*****We will follow this script*****

****Configure InfluxDB****

- Download Influx DB

  ```
  docker pull influxdb:1.2.3
  ```

- Get and update the configuration file

  ```
  docker run --rm influxdb influxd config > influxdb.conf
  vi influxdb.conf
  ```

- turn on the admin interface

  ```
  [admin]
    enabled = true
    bind-address = ":8083"
    https-enabled = false
    https-certificate = "/etc/ssl/influxdb.pem"
  ```

- Bring up InfluxDB

  ```
  docker-compose up -d influxdb
  docker-compose logs influxdb
  ```

- Go to administrative interface.

  ```
  show measurements
  ```

****Configure Telegraf****

- Download Telegraf

  ```
  docker pull telegraf:1.2.1
  ```

- Export sample config for Telegraf

  ```
  docker run --rm telegraf -sample-config > telegraf.conf
  ```

- Configure Telegraf to talk to InfluxDB

  ```
  [[outputs.influxdb]]
  ## The full HTTP or UDP endpoint URL for your InfluxDB instance.
  ## Multiple urls can be specified as part of the same cluster,
  ## this means that only ONE of the urls will be written to each interval.
  # urls = ["udp://localhost:8089"] # UDP endpoint example
  urls = ["http://influxdb:8086"] # required
  ## The target database for metrics (telegraf will create it if not exists).
  database = "telegraf" # required
  ```

- Configure Telegraf to listen for StatsD Metrics

  ```
  # # Statsd Server
  [[inputs.statsd]]
  #   ## Address and port to host UDP listener on
  ```

```
service_address = ":8125"
```

- Start StatsD and check the logs

```
docker-compose up -d telegraf
docker-compose logs telegraf
```

- Validate the metrics made it to InfluxDB, from the administrative interface

```
show measurements
select * from cpu
```

****Learn about the different types of counters****

How does Telegraf summarize data for InfluxDB? Default 10s, summarizes and writes to the database.

Counters

Counters are added up written to the DB every 10s.
`docker-compose up test_counter` demonstrate how counters work.
`select * from test_counter`

- leave one counter up and running `docker-compose up -d test_counter`

Gauges

Only the last gauge measurement will be written to the database. Change sleep interval to demonstrate the behavior of multiple samples during a 10s intervale (last gauge is winner)

```
docker-compose run -e SLEEP=1 --rm --name=test_gauge test_gauge
docker-compose run -e SLEEP=3 --rm --name=test_gauge test_gauge
docker-compose run -e SLEEP=5 --rm --name=test_gauge test_gauge
```

- leave one gauge up and running `docker-compose up -d test_gauge`

Timings and Histograms

Timings are special, and have additional statics captured and calculated for the 10s interval. Each interval will count how many measurements were received, the minimum, maximum, stddev and mean are tracked. Run the below commands to demonstrate 10 second intervals, show metrics, repeat with 1 second interval

```
docker-compose run -e SLEEP=1 --rm --name=test_timing test_timing
docker-compose run -e SLEEP=3 --rm --name=test_timing test_timing
docker-compose run -e SLEEP=5 --rm --name=test_timing test_timing
```

- leave one timer up and running `docker-compose up -d test_timing`

****Configure Grafana****

We will now fire up Grafana. `docker-compose up -d grafana`, log into the grafana console

- Username/Pass = admin/admin (class should change this)
- add datasource

Gauge Counter * show mean and count in select. * Discuss granularity of time interval

Timing Counter * show field(count) available values * scale the counter * introduce tags to the statsd line

Single Metric Counters * simulate response time with timers counters * Min response * Avg response * Worst Response

Configure Telegraf Simulated Clients

We will simulate a number of machines running telegraf by simply defining a new service in our `docker-compose.yml` file. Each node will only show the stats from the docker container, but this will serve our purposes for the demonstration. In the real world, telegraf would be on each node we want to monitor for system stats.

```
docker-compose up -d test_client
docker-compose scale test_client=10
```

****Create a Parameterized Dashboard****