into the cloud because:
  - Well tested
  - Well documented



Demo Scheme: How To Use the Picker, Pick Filter and Associator By Accessing Docker-Staged Endpoints



- See us at the end of the presentation if interested in the demo.

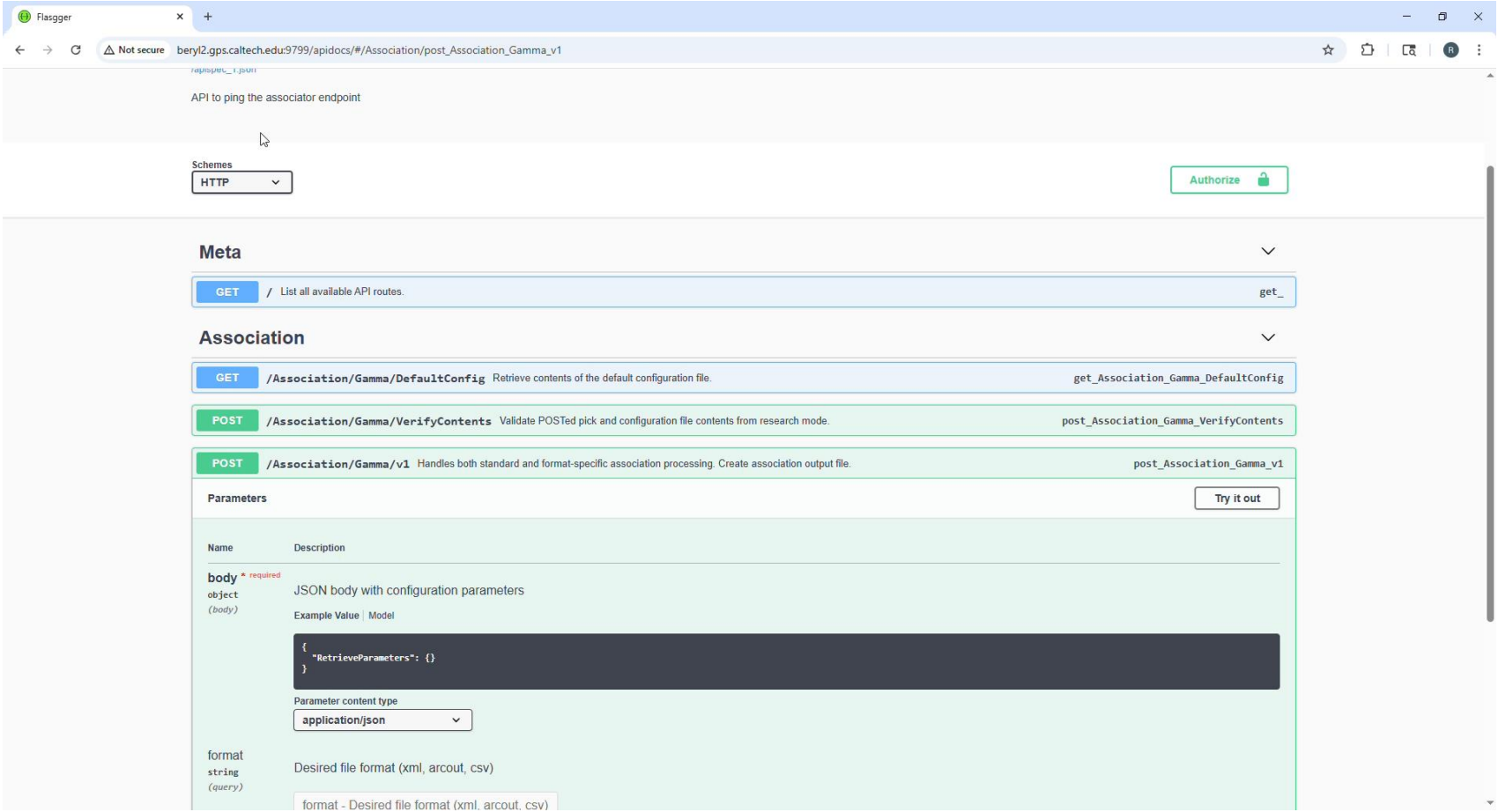
Formats

- Our inputs and outputs adhere to JSON structure
- We are working with NEIC to use common formats.
- The data is the interface
  - As long you adhere to our specified data structure, you can send it to the URL and it will work.


```
[
  {
    "filename": "AZ.CRY.HHN",
    "network": "AZ",
    "station": "CRY",
    "location": "",
    "channel": "HHN",
    "starttime": "2024-02-01T12:38:17.508300Z",
    "endtime": "2024-02-01T12:39:17.108300Z",
    "sampling_rate": 100.0,
    "delta": 0.01,
    "data": [
      972,
      971,
```


Example waveform file format

Demo: Associator, Interactive API documentation and exploration


Flasgger
Not secure
beryl2.gps.caltech.edu:9799/apidocs/#/Association/post_Association_Gamma_v1
API to ping the associator endpoint
Schemes
HTTP
Authorize
Meta
GET / List all available API routes. get_
Association
GET /Association/Gamma/DefaultConfig Retrieve contents of the default configuration file. get_Association_Gamma_DefaultConfig
POST /Association/Gamma/VerifyContents Validate POSTed pick and configuration file contents from research mode. post_Association_Gamma_VerifyContents
POST /Association/Gamma/v1 Handles both standard and format-specific association processing. Create association output file. post_Association_Gamma_v1
Parameters
Try it out
Name
Description
body * required
object
(body)
JSON body with configuration parameters
Example Value | Model
{
"RetrieveParameters": {}
}
Parameter content type
application/json
format
string
(query)
Desired file format (xml, arcout, csv)
format - Desired file format (xml, arcout, csv)

CI/CD

- CI/CD has allowed us to update features within each of our services and provided observability to members of our team

```yaml
image: python

stages:
    - build
    - test

stage1:
    stage: build
    script:
        - python --version
        - python -m pip install -U pip
        - python -m pip install -U numpy pandas streamlit
        - python main.py
        - echo 'successful'

stage2:
    stage: test
    script:
        - python --version
        - python -m pip install -U pip
        - pip install pytest
        - pytest main.py
        - echo 'successful'
```

```
Running validation scripts in dataretrievalapi container, with dataretrieval outputs
.......
----------------------------------------------------------------------
Ran 7 tests in 9.878s
OK
Running after_script
Running after script...
$ echo "Tearing down the Docker Compose service..."
Tearing down the Docker Compose service...
$ docker-compose down || true
 Container dataretrievalapi  Stopping
 Container dataretrievalapi  Stopped
 Container dataretrievalapi  Removing
 Container dataretrievalapi  Removed
Cleaning up project directory and file based variables
Job succeeded
```

Project Goals

Key Services Offered

- ✓ Retrieving windows from events or triggers
- ✓ Retrieving waveforms from windows
- ✓ Picker
- ✓ Pick Filter
- ✓ Pick Associator

Expectations

- ✓ Utilize a service-oriented architecture
- ✓ Utilize JSON or NEIC-based format
- ✓ Resilient and fast. Each service should be fast and resilient enough to minimize backlog during periods of large earthquakes/swarms.
- ✓ Each service's outputs can be processed into its downstream service's inputs.
- ✓ Each service should be adaptable and easy to use by users.

How Your Work Can Plug In

- **We plan on open sourcing our associator service soon.**
  - **Utilize our service:**
    - Just spin up our Dockerized container and start calling the API endpoint from any app or language.
    - Fewer libraries to install, fewer setup headaches
  - **Enhance our service:**
    - **If you have any machine learning pickers, associators, or other algorithms that you want deployed as a service, let us know. We can add them as different services within our endpoint.**
  - Results from the SoA to be published in a paper by Gabrielle Tepp
  - See Tepp et. al "***Improvements from incorporating machine learning algorithms into near real-time operational post-processing***"

- Our microservices serve two near realtime post-processing pipelines:
  - Event post-processing
  - Automatic post-processing pipeline (ST-Proc) for sub-network triggers
  - **Both have reduced analyst workload and streamlined event processing**

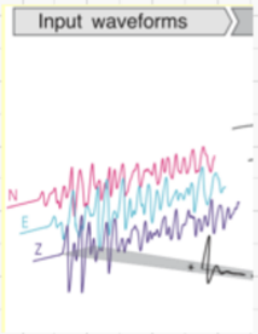
How Your Work Can Plug In

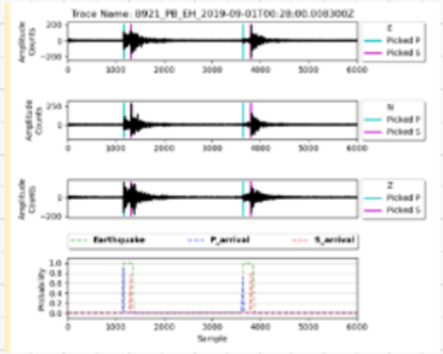
- **We hope to facilitate others to develop against our APIs, providing an environment for them to work with our codebase.**
- We are applying software engineering methodologies alongside machine learning algorithms for seismic picking and association
  - Goal of making our approach as futureproof as possible with new algorithms/changes in technology
  - We hope our work will encourage further development
- As mentioned, we have a notebook that demonstrates stringing together results across the picker, pick-filter and associator API endpoints
  - See us at the end of the presentation if interested

Supplementary Slides

Ryan Tam

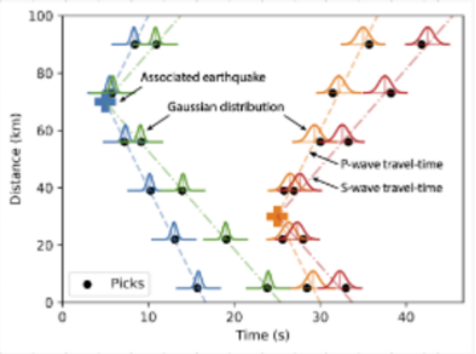
Demo Scheme





Picking

Pick Filtering



Association

INPUT: Waveforms
OUTPUT: Events with Associated Picks

Demo: How To Use the Picker, Pick Filter and Associator By Accessing Docker-Staged Endpoints

picker_pickfilter_associator_doc

localhost:8888/notebooks/picker_pickfilter_associator_dockerapi.ipynb

jupyter picker_pickfilter_associator_dockerapi Last Checkpoint: 49 seconds ago

File Edit View Run Kernel Settings Help

Not Trusted

Markdown

JupyterLab Python 3 (ipykernel)

Launch pickfilter and associator APIs via Docker Compose

This notebook demonstrates how to:

1. Start the `picker`, `pickfilter` and `associator` services defined in `docker-compose.yml` using detached mode.
2. Verify that the API for the `picker`, `pickfilter` and `associator` services are running inside the container.
3. Ping the `picker`, `pickfilter` and `associator` service endpoints with a payload.
4. Shut down the containers afterward.

Import Helper Scripts and Set Up API Endpoints

This section imports required libraries, defines file paths, and sets the base URLs for different microservices used in the application.

[ ]:
```
#import helper scripts

import subprocess, time, requests, json, os
from pathlib import Path

# Paths
compose_path = Path("/app/apicall/docker-compose.yml")

pick_filter_url = "http://pickfilter-api:9797"
association_url = "http://associator-api:9595"
picker_url = "http://picker-api:9494"

aws_picker_model ="https://0vpnsbvff5.execute-api.us-west-2.amazonaws.com/predict/{proxy+}"
```

[ ]:
```
import subprocess
import time
from pathlib import Path

# Path to your docker-compose.yml inside the container
compose_path = Path("/app/apicall/docker-compose.yml")

print("Starting container via host Podman socket...\n")

try:
    # Run Podman Compose via host socket
    subprocess.run([
        "podman",
```

picker_pickfilter_associator_doc

localhost:8888/notebooks/picker_pickfilter_associator_dockerapi.ipynb

Finish update

jupyter picker_pickfilter_associator_dockerapi Last Checkpoint: 1 minute ago

File Edit View Run Kernel Settings Help

Not Trusted

Markdown

JupyterLab Python 3 (ipykernel)

saved to a JSON file.

+1 cell hidden

Plot found picks on top of their respective waveforms

This notebook loads seismic waveform data and pick data (P and S phases), matches them by station/network/channel, and overlays the picks on top of the waveform plots. The results are saved as PNG files.

[7]:

```
#Plot found picks on top of its respective waveforms

import json
import matplotlib.pyplot as plt
from datetime import datetime
import numpy as np
from dateutil import parser
import os

from IPython.display import Image, display

def load_json(file_path):
    """Load and return JSON data from a file."""
    with open(file_path, 'r', encoding='utf-8') as f:
        return json.load(f)

def find_matching_waveform(pick, waveforms):
    """Find the matching waveform by station, network, and channel."""
    pick_station = pick["Site"]["Station"]
    pick_network = pick["Site"]["Network"]
    pick_channel = pick["Site"]["Channel"]

    # Find waveform with matching station, network, and channel
    for waveform in waveforms:
        if (waveform["station"] == pick_station and
            waveform["network"] == pick_network and
            waveform["channel"] == pick_channel):
            return waveform
    return None  # Return None if no match found

def parse_iso8601(date_str):
    """Parse an ISO 8601 date string to a datetime object using dateutil.parser."""
    return parser.isoparse(date_str)

def plot_waveform_with_picks(waveform, picks, output_dir='plots/'):
    """Plot waveform and overlay multiple picks."""
```