Documentation

This documentation highlights how I set up the AWS instances as well as how to run Node.js and Android Studio.

AWS:

All our AWS instances are self hosted using Amazon Linux 2, this includes Kafka cluster, Telegraf, InfluxDB, and Grafana.

Here's how I set up kafka cluster:

Setup zookeeper:

SSH into the ec2 instance:

```
sudo yum update -y
sudo yum install -y java-1.8.0-openjdk

wget https://downloads.apache.org/kafka/3.9.0/kafka_2.13-3.9.0.tgz
tar -xzf kafka_2.13-3.9.0.tgz
mv kafka_2.13-3.9.0 kafka
sudo mv kafka /opt

nano /opt/kafka/config/zookeeper.properties
```

ADD THESE IN:

```
maxClientCnxns=50
tickTime=2000
initLimit=5
syncLimit=2
```

Add this after you make the other 2 instances:

```
server.1=0.0.0.0:2888:3888
server.2=zookeeper-node2:2888:3888
server.3=zookeeper-node3:2888:3888
```

Use this to test the Zookeeper(will need to set up as systemctl later):

```
/opt/kafka/bin/zookeeper-server-start.sh /opt/kafka/config/zookeeper.properties &

nano /opt/kafka/config/server.properties
```

Set up the broker.id based on which instance it is (ex. Instance 1 is broker.id = 1)
Zookeeper.connect needs to point to all instances with the current instance as localhost.

```
Zookeeper.connect = localhost:2181,<other Zookeeper public ip>:2181,<other Zookeeper public ip>:2181
```

Add this in

```
listeners=PLAINTEXT://0.0.0.0:9092
```

```
advertised.listeners=PLAINTEXT://<the ec2 instance's public ip>:9092
Listener.security.protocol.map=PLAINTEXT:PLAINTEXT
```

Use this to test the Kafka(will need to set up as systemctl later):

```
/opt/kafka/bin/kafka-server-start.sh /opt/kafka/config/server.properties &
```

Create test topic:

```
/opt/kafka/bin/kafka-topics.sh --create --topic test-topic --bootstrap-server localhost:9092
--replication-factor 3 --partitions 1
```

Test sending a message:

```
/opt/kafka/bin/kafka-console-producer.sh --topic test-topic --bootstrap-server localhost:9092
```

Test consuming a message:

```
/opt/kafka/bin/kafka-console-consumer.sh --topic test-topic --from-beginning --bootstrap-server
localhost:9092
```

Setting up Telegraf:

```
sudo yum update -y
sudo rpm –import https://repos.influxdata.com/influxdata-archive_compat.key
sudo yum install telegraf
```

Modify telegraf config:

```
sudo nano /etc/telegraf/telegraf.conf
```

Add these or uncomment them in the config:

```
[[inputs.kafka_consumer]]
  ## List of Kafka broker addresses (format: "host:port").
  brokers = ["localhost:9092", "<other kafka instance public private ip>:9092", "<other kafka
instance public private ip>:9092"]

  ## Topics to consume messages from.
  topics = ["test-topic"]  # Replace with your Kafka topic name

  ## The name of the consumer group.
  consumer_group = "telegraf-group"

  ## Offset (either "oldest" or "newest").
  offset = "oldest"

  ## Maximum number of messages to consume at once.
  max_message_len = 1000000

  ## Data format to parse incoming messages.
  data_format = "json"  # Assuming your messages are in JSON format



[[outputs.influxdb_v2]]
  ## InfluxDB URL.
  urls = ["http://<influxdb ec2 instance>:8086"]

  ## InfluxDB authentication token.
  token = "your-influxdb-token"

  ## InfluxDB organization name.
  organization = "your-org"

  ## InfluxDB bucket name.
  bucket = "your-bucket"
```

```
sudo systemctl start telegraf
sudo systemctl enable telegraf
sudo systemctl status telegraf
```

Test data flow

```
/opt/kafka/bin/kafka-console-producer.sh --topic test-topic --bootstrap-server localhost:9092
```

REPEAT THIS FOR TWO MORE INSTANCES THEN UPDATE THE PARTS THAT SAY <OTHER INSTANCE IP>.

<u>Setup as systemctl so it runs even when SSH is closed:</u>

```
sudo nano /etc/systemd/system/zookeeper.service
```

Paste this in:

```
[Unit]
Description=Apache Zookeeper Server
Documentation=https://zookeeper.apache.org
Requires=network.target
After=network.target

[Service]
Type=simple
ExecStart=/opt/kafka/bin/zookeeper-server-start.sh /opt/kafka/config/zookeeper.properties
ExecStop=/opt/kafka/bin/zookeeper-server-stop.sh
Restart=on-abnormal
```

```
[Install]
WantedBy=multi-user.target
```

```
sudo systemctl daemon-reload
sudo systemctl enable zookeeper
sudo systemctl start zookeeper
sudo systemctl status zookeeper
```

Do this step for all Zookeeper instances and make sure it is running without failing before going on to Kafka.

```
sudo nano /etc/systemd/system/kafka.service
```

Paste this in:

```
[Unit]
Description=Apache Kafka Server
Documentation=https://kafka.apache.org/documentation/
Requires=zookeeper.service
After=zookeeper.service

[Service]
Type=simple
Environment="KAFKA_HEAP_OPTS=-Xmx1G -Xms1G"
ExecStart=/opt/kafka/bin/kafka-server-start.sh /opt/kafka/config/server.properties
ExecStop=/opt/kafka/bin/kafka-server-stop.sh
Restart=on-abnormal
User=kafka
Group=kafka

[Install]
WantedBy=multi-user.target
```

```
sudo useradd -r -m -U -d /opt/kafka -s /bin/bash kafka
sudo chown -R kafka:kafka /opt/kafka
sudo systemctl daemon-reload
sudo systemctl enable kafka
sudo systemctl status kafka
```

REPEAT FOR ALL INSTANCES

If Kafka says failed after starting service try:

```
sudo chown -R kafka:kafka /tmp/kafka-logs
sudo chmod -R 755 /tmp/kafka-logs
```

If this doesn’t work:

```
sudo nano /opt/kafka/config/server.properties
log.dirs=/var/lib/kafka-logs
sudo mkdir -p /var/lib/kafka-logs
sudo chown -R kafka:kafka /var/lib/kafka-logs
sudo chmod -R 755 /var/lib/kafka-logs

sudo systemctl restart kafka
sudo systemctl status kafka
```

If this works repeat for all instances

Setup InfluxDB:

Same setup as Kafka instance.

```
sudo yum update -y

sudo yum install https://dl.influxdata.com/influxdb/releases/influxdb2-2.7.1.x86_64.rpm

sudo nano /etc/influxdb/influxdb.conf
[http]
  enabled = true
  bind-address = ":8086"
  auth-enabled = true

sudo systemctl start influxdb
sudo systemctl enable influxdb
sudo systemctl status influxdb
```

Set up the database here:

http://<InfluxDB ec2 instance public ip>:8086

AFTER THIS IS DONE REPLACE THE INFLUXDB IP IN THE TELEGRAF CONFIG.

Setup Grafana:

Setup new instance similarly to InlfuxDB

I didn’t need to import the package from grafana.repo but the gpg key and link are online.

```
sudo yum install grafana -y
sudo systemctl start grafana-server
sudo systemctl enable grafana-server
```

http://<Grafana ec2 public ip>:3000

Then follow this:
https://docs.influxdata.com/influxdb/v2/tools/grafana/

If data flow works then it's time to set up the Arduino. Go to https://github.com/floodnet-nyc/flood-sensor. Then copy their steps for setting up the firmware. We weren't able to set up LoRaWAN but the next group should set it up.

My use case for this documentation is based on the LoRaWAN router not working and will be sending information via wifi. Look at the serialReader.js file to see the logic. That file reads serial inputs via usb to write data but ideally it should run without the need to plug in a usb cable.

Node.js:

The file included for Node.js should include a package. json file and package lock file. I removed all the API tokens, buckets, etc so it won't run without the variables but the next group should include their own.

```
npm install
node tomorrow.js
```

My USB port is COM3 but change port to the USB port being used

```
node serialReader.js
```

In case of error:
Delete package .json and package lock .json then re-initialize the project.

cd into project:
```
npm init
npm install
```

Android Studio:

I removed the API tokens, buckets, etc as well as the images used in the about us section for privacy reasons. If needed, remove the image tags to ensure the app runs.

I was running a different APG and Android Koala. This means it might be necessary to completely remove the Gradle files as well as the generated .idea files then delete cache and re-sync the project. After removing the Gradle files and clearing cache add this in:

```
implementation("com.squareup.retrofit2:retrofit:2.11.0")
implementation("com.squareup.retrofit2:converter-gson:2.11.0")
implementation("com.squareup.okhttp3:logging-interceptor:4.12.0")
implementation("com.influxdb:influxdb-client-java:3.3.0")
implementation("androidx.swiperefreshlayout:swiperefreshlayout:1.2.0-alpha01")
implementation("com.google.android.material:material:1.2.0")
```

Re-sync the project again. This allows the project to get the necessary dependencies. If none of this works, create a new project and manually paste the source code in. Merging code with different versions gets messy.