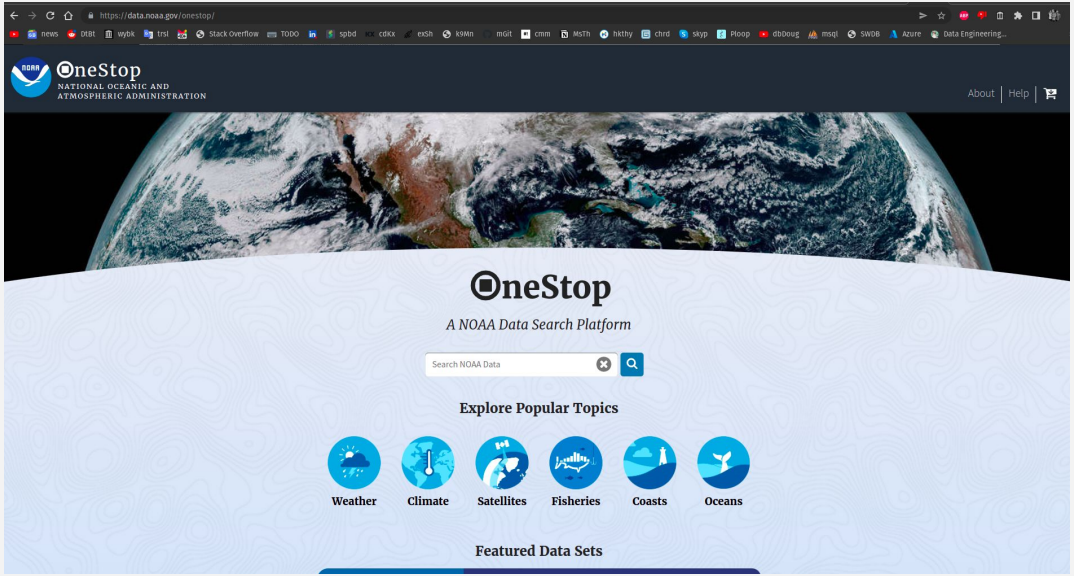
Final Submission

Capstone

Background



Index of /pub/data/swdi/stormevents/csvfiles

Name	Last modified	Size	Description
Parent Directory		-	
Storm-Data-Bulk-csv-Format.pdf	2020-07-17 13:10	161K	
Storm-Data-Export-Format.pdf	2020-07-17 09:17	163K	
StormEvents_details-ftp_v1.0_d1950_c20210803.csv.gz	2021-08-05 09:53	10K	
StormEvents_details-ftp_v1.0_d1951_c20210803.csv.gz	2021-08-05 09:56	12K	
StormEvents_details-ftp_v1.0_d1952_c20210803.csv.gz	2021-08-05 09:56	12K	
StormEvents_details-ftp_v1.0_d1953_c20210803.csv.gz	2021-08-05 09:56	21K	
StormEvents_details-ftp_v1.0_d1954_c20210803.csv.gz	2021-08-05 09:56	26K	
StormEvents_details-ftp_v1.0_d1955_c20210803.csv.gz	2021-08-05 09:56	52K	
StormEvents_details-ftp_v1.0_d1956_c20210803.csv.gz	2021-08-05 09:56	62K	
StormEvents_details-ftp_v1.0_d1957_c20210803.csv.gz	2021-08-05 09:56	80K	
StormEvents_details-ftp_v1.0_d1958_c20210803.csv.gz	2021-08-05 09:56	69K	
StormEvents_details-ftp_v1.0_d1959_c20210803.csv.gz	2021-08-05 09:56	66K	
StormEvents_details-ftp_v1.0_d1960_c20210803.csv.gz	2021-08-05 09:56	70K	
StormEvents_details-ftp_v1.0_d1961_c20210803.csv.gz	2021-08-05 09:56	81K	
StormEvents_details-ftp_v1.0_d1962_c20210803.csv.gz	2021-08-05 09:56	83K	
StormEvents_details-ftp_v1.0_d1963_c20210803.csv.gz	2021-08-05 09:56	70K	
StormEvents_details-ftp_v1.0_d1964_c20210803.csv.gz	2021-08-05 09:56	84K	
StormEvents_details-ftp_v1.0_d1965_c20210803.csv.gz	2021-08-05 09:56	102K	
StormEvents_details-ftp_v1.0_d1966_c20210803.csv.gz	2021-08-05 09:56	81K	
StormEvents_details-ftp_v1.0_d1967_c20210803.csv.gz	2021-08-05 09:56	95K	
StormEvents_details-ftp_v1.0_d1968_c20210803.csv.gz	2021-08-05 09:56	112K	
StormEvents_details-ftp_v1.0_d1969_c20210803.csv.gz	2021-08-05 09:56	100K	
StormEvents_details-ftp_v1.0_d1970_c20210803.csv.gz	2021-08-05 09:56	112K	
StormEvents_details-ftp_v1.0_d1971_c20210803.csv.gz	2021-08-05 09:56	123K	
StormEvents_details-ftp_v1.0_d1972_c20220425.csv.gz	2022-04-25 15:06	80K	
StormEvents_details-ftp_v1.0_d1973_c20220425.csv.gz	2022-04-25 15:06	157K	
StormEvents_details-ftp_v1.0_d1974_c20220425.csv.gz	2022-04-25 15:06	183K	
StormEvents_details-ftp_v1.0_d1975_c20220425.csv.gz	2022-04-25 15:06	172K	
StormEvents_details-ftp_v1.0_d1976_c20220425.csv.gz	2022-04-25 15:06	133K	

Alternative Stacks Explored

Initially explored data in q/KDB+ (instead of MySQL).

Speed using my laptop was orders of magnitude faster (~20 secs to copy full dataset from gzip'd files into tables, ½ that being decompression)

unfortunately, licensing costs squanders commercial feasibility for all but the most lucrative ventures

```
//DECOMPRESS FILES
tz0:.z.p
system "gzip -kd gzipped/*"
tz1:.z.p
tgz:tz1-tz0

//INGEST DETAILS
t0:.z.p
detfiles:asc hsym each `$" ":/home/conner/testgz/gzipped/",/:  system "ls gzipped | grep -v gz | grep details"
details: (,/) {(51#"*";enlist ",") 0: x} each detfiles
t1:.z.p

//CREATE DETAILS SUBSET TABLE AND CAST COLUMN TYPES
det:select BEGIN_YEARMONTH,BEGIN_DAY,END_YEARMONTH,END_DAY,"I"$EPISODE_ID,"I"$EVENT_ID,EVENT_TYPE,
    `$STATE,"I"$INJURIES_DIRECT,"I"$INJURIES_INDIRECT,"I"$DEATHS_DIRECT,"I"$DEATHS_INDIRECT,"F"$BEGIN_LAT,"F"$BEGIN_LON from details
update BEGIN_DATE:(BEGIN_YEARMONTH,'BEGIN_DAY) from `det where not 1=count each BEGIN_DAY;
update BEGIN_DATE:(BEGIN_YEARMONTH,'("0",'BEGIN_DAY)) from `det where 1=count each BEGIN_DAY;
update END_DATE:(END_YEARMONTH,'END_DAY) from `det where not 1=count each END_DAY;
update END_DATE:(END_YEARMONTH,'("0",'END_DAY)) from `det where 1=count each END_DAY;

//CALC DETAILS ELAPSED TIMES
t2:.z.p;td1:t1-t0;td2:t2-t1;td3:t2-t0;t4:.z.p

//INGEST FATALITES
fatfiles:asc hsym each `$" ":/home/conner/testgz/gzipped/",/:  system "ls gzipped | grep -v gz | grep fatalities"
fatalities: (,/) {(11#"*";enlist ",") 0:x} each fatfiles
t5:.z.p

//CREATE FATALITES SUBSET TABLE AND CAST COLUMN TYPES
fat: select FATALITY_DATE,FATALITY_ID,EVENT_ID,FATALITY_TYPE,FATALITY_AGE,FATALITY_SEX,FATALITY_LOCATION from fatalities
update "I"$FATALITY_ID,"I"$EVENT_ID,`$FATALITY_TYPE,"I"$FATALITY_AGE,`$FATALITY_SEX from `fat;
update "D"$10#'FATALITY_DATE from `fat;

//CALC FATALITIES ELAPSED TIMES
t6:.z.p;td4:t5-t4;td5:t6-t5;td6:t6-t4;td7:t6-tz0;show ""

//PRINT SCRIPT TOTAL ELAPSED TIME
show (enlist `$"UNZIPPING TIME: "}!enlist `$((-6_8_string tgz)," secs")
show ""
//PRINT DETAILS SUMMARY DICT
show (`$"TABLE: ";`$"ROWS:";`$"COLS:";`$"COPY:";`$"CAST:";`$"TOTAL:")!
    `details,(`$string count details),(`$string count cols details),`$'(-6_'8_'string value each `td1`td2`td3), \: " secs"
show ""

//PRINT FATALITIES SUMMARY DICT
show (`$"TABLE: ";`$"ROWS:";`$"COLS:";`$"COPY:";`$"CAST:";`$"TOTAL:")!
    `fatalities,(`$string count fatalities),(`$string count cols fatalities),`$'(-6_'8_'string value each `td4`td5`td6), \: " secs"
show ""

//PRINT SCRIPT TOTAL ELAPSED TIME
show (enlist `$"FULL SCRIPT RUN ELAPSED TIME: "}!enlist `$((-6_8_string td7)," secs")
show ""
\\
```

```
conner@Linuxclevo:~/testgz$ q ingest_all_gz.q
KDB+ 4.0 2022.05.11 Copyright (C) 1993-2022 Kx Systems
l64/ 8(24)core 23735MB conner linuxclevo 127.0.1.1 EXPIRE 2023.06.30 connermcnicholas@gmail.com KXCE #72902

()
""
UNZIPPING TIME:| 08.057 secs
""
TABLE:| details
ROWS: | 1740597
COLS: | 51
COPY: | 10.097 secs
CAST: | 03.174 secs
TOTAL:| 13.272 secs
""
TABLE:| fatalities
ROWS: | 20368
COLS: | 11
COPY: | 00.029 secs
CAST: | 00.007 secs
TOTAL:| 00.036 secs
""
FULL SCRIPT RUN ELAPSED TIME:| 21.366 secs
""
conner@Linuxclevo:~/testgz$
```

Alternative Stacks Explored

Apache Airflow as ELT orchestration framework (instead of Azure Data Factory)

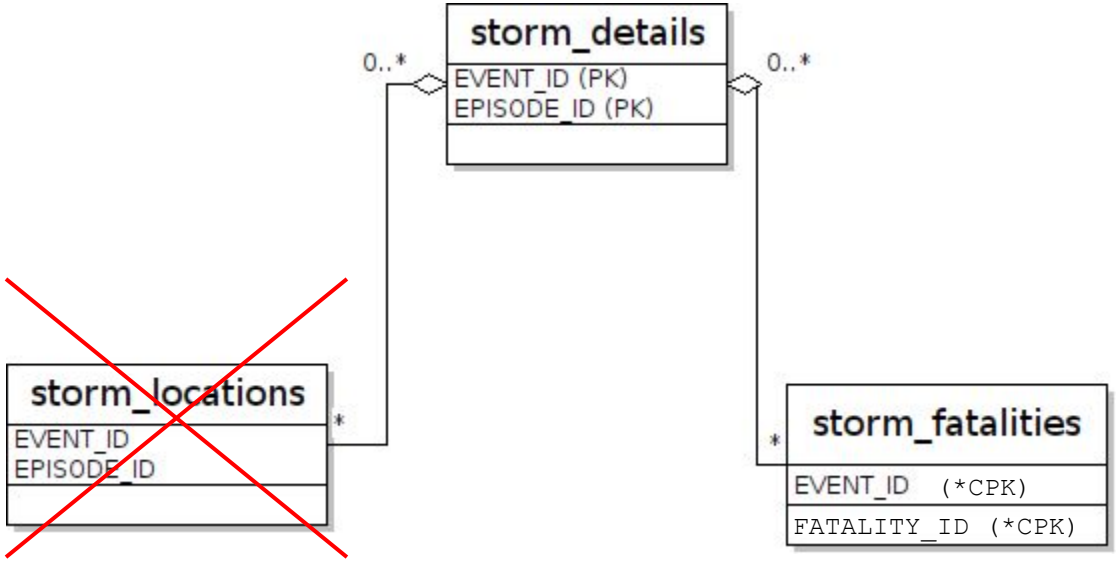
Spark as datastore and analytics engine via Databricks cluster (instead of MySQL+/Workbench)

Azure MS SQL Server+/SandDance (instead of MySQL+/Workbench)

Azure Batch as compute runtime (instead of Databricks cluster)

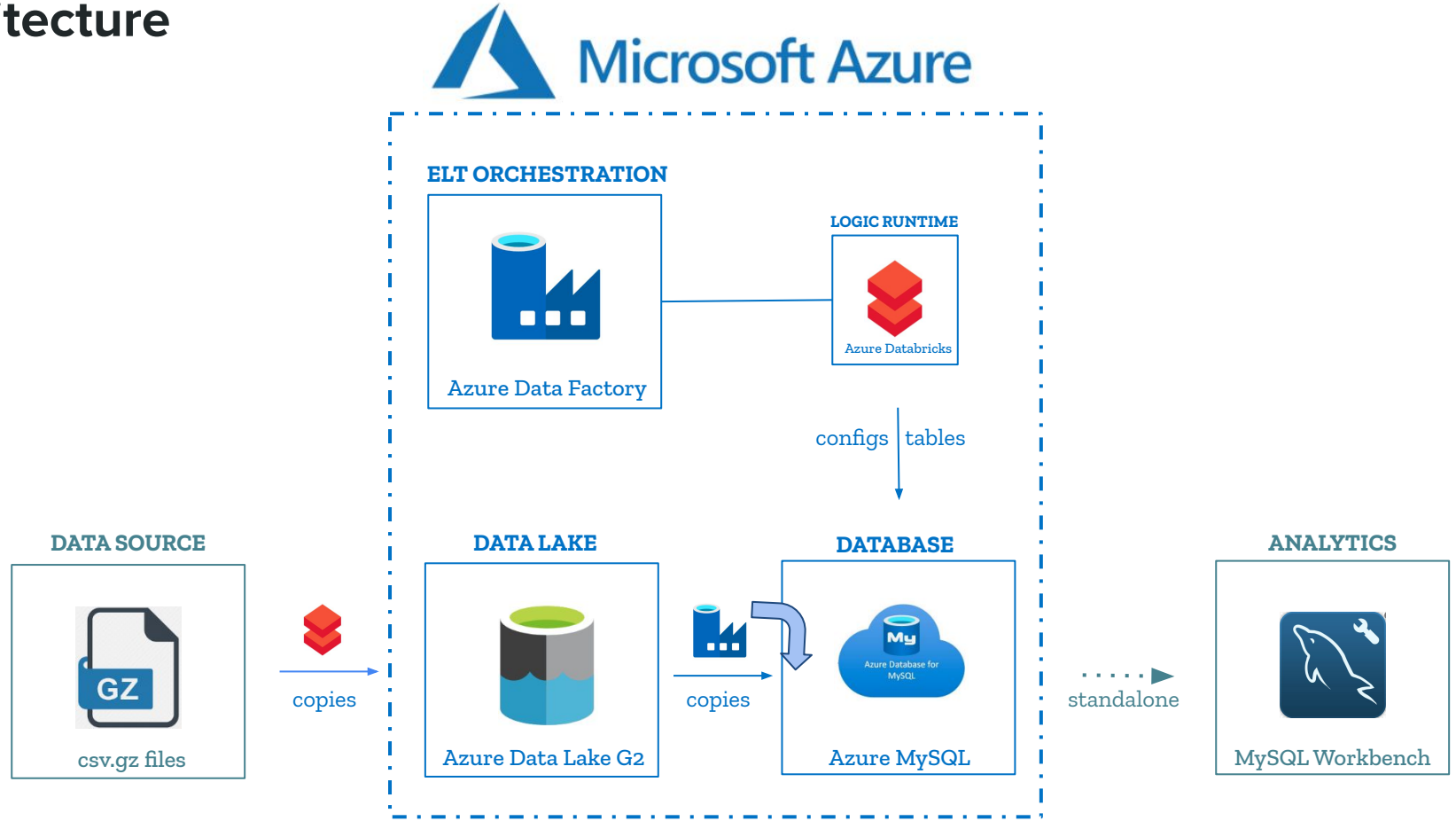
Azure Synapse Analytics as analytics engine (instead of MySQL+/Workbench)

Data Model



ALL LOCATIONS DATA REPRODUCED IN DETAILS

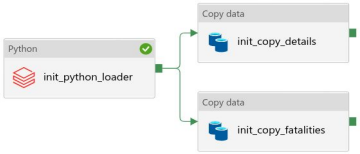
Architecture



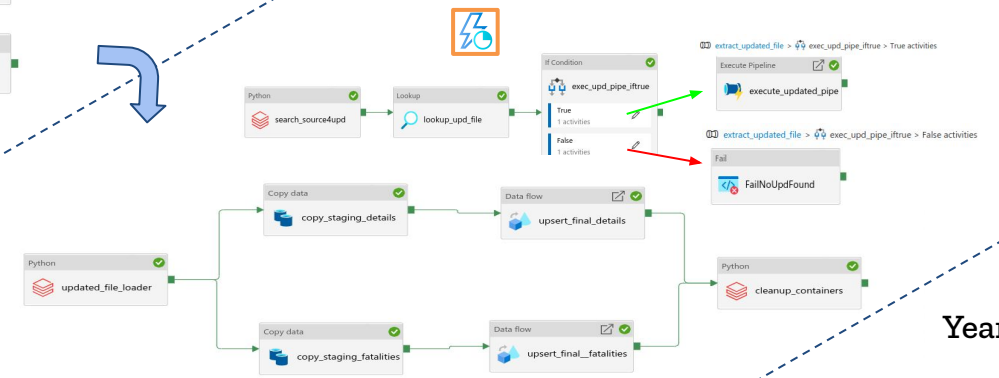
ETL Overview

Three Azure Data Factory pipelines tailored for different use case based on source data release cadence

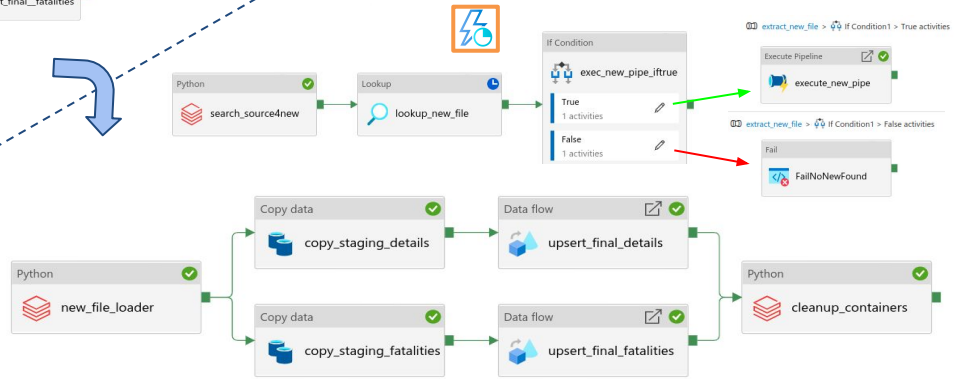
Initial Load



Monthly Update

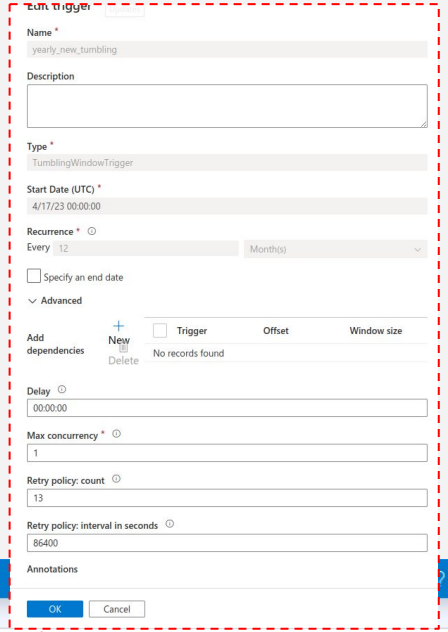


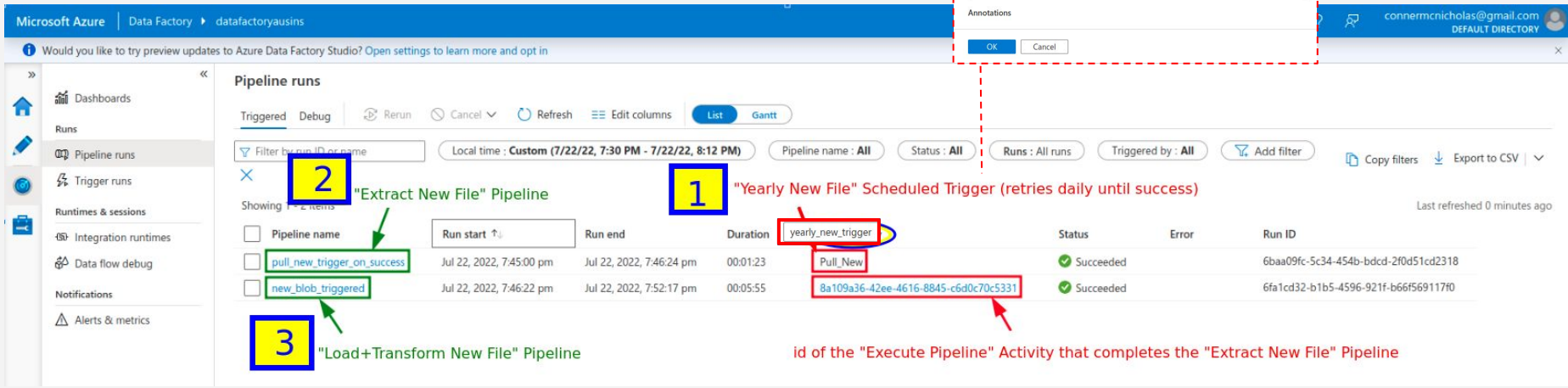
Yearly New



Deep Dive: “Yearly New” ETL

1) New File Trigger
   - Annually on April 17th Triggers “Extract New File” Pipeline
2) “Extract New File” Pipeline
   - check source url for new file drop
     - pull if available
       - success: triggers “Load+Transform New File” Pipeline
       - failure: triggers retry in 1 day
3) “Load+Transform New File” Pipeline



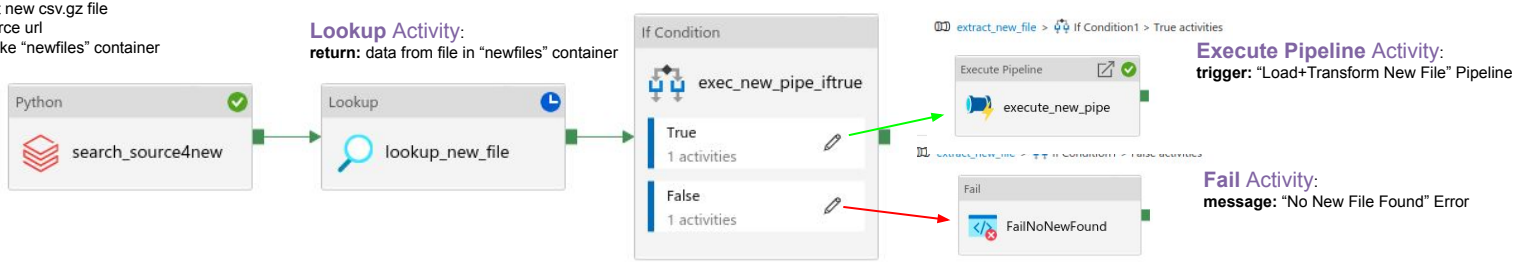


"Extract New File" Pipeline

Databricks Python Activity:
spark cluster: initialized with environment variables for auth
runs: `python new_file_to_blob_only.py`
if: source url has dropped a new filename for new year
then: extract new csv.gz file
from: source url
to: data lake "newfiles" container

Lookup Activity:
return: data from file in "newfiles" container

If Condition Activity:
if: # of rows from Lookup'd data > 0
then: trigger "Load+Transform New File" Pipeline
else: trigger "Fail" of pipeline, enabling retry in 1 day



Execute Pipeline Activity:
trigger: "Load+Transform New File" Pipeline

Fail Activity:
message: "No New File Found" Error

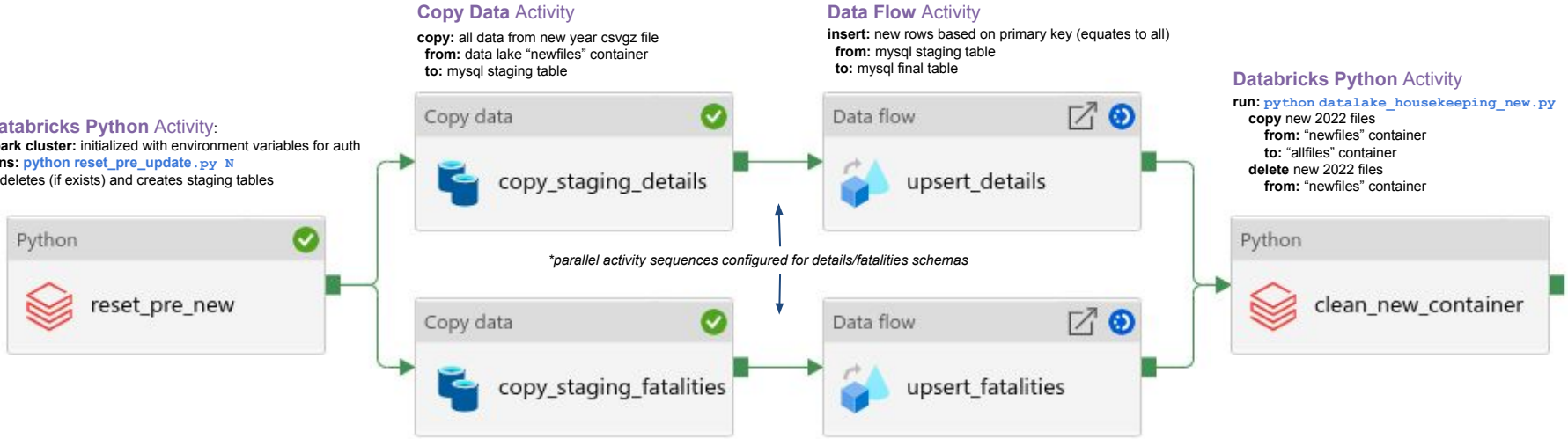
"Load+Transform New File" Pipeline

Databricks Python Activity:
spark cluster: initialized with environment variables for auth
runs: `python reset_pre_update.py N`
deletes (if exists) and creates staging tables

Copy Data Activity
copy: all data from new year csvgz file
from: data lake "newfiles" container
to: mysql staging table

Data Flow Activity
insert: new rows based on primary key (equates to all)
from: mysql staging table
to: mysql final table

Databricks Python Activity
run: `python datalake_housekeeping_new.py`
copy new 2022 files
from: "newfiles" container
to: "allfiles" container
delete new 2022 files
from: "newfiles" container

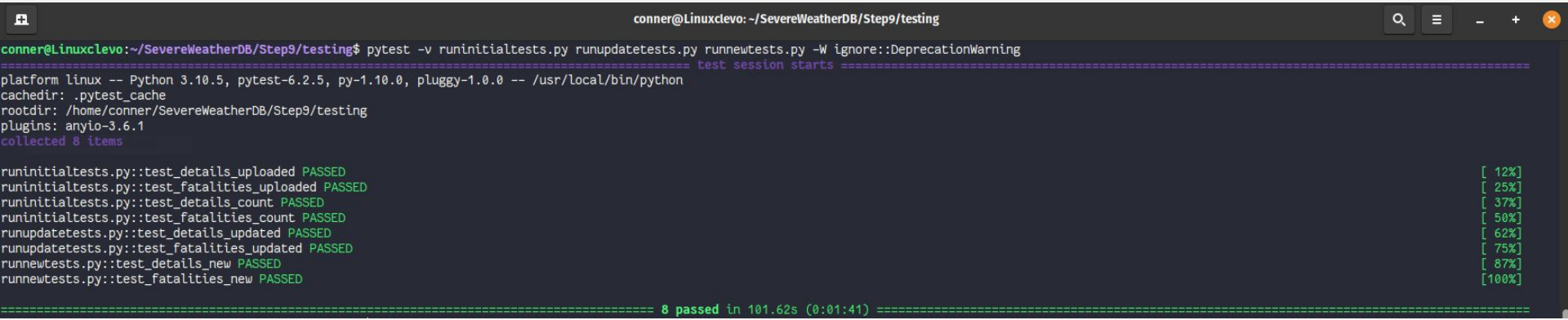


Testing

General Tests verify each year has a file in Data Lake for both table

Pipeline Tests verify each line from source files have rows in both MySQL tables

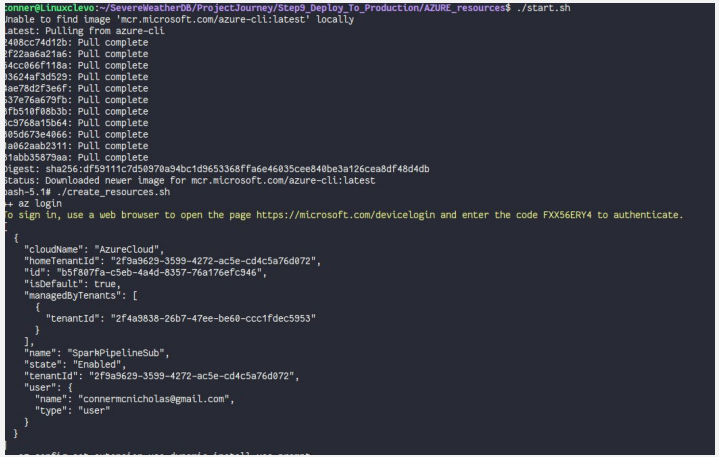
- 8 *Total Tests* = 2 *General Tests* + 6 *Pipeline Tests*
  - 2 *General Tests* = 1 *General Test* x 2 *Tables*
  - 6 *Pipeline Tests* = 3 *Pipeline Tests* x 2 *Tables*



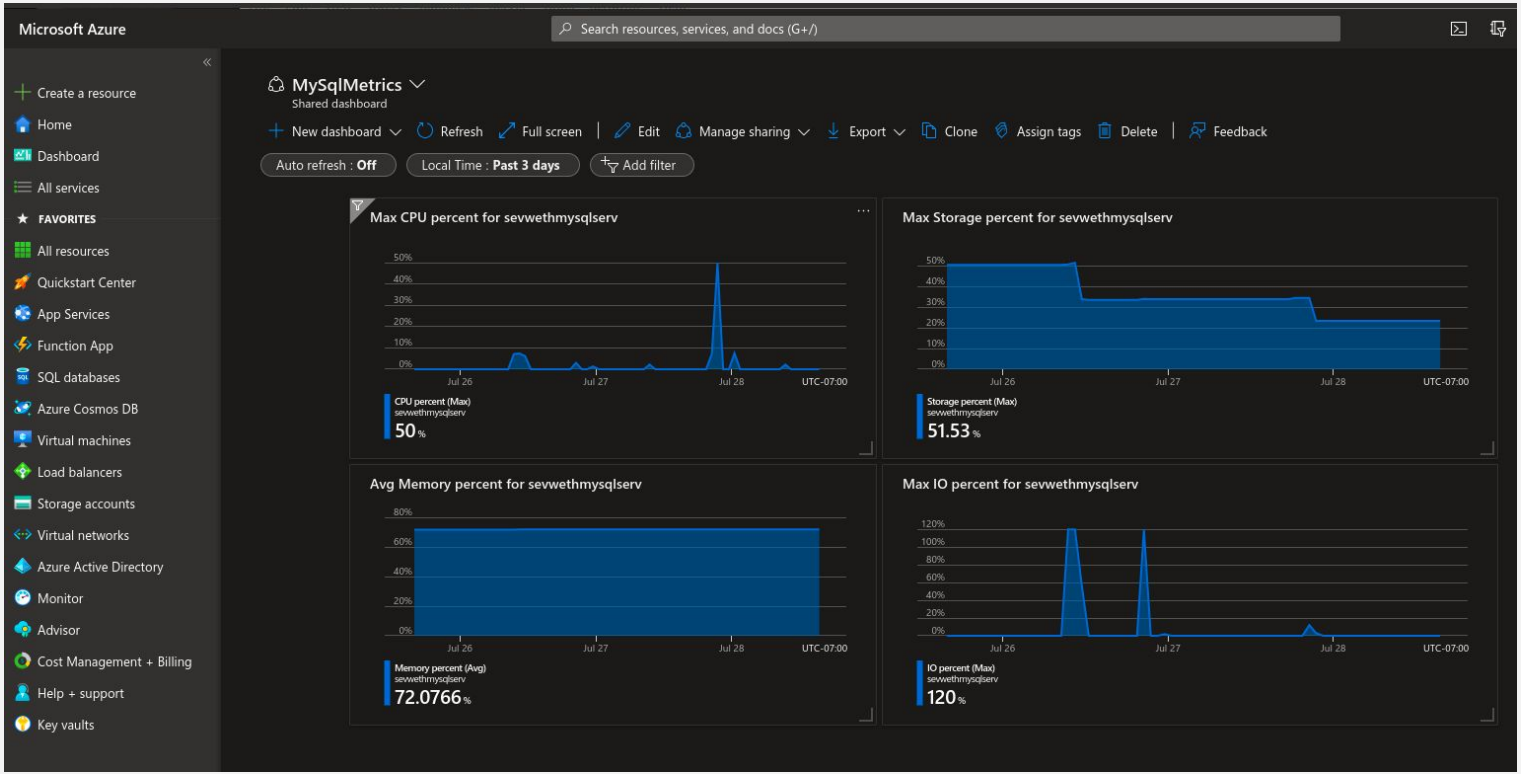
Deployment

Deploys Azure resources via Docker container image:

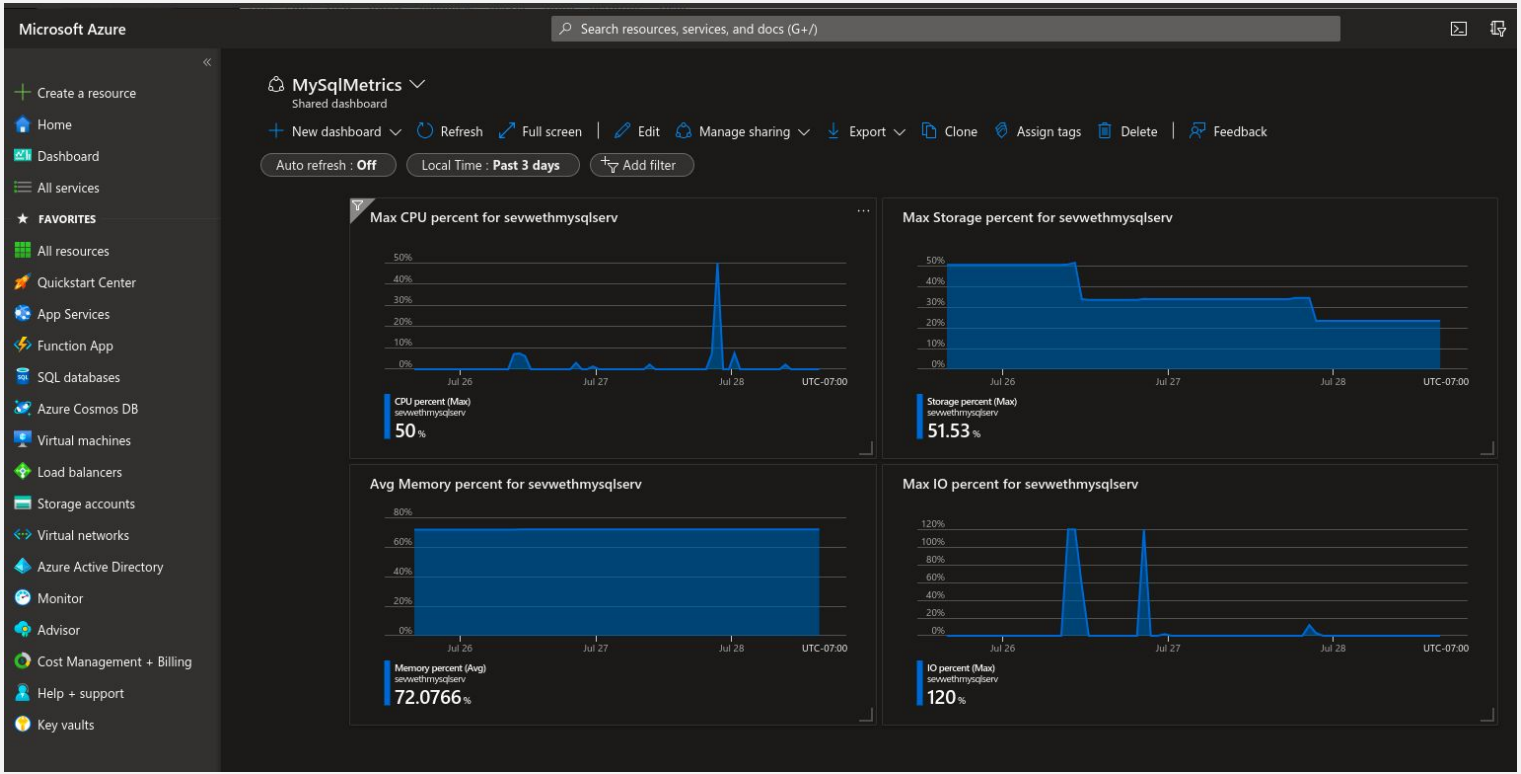
- data lake blob storage
- mysql database
- databricks
- data factory
    iii. creates pipelines based on json configs in DATAFACTORY_pipelines
    iv. runs init pipeline to ingest all available data at source
- Requirement: Docker
    iii. to start azure cli container, run from local terminal:
- ./start.sh
    iii. now from inside azure cli container shell, run:
- `bash-5.1#./create_resources.sh`
    iii. login to az by entering given code at https://microsoft.com/devicelogin
- json metadata describing created resources will print to stdout (execution_log.txt)



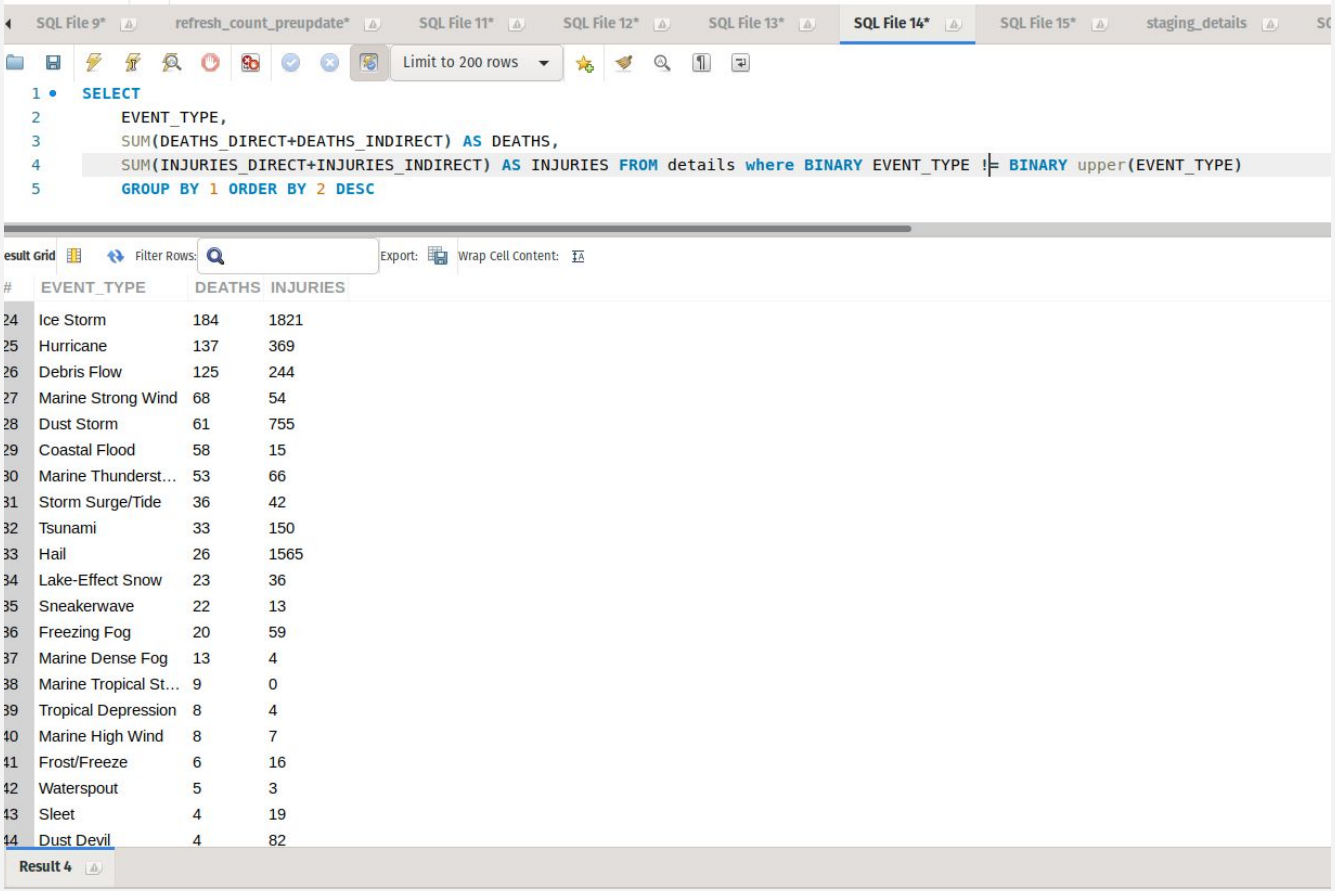
Monitoring


Microsoft Azure
Search resources, services, and docs (G+/)
Create a resource
Home
Dashboard
All services
FAVORITES
All resources
Quickstart Center
App Services
Function App
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Azure Active Directory
Monitor
Advisor
Cost Management + Billing
Help + support
Key vaults
MySqlMetrics
Shared dashboard
New dashboard
Refresh
Full screen
Edit
Manage sharing
Export
Clone
Assign tags
Delete
Feedback
Auto refresh : Off
Local Time : Past 3 days
Add filter
Max CPU percent for sevwethmysqlserv
50%
40%
30%
20%
10%
0%
Jul 26
Jul 27
Jul 28
UTC-07:00
CPU percent (Max)
sevwethmysqlserv
50 %
Max Storage percent for sevwethmysqlserv
50%
40%
30%
20%
10%
0%
Jul 26
Jul 27
Jul 28
UTC-07:00
Storage percent (Max)
sevwethmysqlserv
51.53 %
Avg Memory percent for sevwethmysqlserv
80%
60%
40%
20%
0%
Jul 26
Jul 27
Jul 28
UTC-07:00
Memory percent (Avg)
sevwethmysqlserv
72.0766 %
Max IO percent for sevwethmysqlserv
120%
100%
80%
60%
40%
20%
0%
Jul 26
Jul 27
Jul 28
UTC-07:00
IO percent (Max)
sevwethmysqlserv
120 %

Monitoring


Microsoft Azure
Search resources, services, and docs (G+/)
Create a resource
Home
Dashboard
All services
FAVORITES
All resources
Quickstart Center
App Services
Function App
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Azure Active Directory
Monitor
Advisor
Cost Management + Billing
Help + support
Key vaults
MySqlMetrics
Shared dashboard
New dashboard
Refresh
Full screen
Edit
Manage sharing
Export
Clone
Assign tags
Delete
Feedback
Auto refresh : Off
Local Time : Past 3 days
Add filter
Max CPU percent for sevwethmysqlserv
50%
40%
30%
20%
10%
0%
Jul 26
Jul 27
Jul 28
UTC-07:00
CPU percent (Max)
sevwethmysqlserv
50 %
Max Storage percent for sevwethmysqlserv
50%
40%
30%
20%
10%
0%
Jul 26
Jul 27
Jul 28
UTC-07:00
Storage percent (Max)
sevwethmysqlserv
51.53 %
Avg Memory percent for sevwethmysqlserv
80%
60%
40%
20%
0%
Jul 26
Jul 27
Jul 28
UTC-07:00
Memory percent (Avg)
sevwethmysqlserv
72.0766 %
Max IO percent for sevwethmysqlserv
120%
100%
80%
60%
40%
20%
0%
Jul 26
Jul 27
Jul 28
UTC-07:00
IO percent (Max)
sevwethmysqlserv
120 %

Query


SQL File 9*
refresh_count_preupdate*
SQL File 11*
SQL File 12*
SQL File 13*
SQL File 14*
SQL File 15*
staging_details
Limit to 200 rows
1 • SELECT
2 EVENT_TYPE,
3 SUM(DEATHS_DIRECT+DEATHS_INDIRECT) AS DEATHS,
4 SUM(INJURIES_DIRECT+INJURIES_INDIRECT) AS INJURIES FROM details where BINARY EVENT_TYPE != BINARY upper(EVENT_TYPE)
5 GROUP BY 1 ORDER BY 2 DESC
esult Grid
Filter Rows:
Export:
Wrap Cell Content:
EVENT_TYPE DEATHS INJURIES
24 Ice Storm 184 1821
25 Hurricane 137 369
26 Debris Flow 125 244
27 Marine Strong Wind 68 54
28 Dust Storm 61 755
29 Coastal Flood 58 15
30 Marine Thunderst... 53 66
31 Storm Surge/Tide 36 42
32 Tsunami 33 150
33 Hail 26 1565
34 Lake-Effect Snow 23 36
35 Sneakerwave 22 13
36 Freezing Fog 20 59
37 Marine Dense Fog 13 4
38 Marine Tropical St... 9 0
39 Tropical Depression 8 4
40 Marine High Wind 8 7
41 Frost/Freeze 6 16
42 Waterspout 5 3
43 Sleet 4 19
44 Dust Devil 4 82
Result 4