Our Plan for the next 30 minutes

- What is Weaviate?
- What is the Contextionary that powers Weaviate?
- What can you do with weaviate?
- How have we built weaviate?
- Live Demo

What do we get out of it?

- Spark your desire to try it out
- Get Feedback from you
What is Weaviate?

- Contextual
- Decentralized
- Knowledge Graph

github.com/creativesoftwarefdn/weaviate
Introducing the Contextionary

FANCY
ROYAL
HAMBURGERS
CELEBRITY
MAN
THRONES
QUEEN
WOMAN
RULER
RICH
MONARCH
MALE
FEMALE
OPPRESSOR
WELL-SPOKEN
Contextionary: Other Examples

location vs. place vs. city

seal vs. seal
Decentralized

- Standalone
- Peer-to-peer network
  - own your data
  - enrich with other data
- Connect weaviates to combine their strengths (e.g. a fast dataset with a large dataset)
Knowledge Graph

● structured way to ask questions and receive answers (as you know from Google)
● building on existing graph technologies: Graph Databases, GraphQL
● adding more and more semantic tools to our infrastructure over time
What can weaviate be used for?

- Combine data across Industries without harmonizing ontologies
- Easily gain more insights into data you already own
- Enrich your data with publicly known, but previously inaccessible data
  - Find new customers
  - Fraud detection and other behavioral analyses
Example Queries

```
{
  Network {
    Fetch {
      Things(where: {
        class: [{
          name: "Plane",
          certainty: 0.8
        }],
        properties: [{
          name: "model",
          certainty: 0.8,
          operator: Equal,
          valueString: "777"
        }]
      } {
        beacon
        certainty
      }
    }
  }
}
```
How is weaviate different from existing Graph Technology?

Weaviate does not replace, but enhance existing graph technologies.

Example: Should I use weaviate or a graph database directly? How do they differ?

- Ease of use (GraphQL vs. SQL/Gremlin/Cipher etc.)
- NLP and Contextionary
- Modularity of Datastore (CAP-Theorem, pick what you need)
  - Janusgraph, Neo4j, RedisGraph, ...
Architecture - what is weaviate made of

- User-facing APIs: REST and GraphQL
- Microservice
  - small concern, 12-factor, docker and kubernetes native
- Written in Golang 1.11 (yeah, modules!)
  - proven to be a very good language in the cloud environment
  - great compromise between stability, ease of use and cloud performance
  - API design first -> go-swagger
- Focus on Modularity & Pluggability
  - Examples next slides
Focus on Modularity

Example: Database Connector

- Anything that implements a DatabaseConnector interface is a database connector
- The remainder of the app is database-agnostic
- First connector we have built: Janusgraph (with C* and ES)
- Get started with FooBar connector

```go
// LocalGetClass resolves a GraphQL request about a single Class like so
//
// `{ Local { Get { Things { City { population } } } } }`
//
// Where "City" is the particular className of kind "Thing". In the example
// above the user asked to resolve one property named "population". This
// information is contained in the Params, together with pagination and filter
// information. Based on this info, the foobar connector can resolve the
// request. It should resolve to a []map[string]interface{} that can be consumed
// by the respective resolver in graphqlapi/local/get.
//
// An example matching the return value to the query above could look like:
//
// []interface{}
//   map[string]interface{}
//     "population": 1000000,
//   },
//   map[string]interface{}
//     "population": 600000,
// },

func (f *FooBar) LocalGetClass(info *get.Params) (interface{}, error) {
  return nil, nil
}
```

https://github.com/creativesoftwarefdn/weaviate/blob/999fc9a4146dfba44803c3f862e3a25951a100dc/database/conn
ectors/foobar/connector.go#L356-L379
Focus on Modularity
Example: Authentication and Authorization

- Inspired by K8s
  - good separation between AuthN and AuthZ
- AuthN highly pluggable
  - Basic Auth
  - OpenID Connect
  - ... let’s see what our users use
  - anything that can decided between authenticated “yes/no” and provide a username and/or group to the AuthZ plugin
- AuthZ
  - Role-based Access
    - (e.g. [“read”, “write”] on [“Things”, “*”])
- full proposal available at
  https://github.com/creativesoftwarefdn/weaviate/issues/628
Who’s behind weaviate and what do they offer?

- Enterprise support / Consulting
- Advanced Networking abilities
- Operate own weaviate with helpful datasets (e.g. Wikipedia-based, SMB directory)
- Custom contextionaries (industry-specific)
- “Playground” user interface
# Roadmap

<table>
<thead>
<tr>
<th>Completed</th>
<th>In Development</th>
<th>In Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST API</td>
<td>Network Fuzzy Requests</td>
<td>Natural Language Interface on top of GraphQL</td>
</tr>
<tr>
<td>GraphQL API Local</td>
<td>AuthN/AuthZ as proposed</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Get/GetMeta</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contextionary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Janusgraph Connector</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Feedback and Contact Info

Feedback areas

General Feedback - How useful can weaviate be to you?

API design - Ease of use vs. Abilities?

Connectors - Abstraction vs. Specific features?

Contact SeMI

semi.network
Reach out to David or Micha

About me

Etienne Dilocker
Core Developer Weaviate
dilocker.de
GitHub/Twitter: etiennedi
YouTube: kubucation