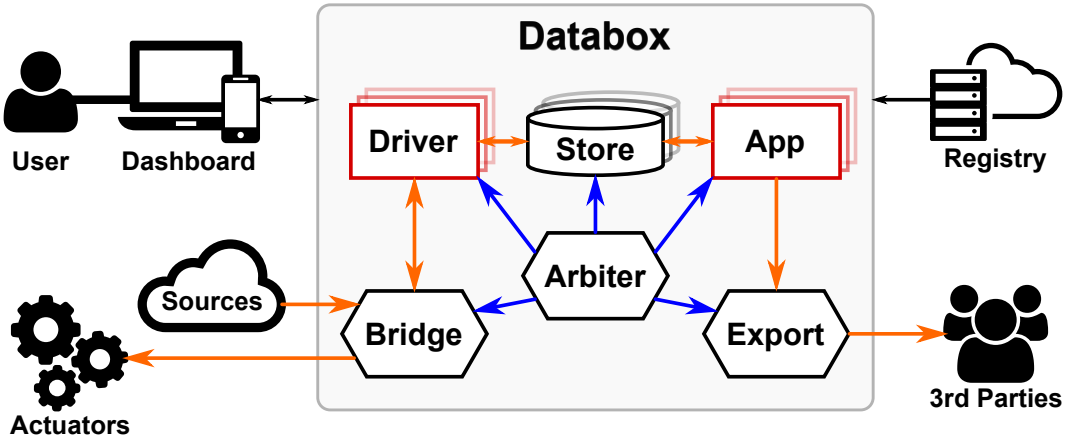
Databox Development

Community Launch 2017

The Databox Team

2017-03-24

Recap
Architecture



What makes an App?

App Manifests

- Docker containers
- Are launched with a set of environment variables
- Connected to certain components
- Deployed with an app *manifest*

```
{
  "manifest-version": 1,
  "name": "databox-app",
  "version": "0.1.0",
  "description": "An app",
  "author": "Yousef",
  "license": "MIT",
  "tags": [ "app" ],
  "packages": [{
    "name": "Package",
    "purpose": "",
    "install": "required",
    "risks": "",
    "benefits": "",
    "datastores": [ "DS_demo" ]
  }],
  "datasources": [{
    "type": "demo",
    "required": true,
    "name": "demo",
    "clientid": "DS_demo"
  }],
  "resource-requirements": {
    "store": "databox-store-blob"
  }
}
```

What makes an App?

The Dockerfile

- Standard Dockerfile contents
- Databox type label
- UI port exposure
- See template apps

```
FROM node:alpine

COPY . .
RUN npm install

LABEL databox.type="app"

EXPOSE 8080

CMD ["npm","start"]
```

What makes an App?

Environment Variables

- DATABOX_LOCAL_NAME
- ARBITER_TOKEN
- CM_HTTPS_CA_ROOT_CERT
- HTTPS_SERVER_PRIVATE_KEY
- HTTPS_SERVER_CERT
- DATASOURCE_[clientid]
- URLs for containers to connect to:
  - DATABOX_ARBITER_ENDPOINT
  - DATABOX_EXPORT_SERVICE_ENDPOINT
  - [STORE_NAME]_ENDPOINT
- Other less important ones in documentation

```
{
 "item-metadata": [{
  "rel": "urn:X-hypercat:rels:hasDescription:
      en",
  "val": "Mobile phone light sensor"
 }, {
  "rel": "urn:X-hypercat:rels:isContentType",
  "val": "text/csv"
 }, {
  "rel": "urn:X-databox:rels:hasVendor",
  "val": "Databox Inc."
 }, {
  "rel": "urn:X-databox:rels:hasDatasourceid",
  "val": "light"
 }],
 "href": "https://databox-driver-mobile-
     databox-store-blob:8080/light"
}
```

Inter-container Communication

Overview

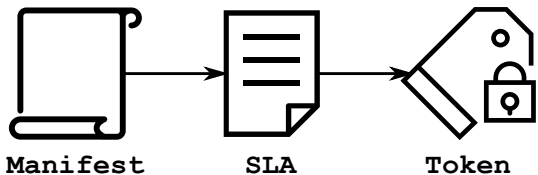
- All communication over HTTPS; certs generated run-time; system root CA
- RESTful APIs for all operations
- Direct mapping of HTTP methods to CRUD functions
- Per-route granular permissions
- Network-level isolation additionally
- Content Security Policy (CSP) to sandbox UIs

```
{
  "target": "smartphone-store",
  "path": "/accelerometer/ts/latest",
  "method": "POST"
}
```

```
{
  "target": "smartphone-store",
  "path": "/(sub|unsub)/gps/*",
  "method": "GET"
}
```

Inter-container Communication

Requesting Tokens



1. `GET DATABOX_ARBITER_ENDPOINT/token`
   - 1.1 `target`: The hostname of the target container that will verify the provided macaroon
   - 1.2 `path`: API path for which the token should be minted for
   - 1.3 `method`: HTTP method for which the token should be minted for
2. Encapsulated by libraries

Inter-container Communication

Reading/writing from/to Stores

1. `(GET|POST)[STORE_NAME]_ENDPOINT/[datasourceid]/[api]`
   - 1.1 `/kv`: Key-value
   - 1.2 `/ts`: Time series
     - 1.2.1 `/latest`: Last sample
     - 1.2.2 `/since`: Since a given timestamp
     - 1.2.3 `/range`: Between given timestamps
2. WebSocket subscription events
   - 2.1 Connect to `[STORE_NAME]_ENDPOINT/ws`
   - 2.2 Subscribe through
     `(GET) [STORE_NAME]_ENDPOINT/(sub|unsub)/[datasourceid]/[api]`
   - 2.3 Event emitted with datasource ID and data
3. Other enpoints documented
4. Encapsulated by libraries

Development Workflow

Setting up Dev Environment

1. Install Docker and Git
2. Git clone `me-box/databox`
3. Launch in dev mode through `./platformDevMode.sh` and follow instructions
4. Create Dockerfile and manifest (templates available)
5. Upload manifest to local manifest server (default `localhost:8181`)
6. Build Docker image and push to local registry (default `localhost:5000`)
7. Browse to dashboard interface and launch container (default `localhost:8989`)

—Demo—

Development Workflow

Iteration Cycle

1. Launch app as staging container (see guides)
2. Make changes
3. `docker cp` files into running container
4. `docker exec -it` build and run commands
5. Repeat

—Demo—

Thank you for your attention!

Questions?

More info: `http://www.databoxproject.uk/`
Contribute: `https://github.com/me-box`