# Big Data Analytics on Container-Orchestrated Systems

Gerard Casas Saez

University of Colorado Boulder

July 20th

# Outline

Introduction

Background

Problem statement

Approach

Implementation

Questions?

# Why?

# Keeping up with data growth

# Problem

- IOT & Social networks
- Internet traffic
  - Current: 72 petabytes/month
  - Prediction 2021: 232 petabytes/month.

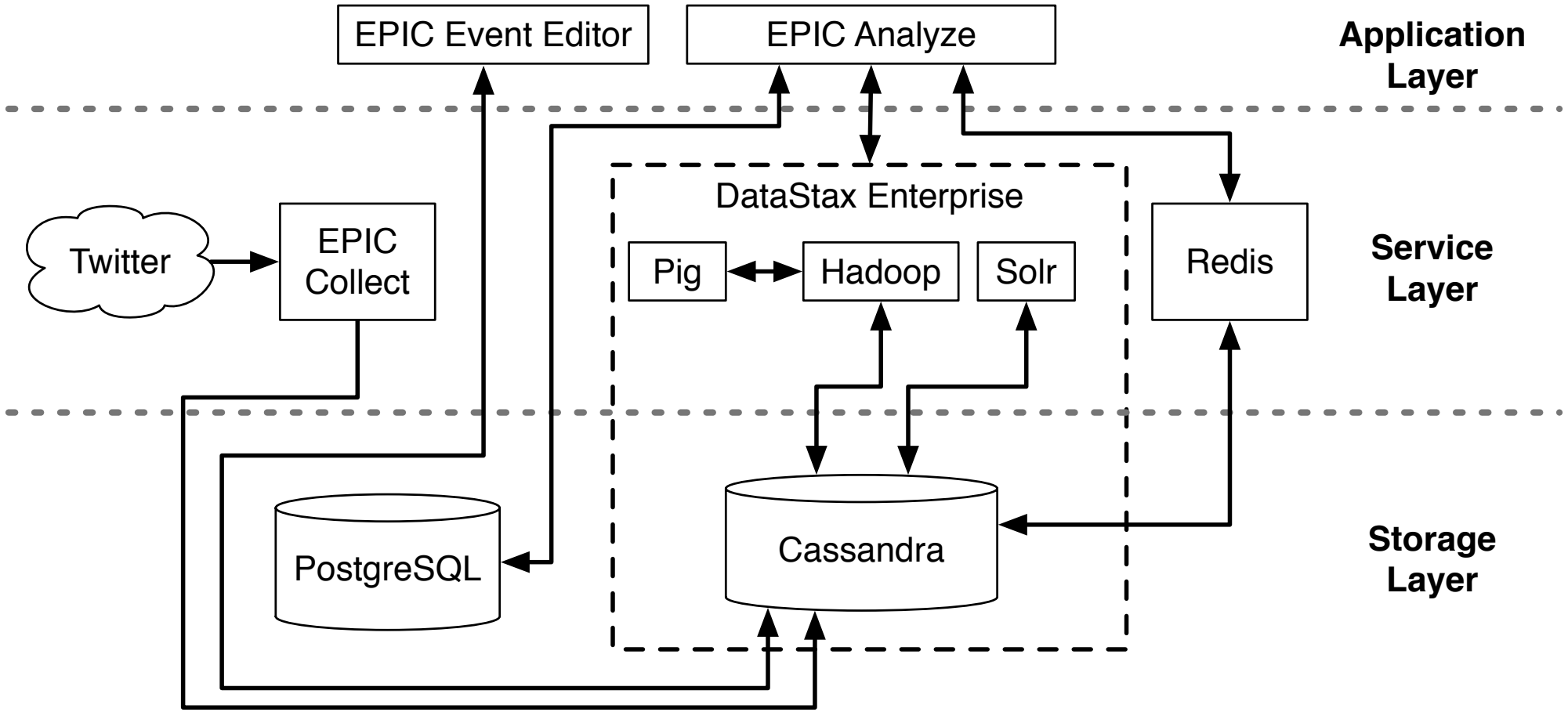
# Another problem

- Need to scale Big Data Analytics System
- Keeping mantainance at low cost
- Container-orchestrated make infrastructure easier
- Migrate Project EPIC architecture to container-orchestration system

# Background

# Project EPIC

- EPIC Collect
- EPIC Analyze


EPIC Event Editor
EPIC Analyze
Application Layer
DataStax Enterprise
Twitter
EPIC Collect
Pig
Hadoop
Solr
Redis
Service Layer
PostgreSQL
Cassandra
Storage Layer

# Containerization

- Operating-system-level virtualization
- Use host machine system resources
- **Docker** most used alternative
- Development microservices

# Container-orchestration systems

- Container interaction abstraction
- Great to deploy microservices architectures
- Apache Mesos vs **Kubernetes**

# Microservices Architecture

- Small & specific
- Better scalability
- Loosely-coupled & highly-cohesive
- Orchestration <> **Coreography**

# Coreography microservice architecture

- Easier to extend
- PubSub interaction
- Messaging system: **Apache Kafka**
- Asyncronous

# Problem statement

# Problem statement

1. Advantages and/or limitations from existing infrastructure
   - 1.1 More reliable?
   - 1.2 More scalable?
2. Lower maintenance costs than the existing infrastructure?
   - 2.1 Easier to deploy?
   - 2.2 Easier to upgrade?
   - 2.3 More resilient to failures?

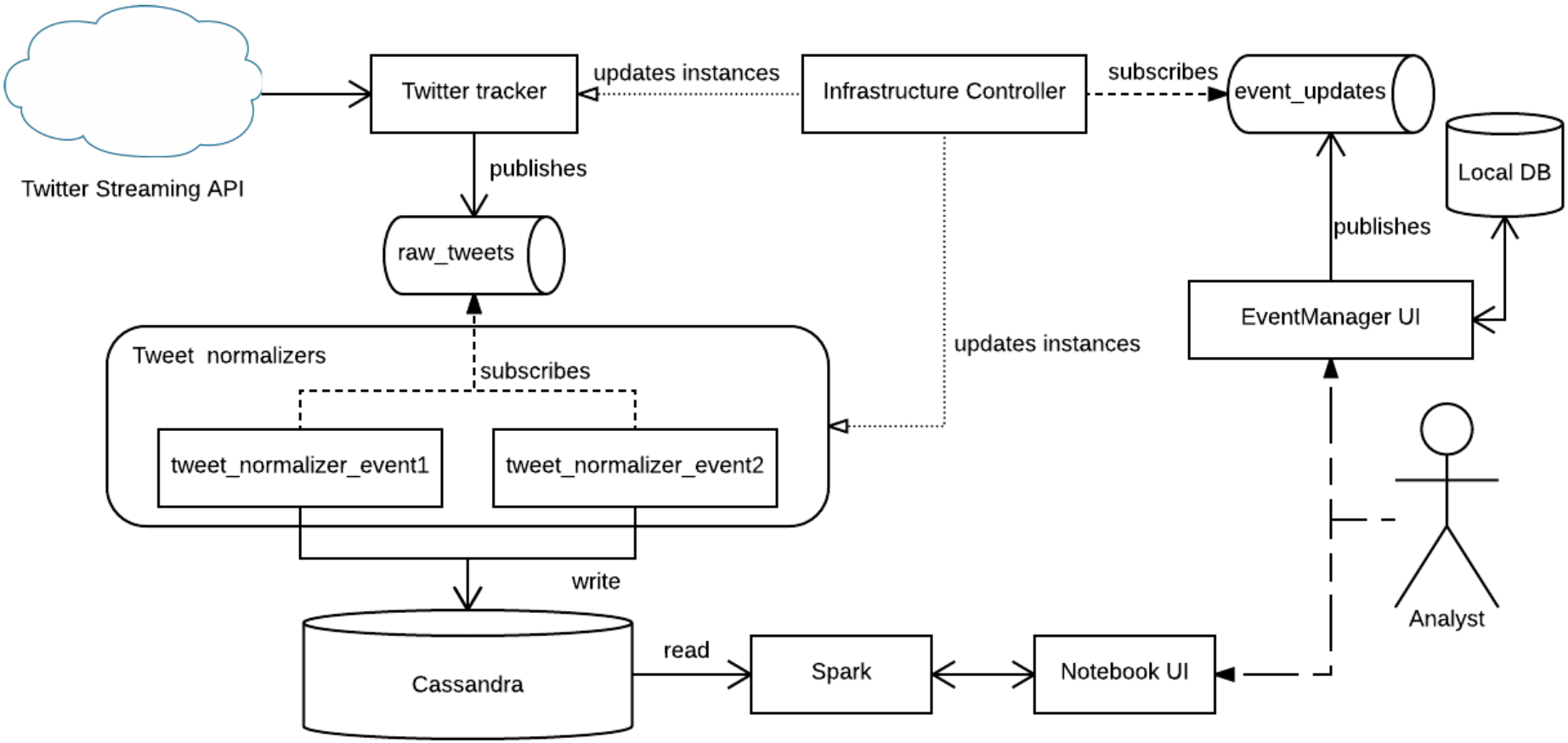
# Approach

# Features

- Event management
- Real-time collection of streaming Twitter data
- Real-time classification of incoming tweets
- Data Analysis

# Non-functional requirements

- Less code
- Easier deployment
- More flexible
- Better scalability


Twitter Streaming API
Twitter tracker
updates instances
Infrastructure Controller
subscribes
event_updates
Local DB
publishes
raw_tweets
publishes
EventManager UI
Tweet normalizers
subscribes
updates instances
tweet_normalizer_event1
tweet_normalizer_event2
write
Analyst
Cassandra
read
Spark
Notebook UI

# Demo time!

# Let’s track an event...

Event Manager UI

# ...and analyze it!

Zeppelin Notebook

# Questions?