CRATE

florian@crate.io

Out of Big Austrian Mountains,
we have created a database for Big Data

Founded in
2013 in
Dornbirn/Austria

Offices in Dornbirn,
Berlin, London, Los Angles,
San Francisco

Team of 14 People
(with and without
strong Austrian dialect)

Won Techrunch
Disrupt
startup battlefield

# What does Crate do?

CRATE

- It is a massively scalable persistence solution
- It combines: storage, search and data analysis

- It is built on a NoSQL architecture
- It is extremely simple to install and operate
- It is super fast and provides powerful search capabilities

- Horizontally scaling
- Elastic scaling
- Resilient
- Read-after-write consistency

# Explosion of Database Solutions...

CRATE

## Wide Column Store / Column Families

- Hadoop / HBase API
- Cassandra
- Hypertable API
- Accumulo
- Amazon SimpleDB Misc
- Cloudata
- MonetDB

## Document store

- Elasticsearch
- MongoDB
- Couchbase Server
- CouchDB
- RethinkDB
- IBM Cloudant

## Key value / Tuple Store

- DynamoDB
- Azure Table Storage
- Riak
- Redis
- BerkeleyDB
- Oracle NoSql Database

## Graph Databases

- Neo4J
- ArangoDB
- Infinite Graph
- GraphBase
- FlockDB

## Multimodel Databases

- ArangaDB
- OrientDB
- Datomic
- CortexDB
- WonderDB

## Object Databases

- Versant API
- db4o
- Gemstone/S
- VelocityDB
- ObjectDB
- CoreObject

## Grid and Cloud Database Solutions

- Oracle Coherence
- Gigaspaces
- Gemfire
- Hazelcast
- ObjectDB
- **Crate Data**


CRATE
We are drowning in data
and starved for knowledge

CRATE

World's data is more than a Zeta Byte!

# 1,000,000,000,000,000,000,000 Bytes

(That's 135GB of data for every person on Earth)

Most data now lives outside of Corporate Data Centers!

Data will be double over the next 4 years!

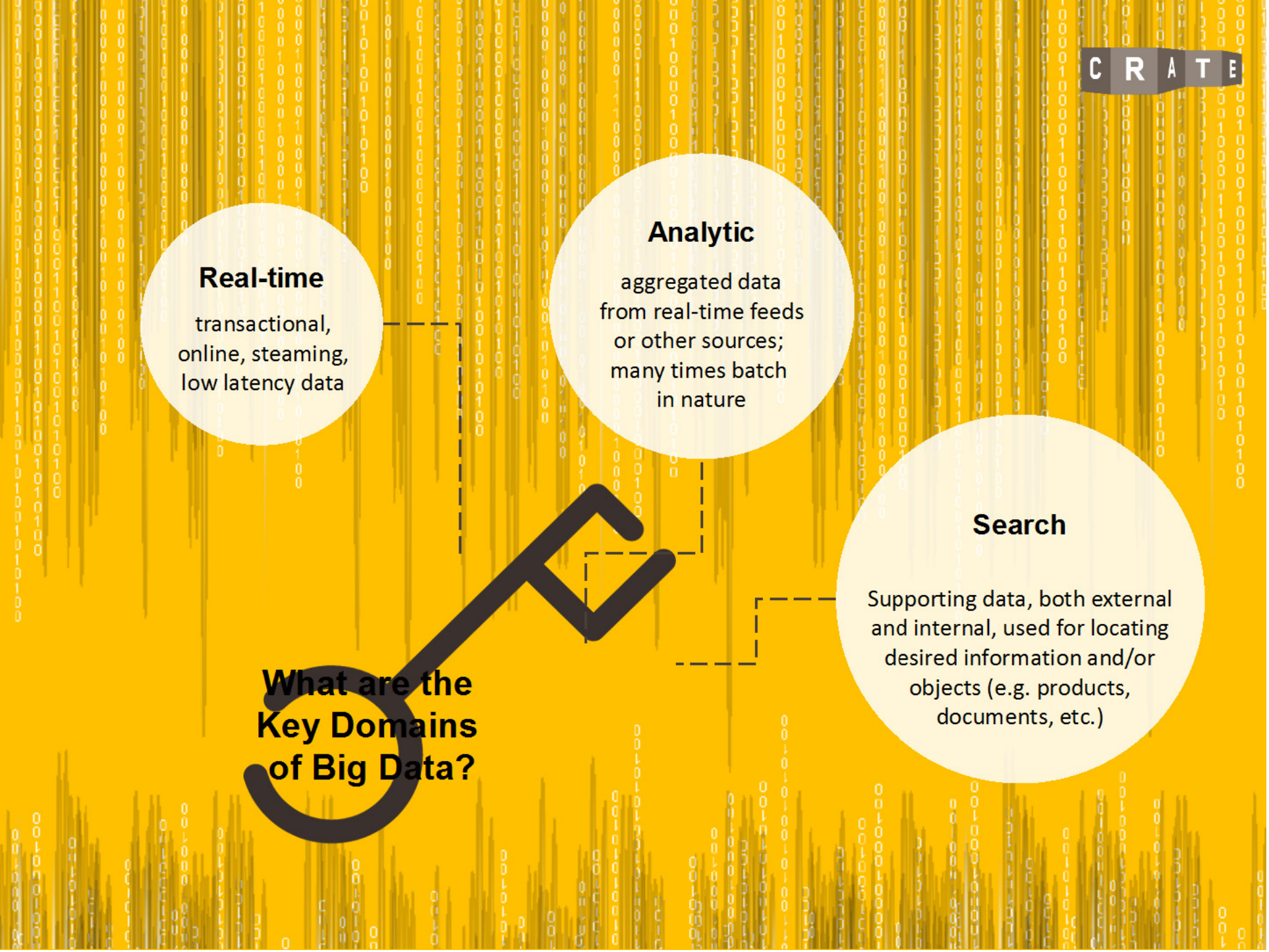
**How much data are you using in your organization ?**

Source: http://www.slideshare.net/DataStax/getting-big-value-from-big-data?next_slideshow=1

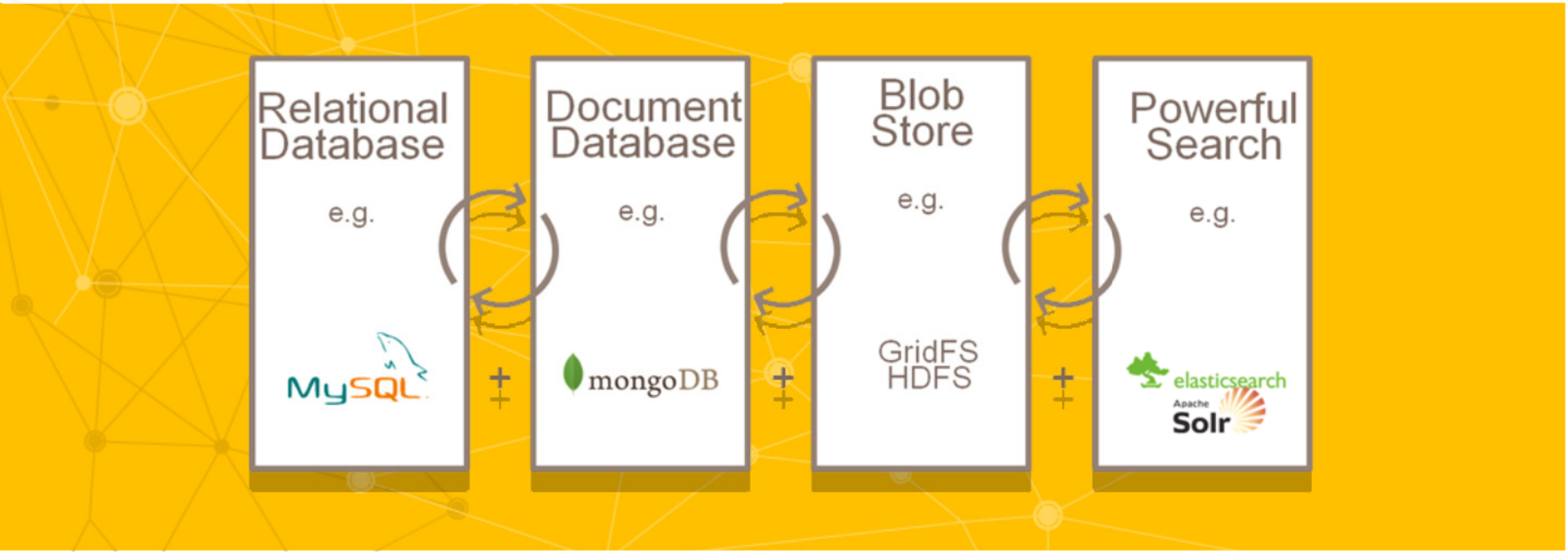
CRATE

## What is Big Data?

| | |
|---|---|
| “traditional” data | **BIG DATA** |
| gigabytes to terabytes | PETABYTES TO EXABYTES |
| centralized | DISTRIBUTED |
| structured | SEMI-STRUCTURED AND UNSTRUCTURED |
| stable data model | FLAT SCHEMAS |
| known complex interrelationships | FEW COMPLEX INTERRELATIONSHIPS |


CRATE
Real-time
transactional, online, steaming, low latency data
Analytic
aggregated data from real-time feeds or other sources; many times batch in nature
Search
Supporting data, both external and internal, used for locating desired information and/or objects (e.g. products, documents, etc.)
What are the Key Domains of Big Data?

- Often Big Data requirements cannot be covered by any of the traditional SQL databases
- ...and NoSQL databases have problems covering them too!
- People already tried complex solutions by gluing/fixing/patching together different technologies like:
  - Riak + Solr + Rados
  - MongoDB + Elasticsearch + GridFS
  - CouchDB + Elasticsearch + HDFS/Hadoop



CRATE

## Before Crate

- We had to compromise on performance if we wanted to keep the ease of use benefits of using SQL stores,

  or

- We had to move to a No-SQL store and deal with the complexities of the query languages and rewriting parts of the backend source code

## Having Crate,

**you get the best of both worlds**

- The No-SQL performance you require

  and

- The SQL syntax you want.

Crate can be used for a variety of use cases, from a classic but scalable SQL database, to advanced usage incorporating full text search, geo shape and analytic support.

# Getting Started...

```
$ aws ec2 run-instance –image-id ami84562dec –instance-type m1.medium –key-name dev

$ ssh –I ~/.aws/keys/dev.pem ubuntu@54.167.184.22

$ bash –c "$(curl –L install.crate.io)"

http://54.167.184.22:4200/_plugin/crate-admin/#/

$ sudo apt-get install python-pip

$ sudo pip install crash

cr> \connect  127.0.0.1:4200

cr> create table test(created_at timestamp, id string primary key, user_id string);

cr> insert into test values (1394182938,'3', 'John')

cr> select * from test where id='2';

http://54.167.184.22:4200/_plugin/crate-admin/#/
```

# Create Table Statement

Getting more complex: a table clustered into 6 shards.

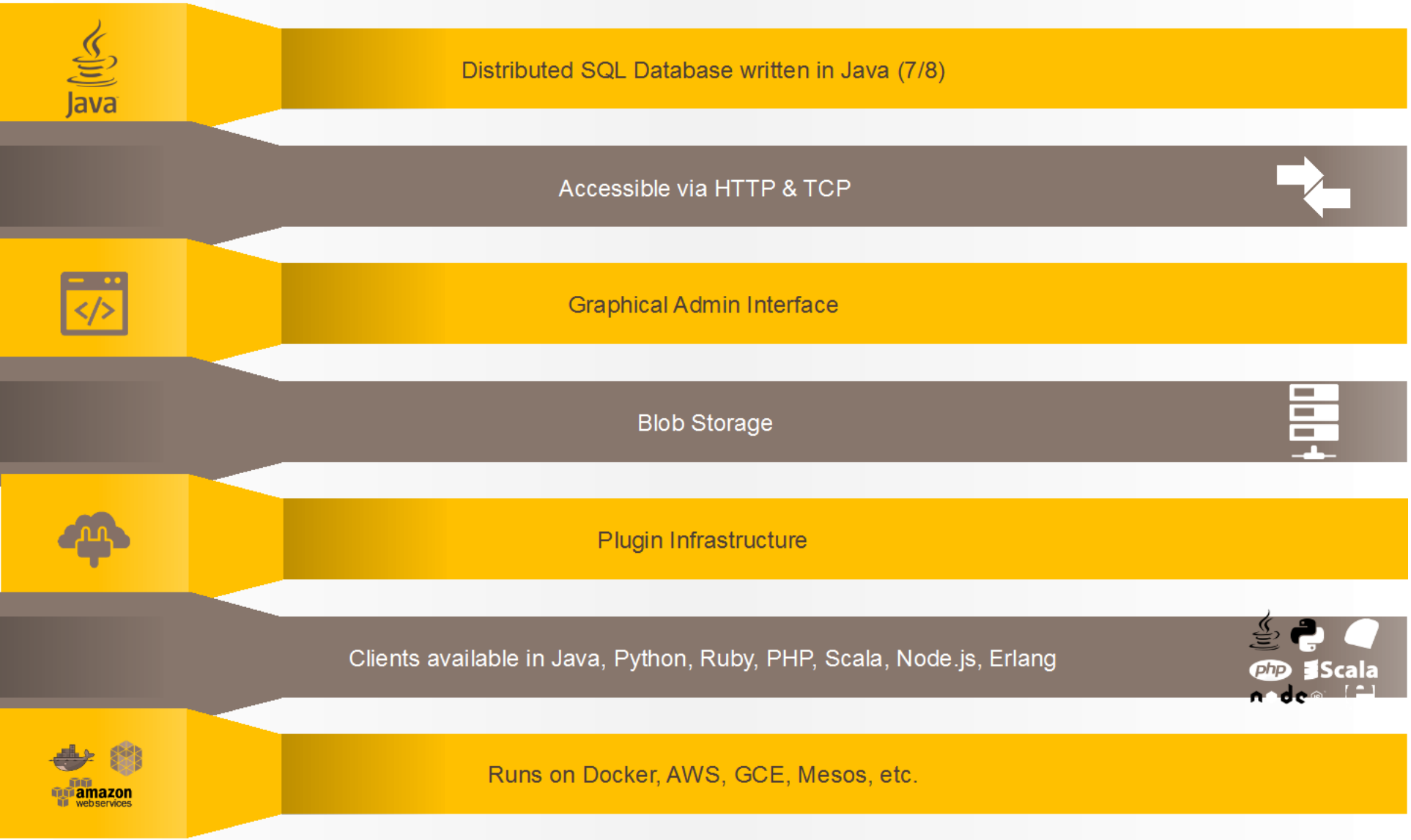
```
cr> CREATE TABLE user (
      id INT primary key,
      name STRING,
      day_joined TIMESTAMP primary key,
      bio STRING INDEX using fulltext,
      address OBJECT (dynamic) AS (
        city STRING,
        country STRING
      )
    ) CLUSTERED INTO 6 shards
    PARTITIONED BY (day_joined)
    WITH (number_of_replicas = 2 );
```

```
cr>  COPY user FROM 'home/ubuntu/users.json';
```

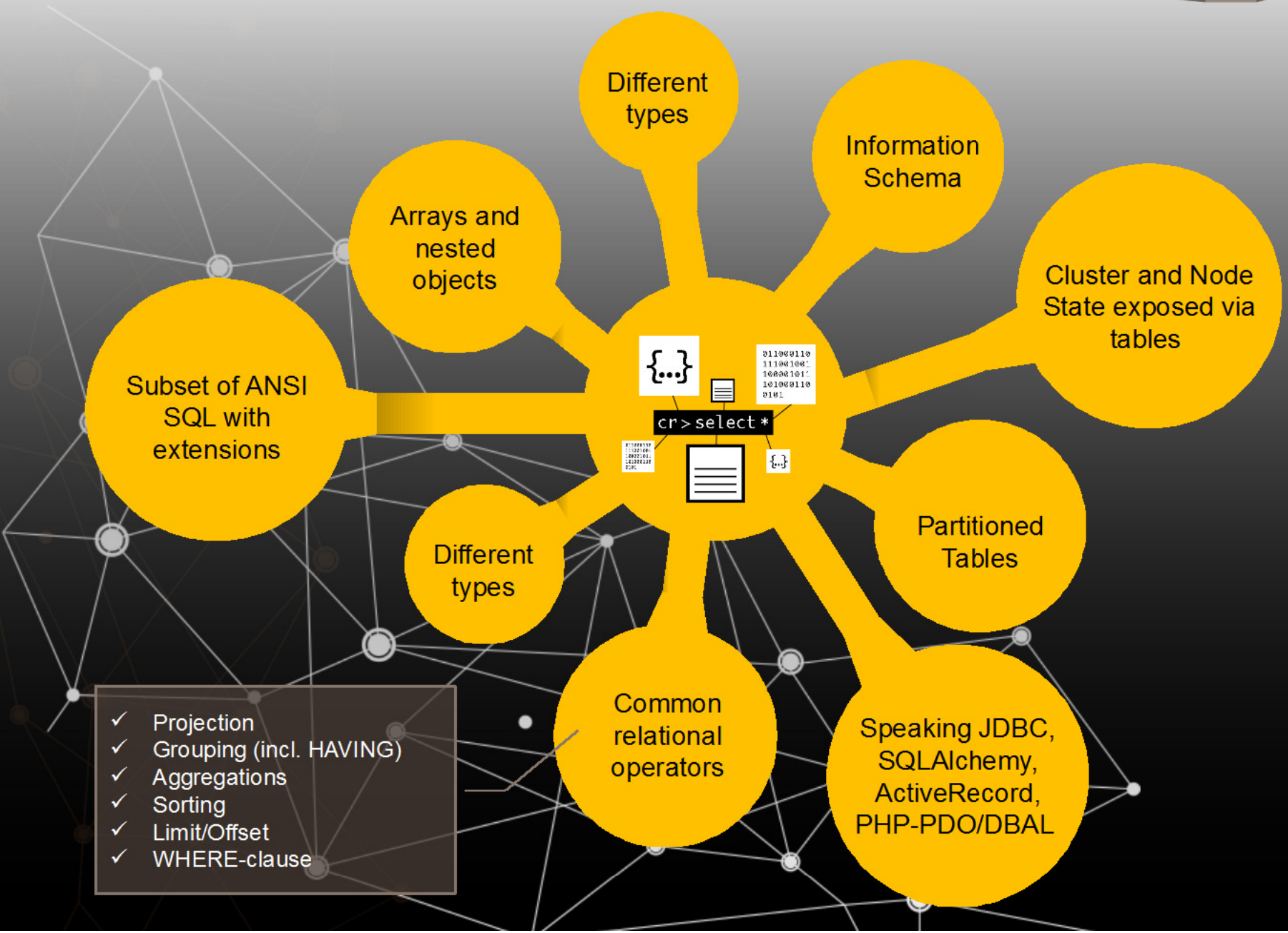
```
1 {"id": 3, "name": "foo", "day_joined": 1408312800, "bio": "Lorem ipsum
  dolor sit amet, consectetuer adipiscing elit.", "address": {"city":
  "Dornbirn", "country": "Austria"}}
2 {"id": 4, "name": "bar", "day_joined": 1408312800, "bio": "Lorem ipsum
  dolor sit amet, consectetuer adipiscing elit.", "address": {"city":
  "Berlin", "country": "Germany"}}
```

Please read Appendix_1_quick_hands_on_crate.docx for more details.

# Core Features

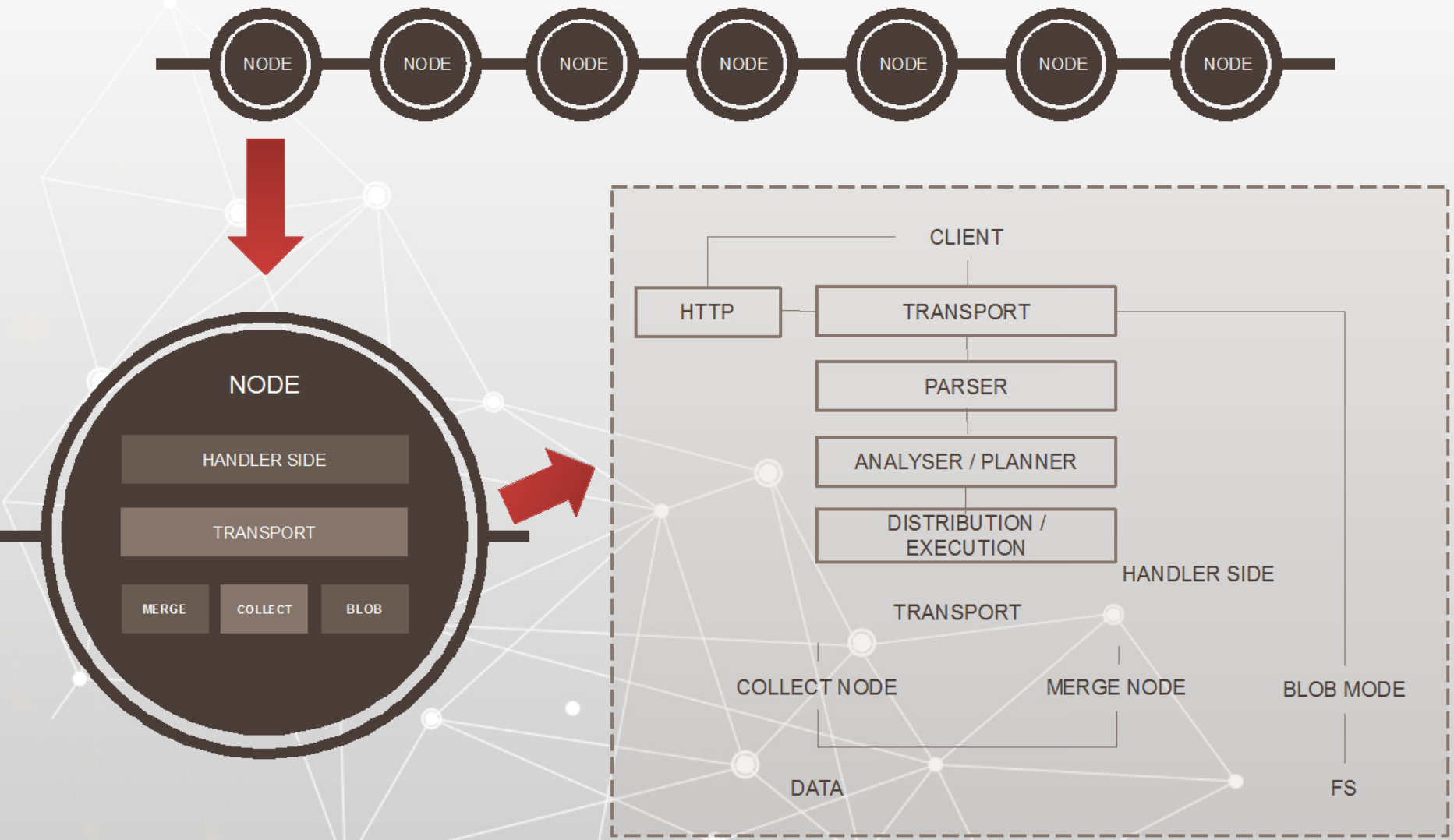


# Core Features




- ✓ Projection
- ✓ Grouping (incl. HAVING)
- ✓ Aggregations
- ✓ Sorting
- ✓ Limit/Offset
- ✓ WHERE-clause

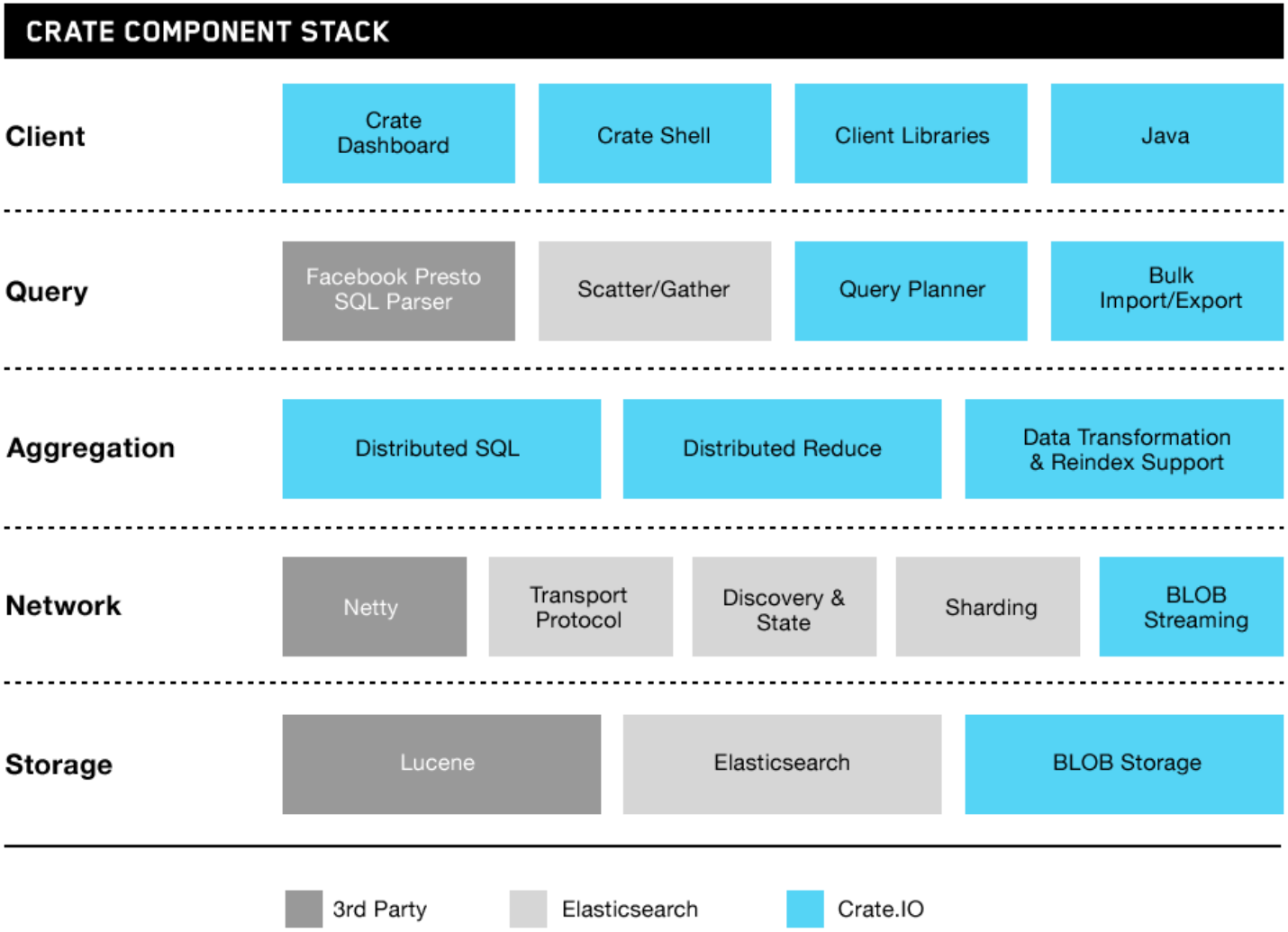
# Architecture




Nothing shared – Equal nodes – Massively scale

# Component Stack

Built on top of trusted components CRATE adds critical functionality wrapped into one downloadable package.



# Technical Highlights

Easy to scale open source solution, based on Crate Clusters

Real time SQL data store for developers

Based on share-nothing architecture

**Developers**

Can store tabular data, unstructured records and binary objects as well as query and search live data with SQL

Supports a super-easy rapid install on commodity hardware or cloud environments with high availability included

# Technical Highlights

CRATE

### Scalability

- Volume of data – Crate handle from GB range to TB range, even PB – a level of Billions records stored.
- Inserts: Up to 300k records per second
- Shards can be moved live easily, either manually or automatic, between nodes.
- Partitioning provides an additional level of granularity on top of sharding.
(e.g a table might be partitioned by date)

### Semi-Structured records

- Dynamic schemas provides a perfect balance between strong SQL schema and schema-less NoSQL
- Tabular approach, with the possibility of nested documents via array and object fields.
- Nested fields are first-class citizens and therefore have no restrictions compared to normal fields.
- Columns can hold objects and therefore remove the need of normalization.

### Fast SQL queries

- Crate planner uses the most performant internal strategy for each query automatically.
- Using collect/ shuffle/ reduce phases for data aggregation allow as distributed queries as possible
- Intelligent usage of node memory to speed up queries
- Non-blocking IO and asynchronous execution is natively built in and needs no spawning of OS

# Open-source is, of Course, the Right Choice!

Crate is licensed under the Apache License, Version 2.0

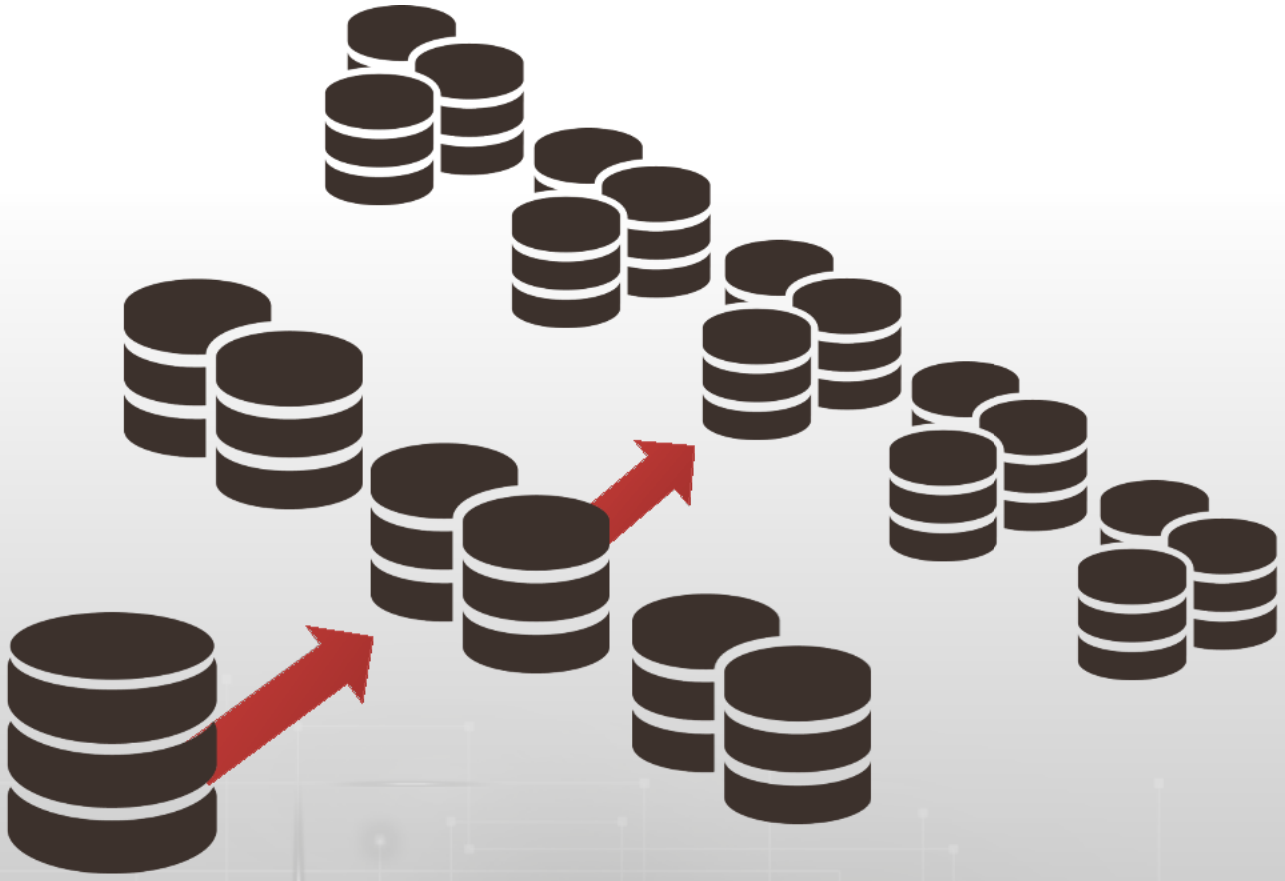
Entire source code can be found here: https://github.com/crate

Crate uses numerous third-party Open Source Libraries. You always can find the most recent and complete list of libraries in the NOTICE file of our corresponding Github Repository. O list f third-party libraries: https://crate.io/legal/notice

Developers who want to use any Crate.IO trademark or service mark in their promotional, instructional or marketing material or make references to our material online or in hard copy need to follow these guidelines: https://crate.io/legal/trademark

# Towards Distributed Databases

Create makes running distributed databases super easy

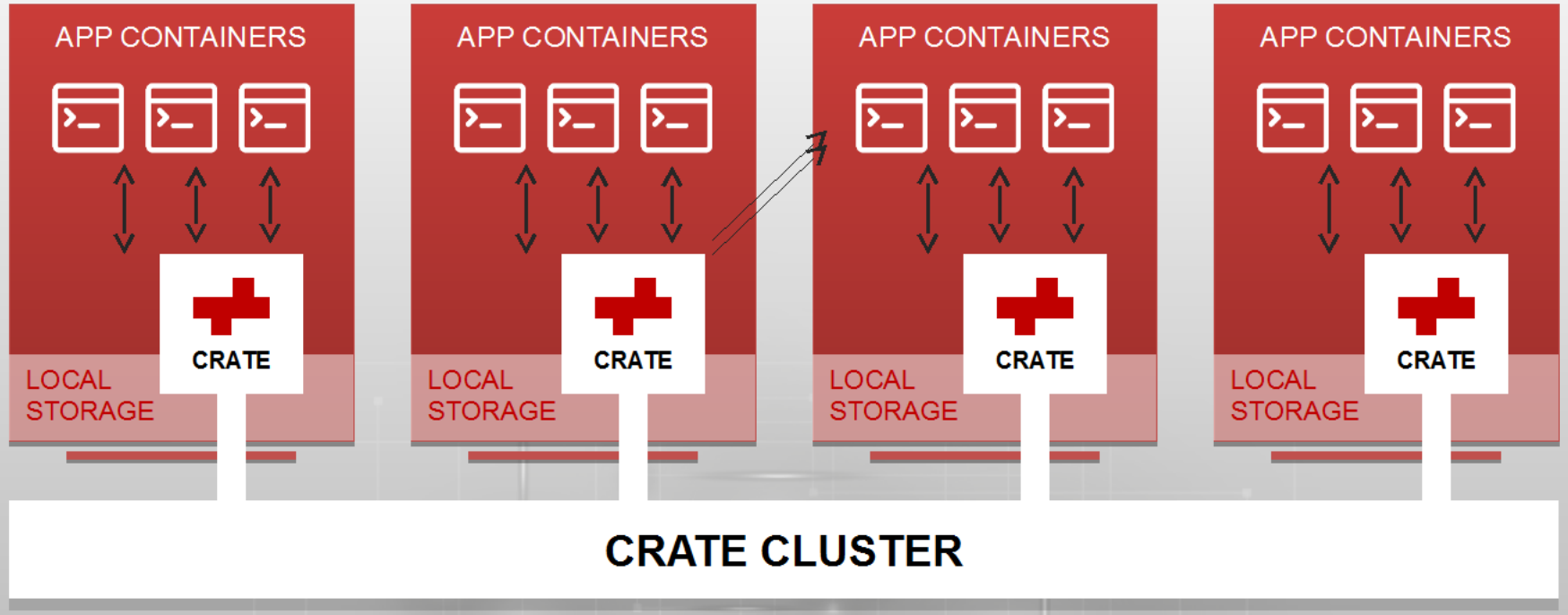


Just like these guy are doing it!

# Towards Distributed Databases

Crate is a distributed data store. Install Crate directly on your application servers and make the monolithic centralized database a thing of the past. Crate takes care of synchronization, sharing, scaling, and replication even for mammoth data sets. If a node becomes unavailable, Crate self-heals and rebalances the cluster automatically.

All nodes in a Crate cluster are equal, a shared-nothing architecture that makes configuration easy.



# Running Crate in Containers

ContainerShip allows you to configure a hosting stack by combining your own Docker images with their library of software and third party integrations

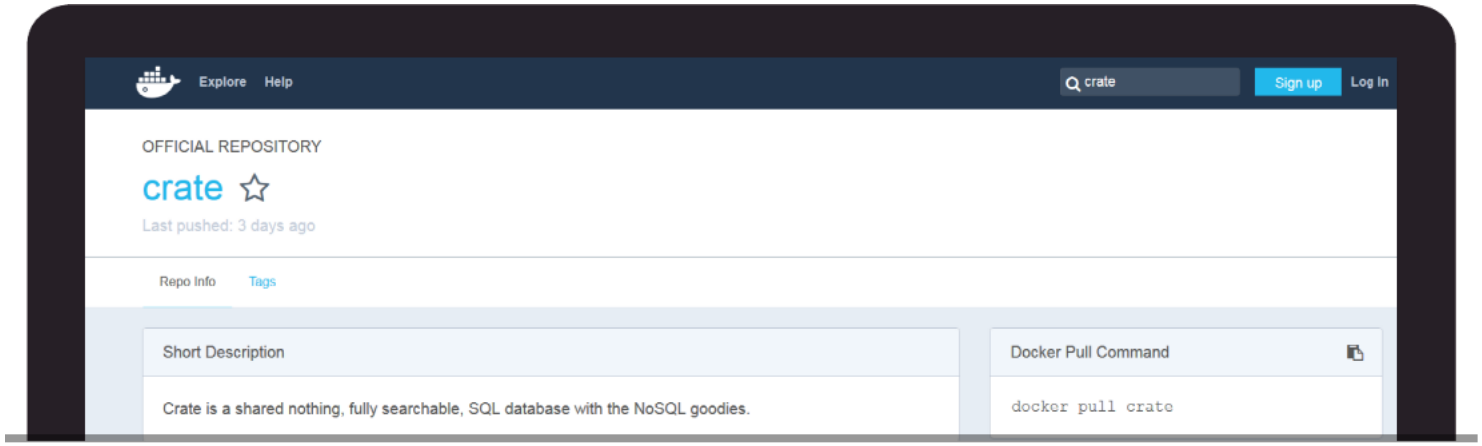
As a Linux distribution targeted at cloud and cluster based environments, CoreOS is a prefect fit for a Crate.IO cluster. We assume you have a CoreOS cluster running and can access the fleetctl tool installed on your local system or on a node in the CoreOS cluster.

Docker allows developers to package applications and dependencies into standard, shippable units, great for easy deployments across platforms and constantly changing needs. Crate is available as a Docker image and is a perfect data store for Docker based applications.

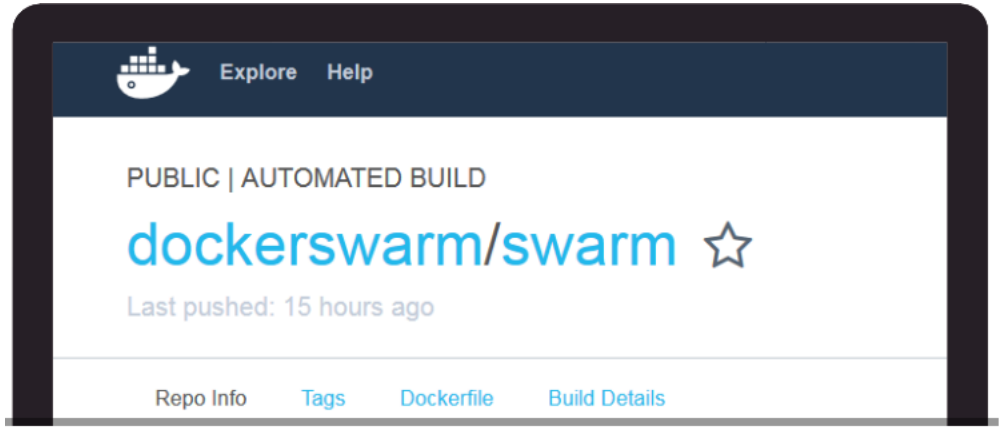
Tutum is platform for building, managing and deploying Docker containers across a variety of cloud providers and a provides many features ideal for Development workflows.

# Running Crate in Containers

- Crate already available as Docker container

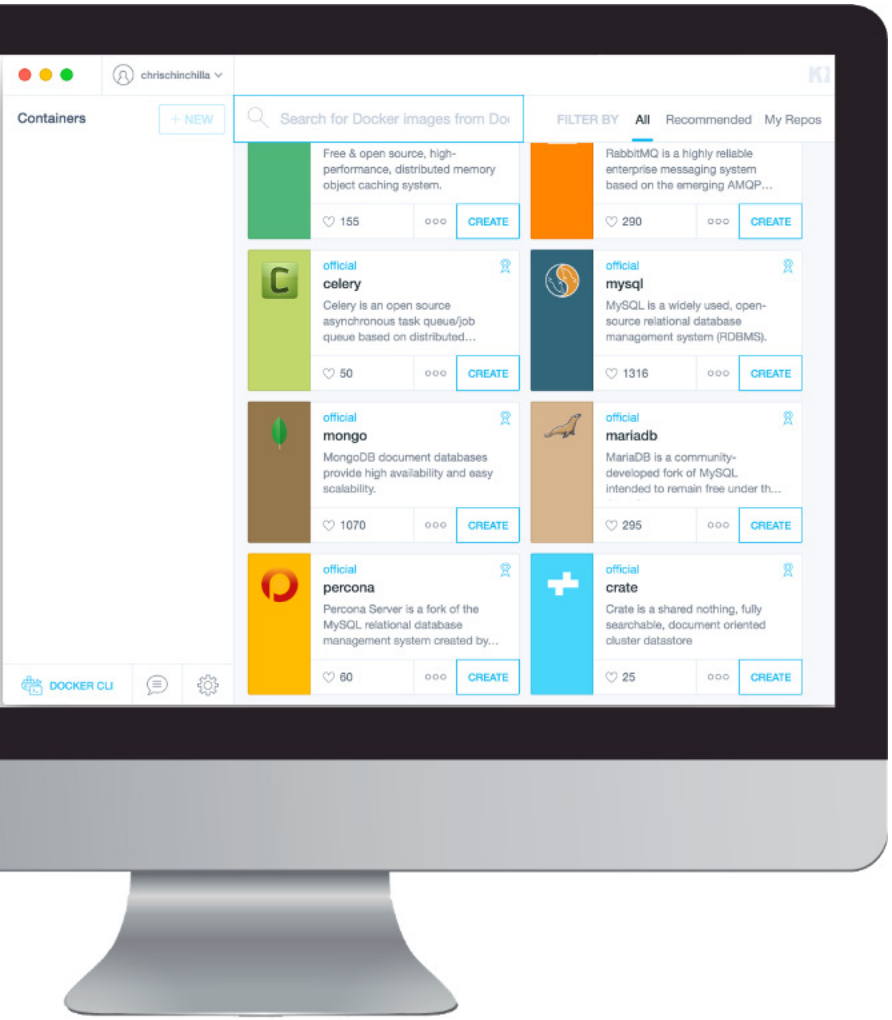


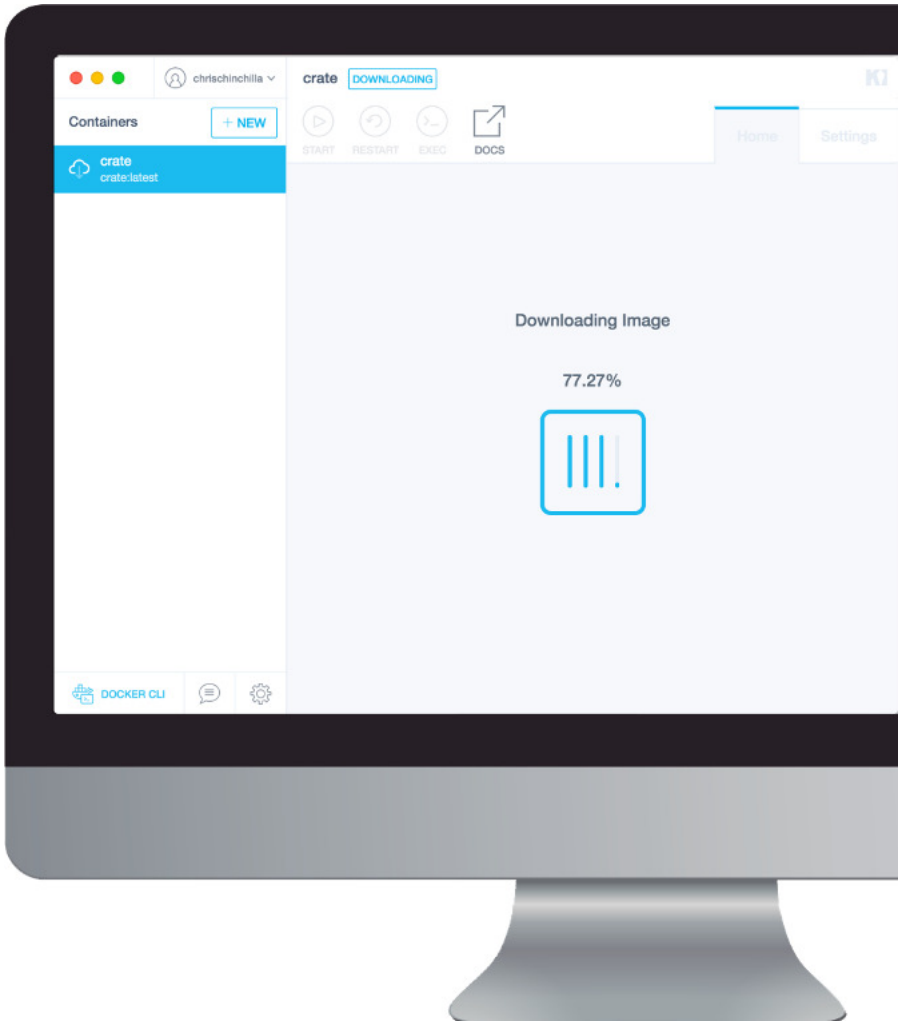

- Or creating a Crate cluster using Docker and Swarm



# Running Crate in Containers

- Crate with Kitematic - Kitematic is Docker's visual interface for creating Docker instances on Windows and Mac. Crate can be found in the list of default images in Kitematic.



# Running Crate in the Cloud

Crate offers a pre-configured AMI for Amazon AWS. With the Crate AMI, this includes dependencies, optimization and configuration to get Crate running as quickly and efficiently as possible. Using the AMI is our recommended method for running Crate clusters on AWS.

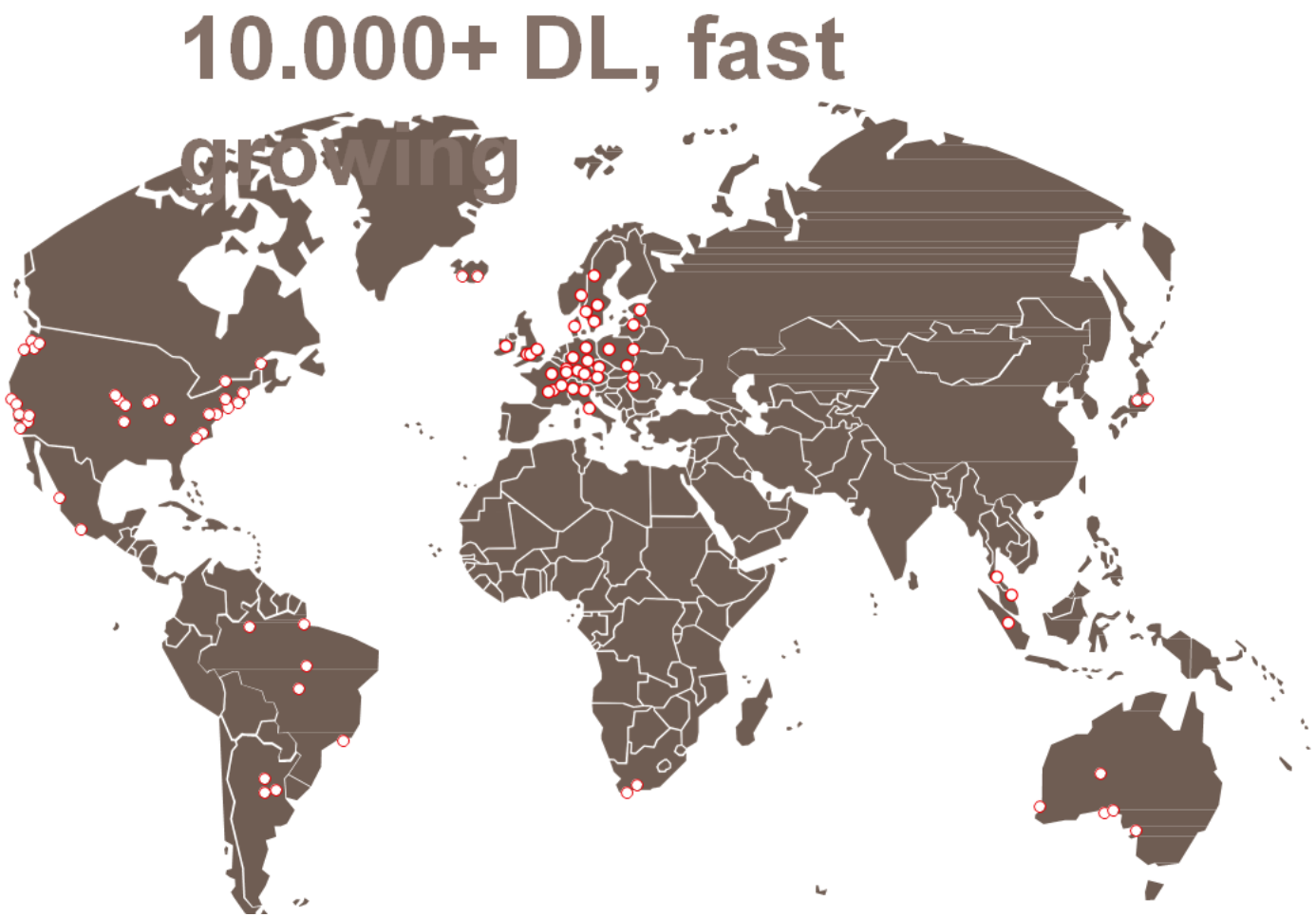
To create a cluster of Crate nodes on Azure cloud hosting we need to rely on unicast for inter-node communication.

Crate 0.50.2 will be deployed in a cluster of Compute Engine instances. Each instance will include a boot disk and separate data disk for data and logs. You can customize the configuration later when deploying this solution.

Crate uses SaltStack's Salt Cloud to launch and provision instances for the test cluster. Salt Cloud provides an abstraction layer for multiple cloud hosting providers, also for SoftLayer.

# Downloads Worldwide

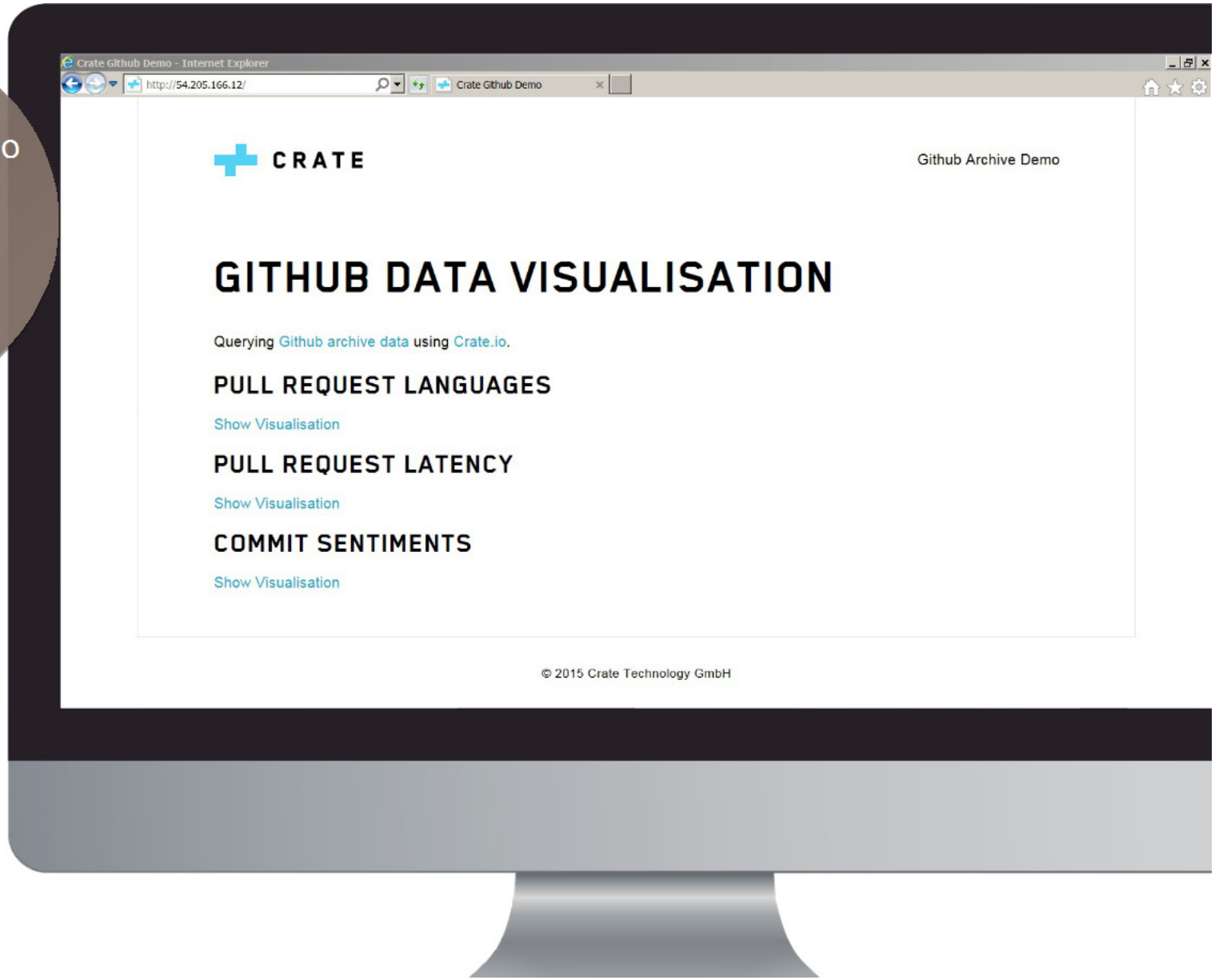
CRATE



# First real case demo – importing and analyzing GitHub data

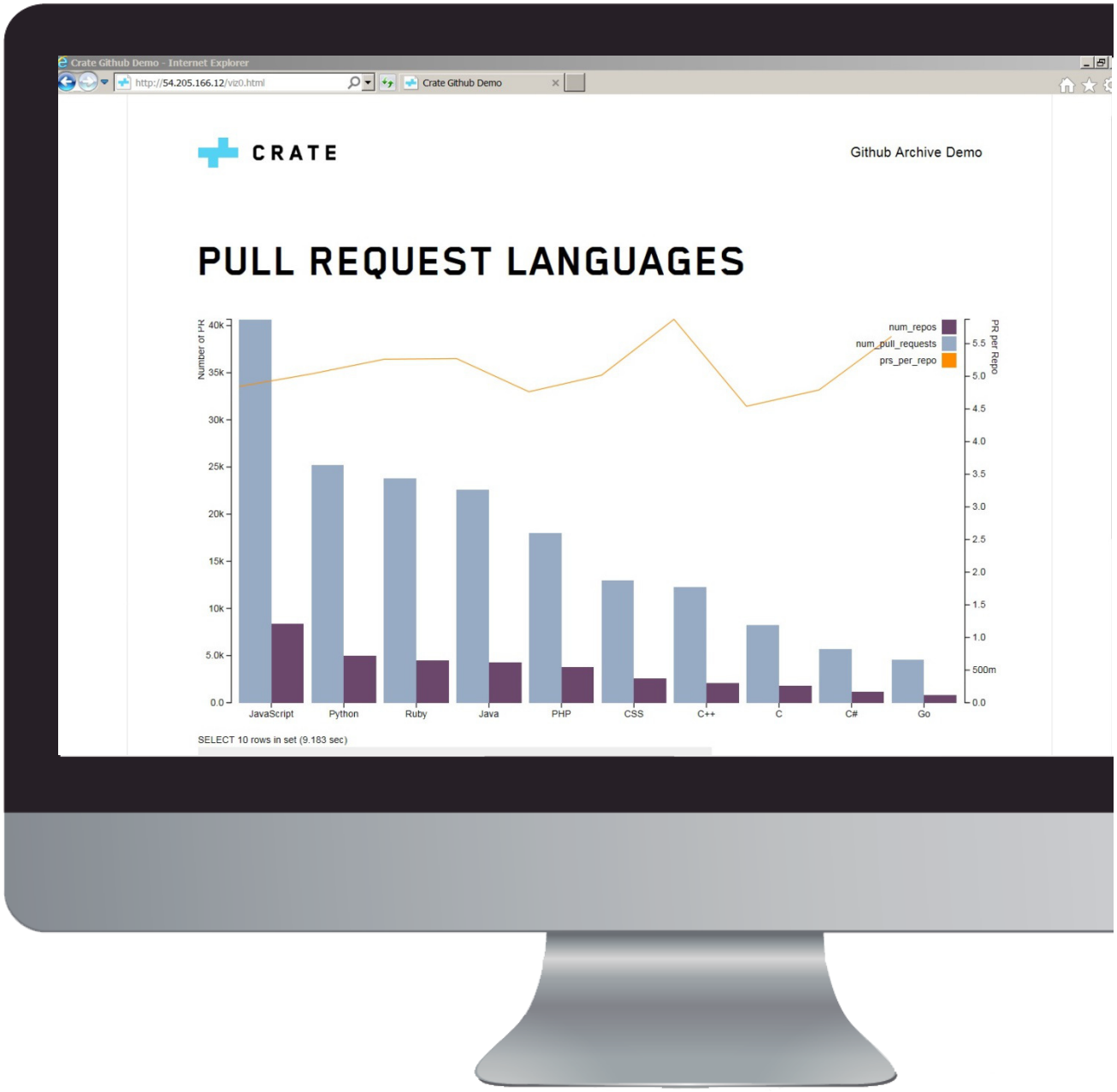
Here is the demo running:

**main page**



# First real case demo – importing and analyzing GitHub data

Here is the demo running: **pull request languages**



# First real case demo – importing and analyzing GitHub data

Here is the demo running: **pull request latencies**