Technical introduction to Public Health Surveillance System

Gunnar and Jonathan (Jyri, Mikolaj, Tomek, Ziad)

World Health Organisation

March 20, 2018

1. Overview and Introduction
2. Principles of implementation
3. Technical Structure
   - Data Collection
   - Data processing
   - Web interface
   - Infrastructure

Mission statement

Public health surveillance mission statement

To enable real-time case-based surveillance at the health facility level that improves public health and clinical decision making at all levels of the health system.

To this end the we want to develop secure and reliable technical solutions that fulfill the following principles:

- Configurable services
- A micro-service architecture
- Supporting multiple data source and multiple data outputs
- Open source

Short history

- Spring 2014: Pilot project with 50 clinics Communicable Diseases
- April – September 2015: National scale up to over 300 clinics
- Spring 2016 RMS system started
- NCD Scale up June 2016
- Mental health March 2017

Microservices Architecture

We aim to design the system so that we have a system of loosely coupled smaller services performing a self contained range of functions.

- Each component has a clearly defined limited scope
- Each component should be independently deployable
- Components are mainly coupled through APIs
- Made much easier by Docker

Benefits:

- Each component can be developed independently
- Can make different design decisions for each component
- Components can be easily replaced if needed
- Separation of responsibilities and non repetition of code

Open Source

We only use open source components and release all code as open source.

Using open source brings many benefits:

- No licensing costs etc.
- Other groups can use and improve our software
- Can determine in detail how everything works if needed
- Much industry leading software is open source

Continuous Integration

We develop our software on a continuous release cycle that can always be deployed.

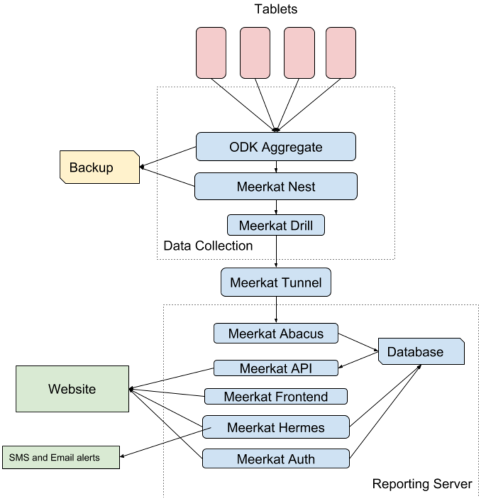
Made possible by:

- Version control via git
- Unit-testing and automatic testing on code push (Travis)
- Nightly builds of development branch
- Automatic build and deploy process
- Flexible infrastructure in the cloud

Benefits:

- Faster deployment of needed changes
- Continuous feedback on changes and incremental improvements

Current Data Flow



Data Flow

The main function of the software is to get the information from the tablets to the users in a useful format.

We want to support many different sources of data and different formats.

E.g: Gender might be coded as M/F in variable gen in one form and as male/female in variable pt1./gender in another.

To overcome this we process all the raw data using configurable codes so that both way of specifying gender can be combined. The end point of this is a clean data table which can easily be used for aggregation.

This data can then be used to power the websites and other data displays

Data Collection

Data collection modalities:

- Forms submitted from tablets (Main modality)
- Automatic status messages from tablets (Under development)
- Uploading spreadsheets (Experimental feature)
- Data from other sources via APIs (Future)

All collected data will be stored on the servers in MoH/RMS and is owned by them.

Data collection from tablets

- Android tablets that run custom version of ODK Collect
- Automatic synchronization of app and forms
- Automated configuration
- Forms are created in Excel (XLS forms)
- http://xlsform.org/
- Automatic status messages via Google Cloud Messaging
- Tablet management needs to be improved

Data Collection Server

- Server in MOH an RMS Data Centres
- Ubuntu Linux operating system
- Full disk encryption and modern security
- Recently updated to enable real-time data. Now using docker.
- Runs ODK Aggregate software
- Java application running in Tomcat behind Nginx
- Daily backup or real time backup
- Meerkat Nest anonymises the data and sends it to a message queue (for real time)

Data Processing

This service starts with the raw anonymised data and provides aggregated data for display

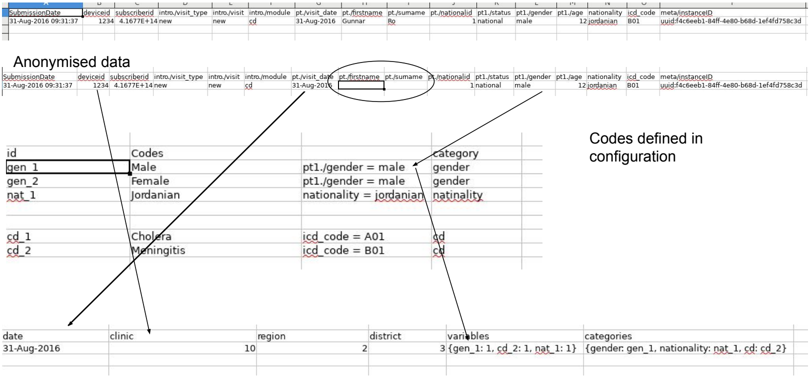
- Configurable codes are used to translate raw data into useful data
- These codes provide a flexible way of giving meaning to the data.
- Support: Matching, calculated values, number in interval, multiple conditions etc
- Data processing tools also deal with important meta data like locations and tablets.
- Data is localised by matching on device-ids or by gps
- Provide an API that gives access to data
- Supports a wide range of aggregated data views.
- Data processing has two components Abacus and API

Meerkat Abacus

- Python package that imports raw data and meta data and processes the data to the finished data table
- Reads data from message queue for real time (currently 10 m) soon 1m.
- Uses a PostgreSQL database for support of JSON
- When processing data alerts are identified and sent out
- Supports both single and threshold alerts
- Supports codes based on linking multiple form entries
- Data sources, locations, codes and links configurable
- https://github.com/meerkat-code/meerkat_abacus

Example

From tablet



SubmissionDate	deviceid	subscriberid	intro./visit_type	intro./visit	intro./module	pt./visit_date	pt./firstname	pt./surname	pt./nationalid	pt1./status	pt1./gender	pt1./age	nationality	icd_code	meta/instanceID
31-Aug-2016 09:31:37	1234	4.1677E+14	new	new	cd	31-Aug-2016	Gunnar	Ro	1	national	male	12	jordanian	B01	uuid:f4c6eeb1-84ff-4e80-b68d-1ef4fd758c3d

Anonymised data

SubmissionDate	deviceid	subscriberid	intro./visit_type	intro./visit	intro./module	pt./visit_date	pt./firstname	pt./surname	pt./nationalid	pt1./status	pt1./gender	pt1./age	nationality	icd_code	meta/instanceID
31-Aug-2016 09:31:37	1234	4.1677E+14	new	new	cd	31-Aug-2016			1	national	male	12	jordanian	B01	uuid:f4c6eeb1-84ff-4e80-b68d-1ef4fd758c3d

id	Codes		category
gen_1	Male	pt1./gender = male	gender
gen_2	Female	pt1./gender = male	gender
nat_1	Jordanian	nationality = jordanian	natinality
cd_1	Cholera	icd_code = A01	cd
cd_2	Meningitis	icd_code = B01	cd

Codes defined in configuration

date	clinic	region	district	variables	categories
31-Aug-2016	10	2	3	{gen_1: 1, cd_2: 1, nat_1: 1}	{gender: gen_1, nationality: nat_1, cd: cd_2}

Final proccessed data

Databases in the Meerkat System

Many databases are used in the Meerkat system. We use PostgreSQL for all.

- ODK Aggregate database - Database for ODK aggregate, follows ODK schema. Raw data
- Nest database - **Truth database** - raw data and anonymised data
- Persistent database on reporting server
- Main meerkat database - Processed data - “recalculated” on each startup
- Notifications and authentication databases

Current structure designed for cloud solution. Re-engineering needed?

Meerkat Api

- Python package that provides an HTTP rest API to query data
- Based on Flask
- Used Pandas for some of the data aggregations
- Supports aggregation over time and location
- Mapping of cases, incidence rates and clinics
- Supports many reports
- Supports explore data functionality
- Supports downloading the raw data using a Celery as task runner
- For the API only the processed data is used
- https://github.com/meerkat-code/meerkat_api

Meerkat Frontend

- The website you can see at iers.moh.gov.jo
- Written in Python and HTML, JS, CSS
- Uses many open source libraries. E.g bootstrap and Highcharts
- Includes public homepage and then a private technical page
- Technical page includes data on demographics, morbidity, completeness, PIP and alerts
- Very configurable, can change the number and content of the tabs
- The technical site also includes download data, explore data and reports
- https://github.com/meerkat-code/meerkat_frontend
- **Meerkat Hermes**: Separate microservice to sends email and SMS notifications and administer notifications

Authentication

- The frontend supports many different user accounts and access levels
- Each component and each tab can have different access levels.
- Authentication is implemented as a separate microservice
- Written in Python and HTML, JS, CSS
- Uses encrypted JSON webtokens to communicate access rules
- We have user accounts and roles
- Completely flexible access role network
- https://github.com/meerkat-code/meerkat_authentication

Example

Example on whiteboard

Documentation

All Meerkat components are documented to various degrees. The documentation will be in the handover and can be access at : http://meerkat-docs.s3-website-eu-west-1.amazonaws.com/

Software review

- Operating system: Linux (mainly ubuntu)
- Containerisation: Docker
- Python, Flask, Celery, SqlAlchemy
- JS (gulp, highcharts, bootstrap, jquery), HTML, CSS
- Message queues: RabbitMQ
- Databases: PostgreSQL, Redis
- Version control: git

Important programming libraries

Python:
Flask : micro-framework for web programming
SqlAlchemy: Database abstraction
pandas: Data analysis
shapely: GIS

Javascript:
jQuery: DOM manipulation
bower: asset management
gulp: task runner

Meerkat Infrastructure

- Use docker for containerisation for both production and development
- A github repository for every country and a demo country
- Data collection server is physical server in MoH and RMS
- Website moving to local server in MOH this week
- We use Travis for automatic testing

Technical Debt from moving to local server

- Many new features that have not been tested very well. Will lead to **instability**
- Development website. It is very useful to have a development version of the online framework where any changes can be tested before being deployed
- Improve and customise utilities and tools.
- Downtime for deployment. Any deployment will take 3 hours. The site will not be working proplery during this.
- There is no redundancy for high availability. If there are any hardware failures for any of the servers there are no replacements ready to be used.
- The infrastructure is now more ad-hoc and requires manual setup.

Training

What do you want to cover?

- **Linux and Docker**
- **Creating forms**
- **PostgreSQL Database**
- Python Programming
- Git
- Web programming
- Data analysis
- Meerkat Storage Solutions
- Linux networking and security
- Meerkat System Excersises

Introductions and program

Today:

Introduction and system architecture
Discussion about training goals
Forms
Lunch
Linux Training

Tomorrow:

Debugging running Meerkat System