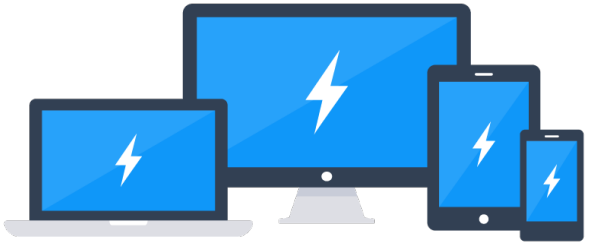
InnoGames

REAL-TIME MARKETING



with Kafka, Storm, Cassandra and a pinch of Spark

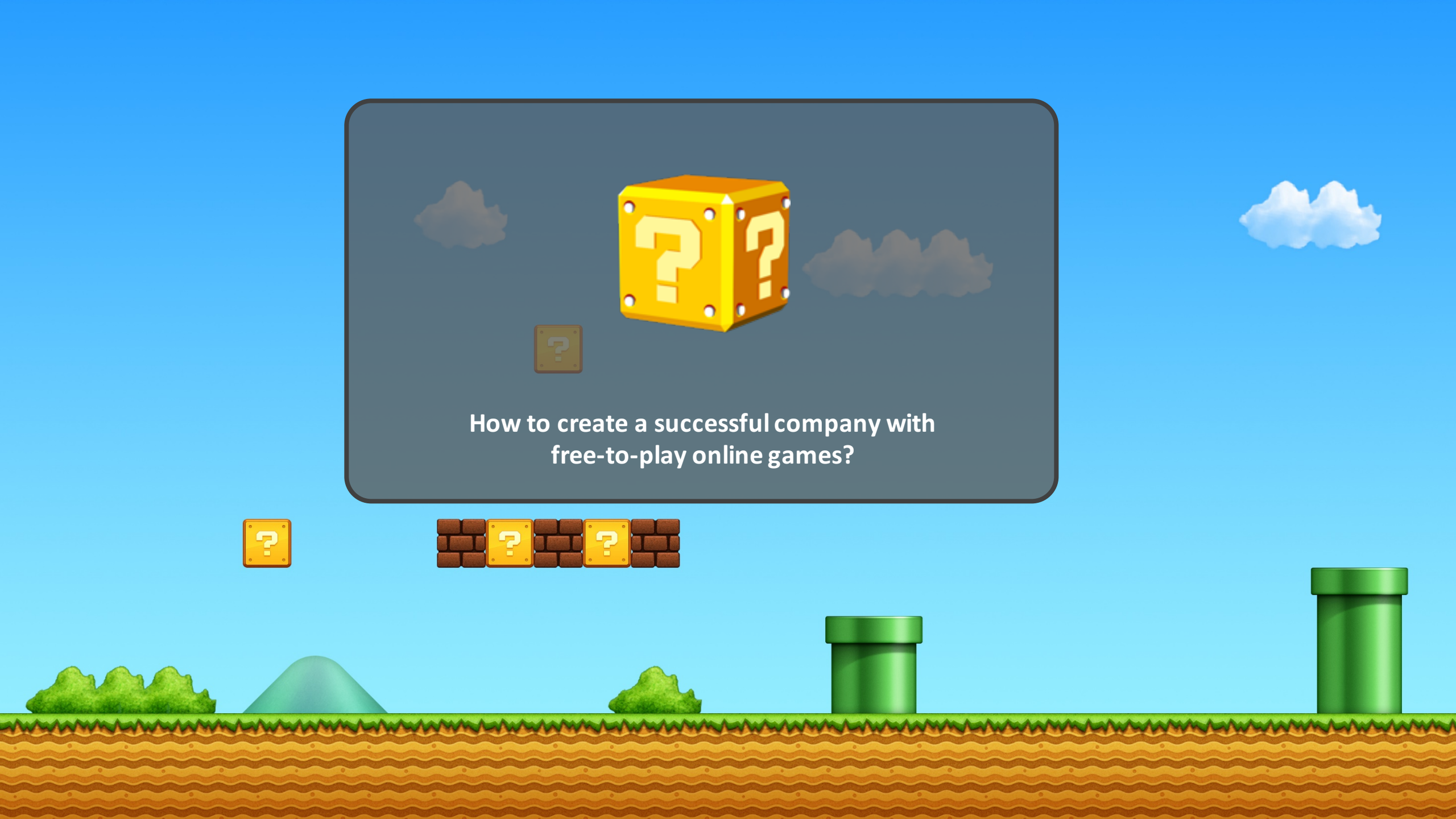
Volker Janz
Team Lead Software Developer

@prenomenon
linkedin.com/in/vjanz
xing.com/profile/Volker_Janz3

InnoGames is one of the worldwide leading developers and publishers of online games.

The company has over 160 million registered players from all over the world.

More than 400 people from over 30 countries are currently working in the offices in Hamburg and Dusseldorf.


How to create a successful company with
free-to-play online games?

Develop an awesome game

Get users

Sell virtual goods

Profit!

THX

</>

Develop an awesome game

Get users

Sell virtual goods

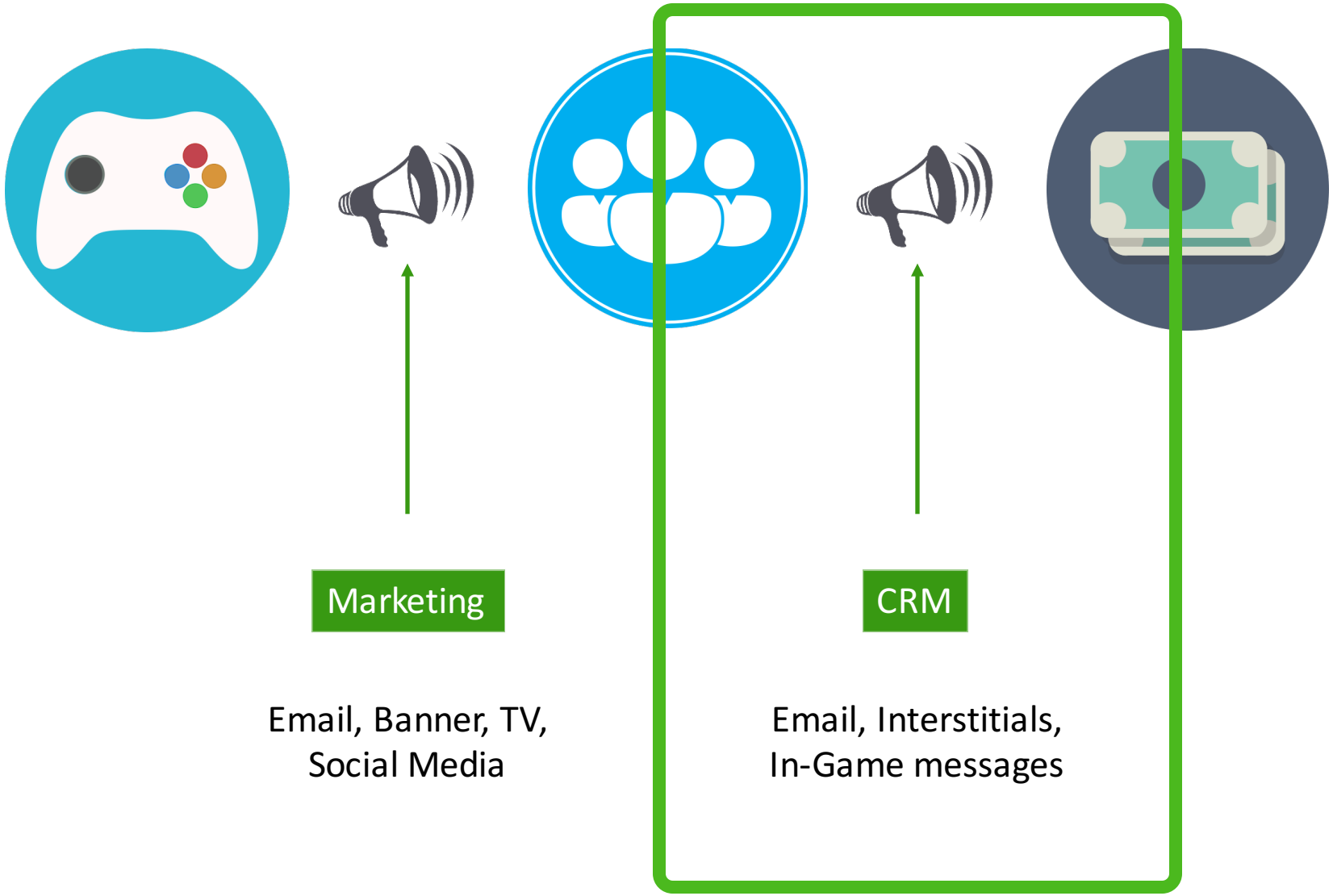
Profit!

Marketing

Email, Banner, TV,
Social Media

CRM

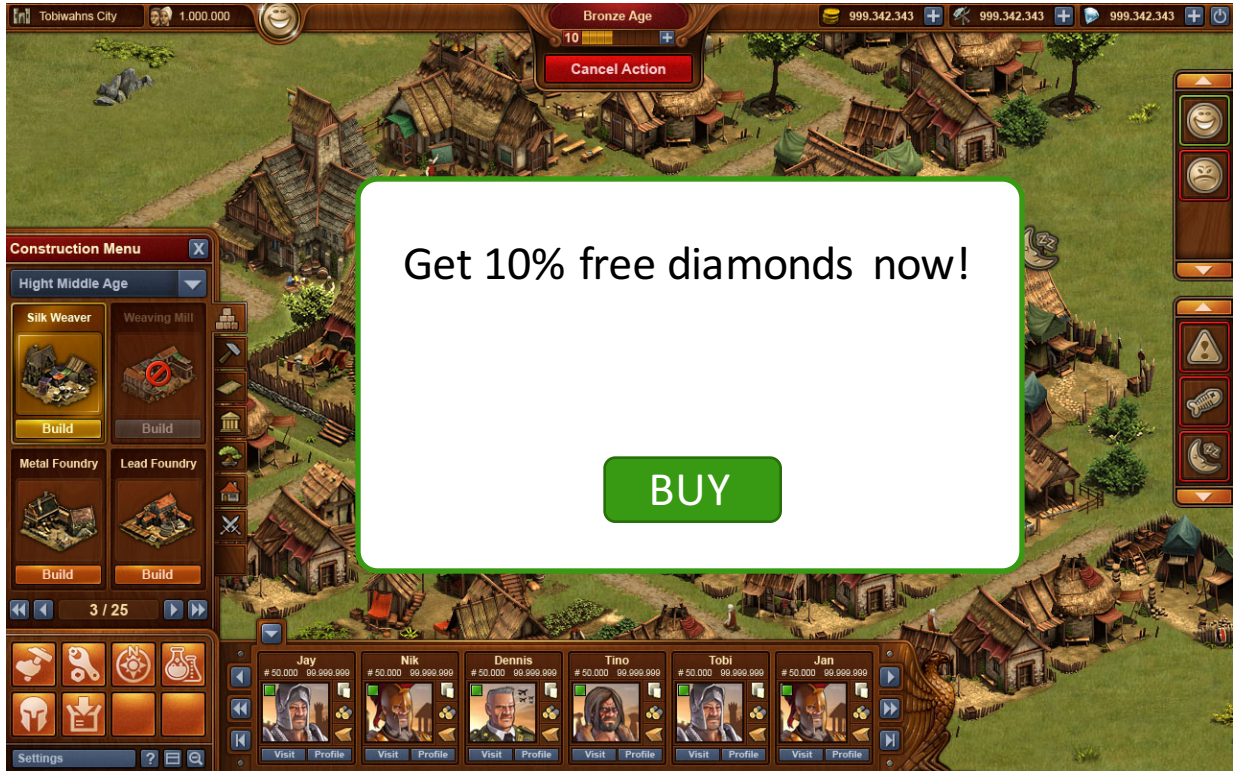
Email, Interstitials,
In-Game messages


Marketing
CRM
Email, Banner, TV,
Social Media
Email, Interstitials,
In-Game messages


Your Valentine's Day offer
For a short time only: Receive 10% more Crowns with your next purchase!
10% BONUS FOR YOU
CLAIM BONUS

This is Bob.
Bob plays Forge of Empires.

Bob gets an interstitial but it is not useful for him.



Bob continues to play the game. Now he is already losing a battle for the third time in a row.

Bob is frustrated but a few seconds later he gets a useful offer.

Bob finally wins the battle and continues to play the game – a churn was prevented.

REAL-TIME CRM

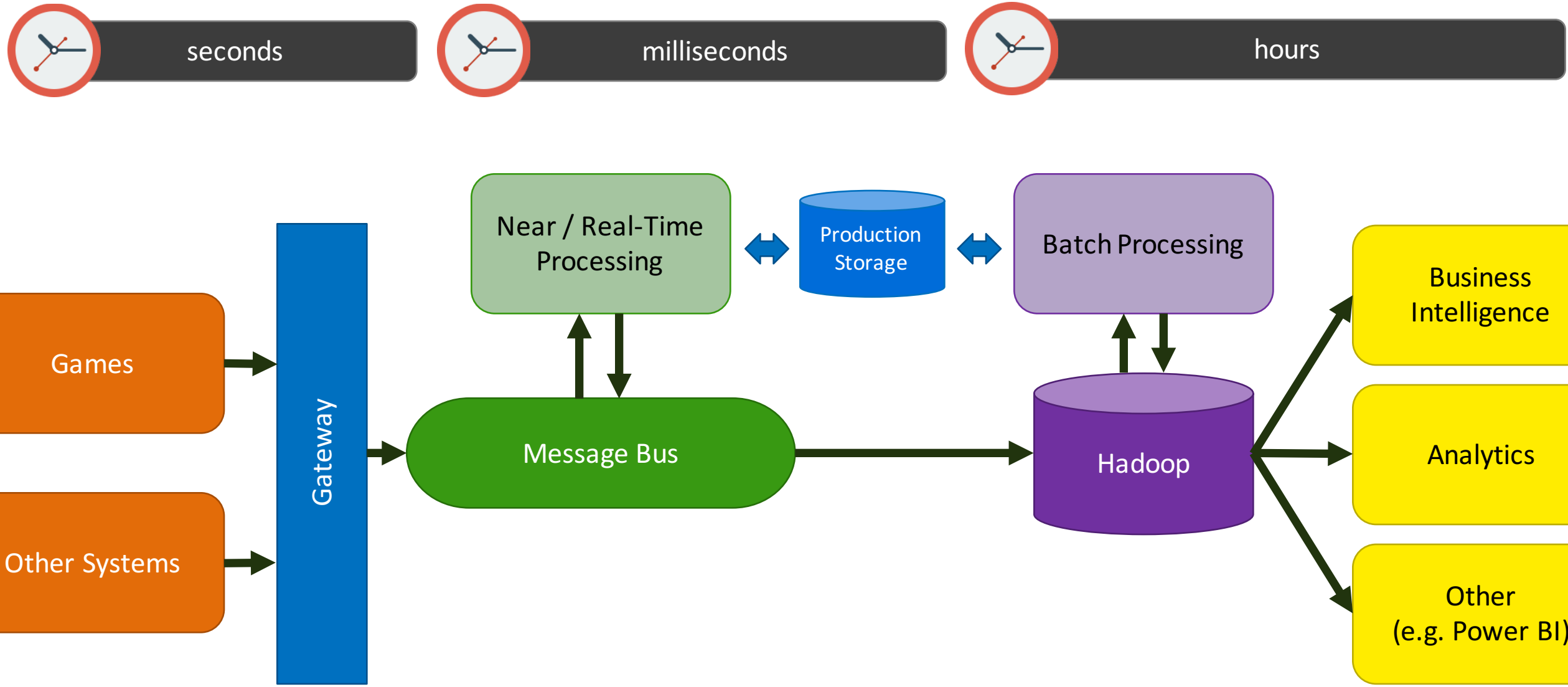
Individual marketing based on user behavior in real-time

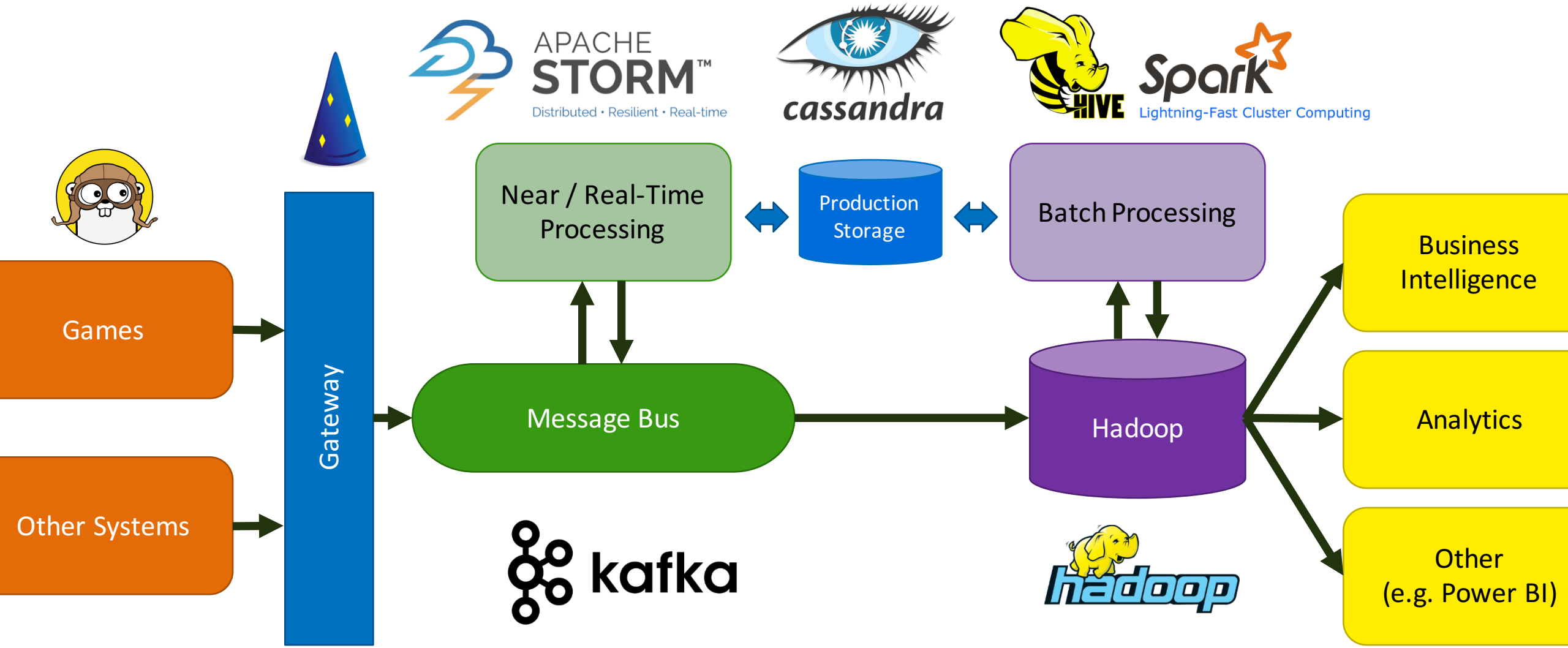

quest
build
fight
invite

500.000.000

500.000.000

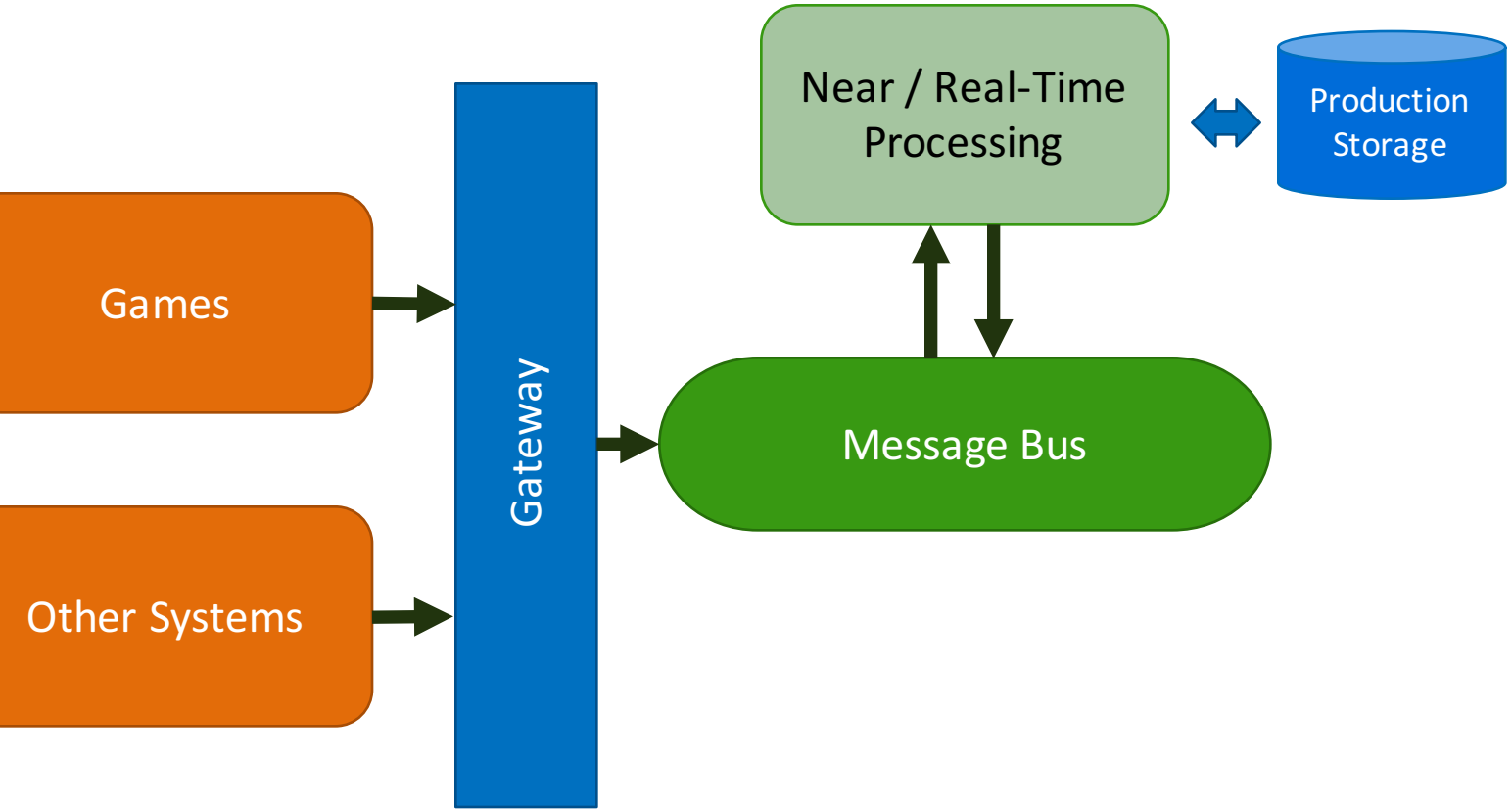
EVENTS PER DAY


seconds
milliseconds
hours
Near / Real-Time Processing
Production Storage
Batch Processing
Games
Other Systems
Gateway
Message Bus
Hadoop
Business Intelligence
Analytics
Other (e.g. Power BI)

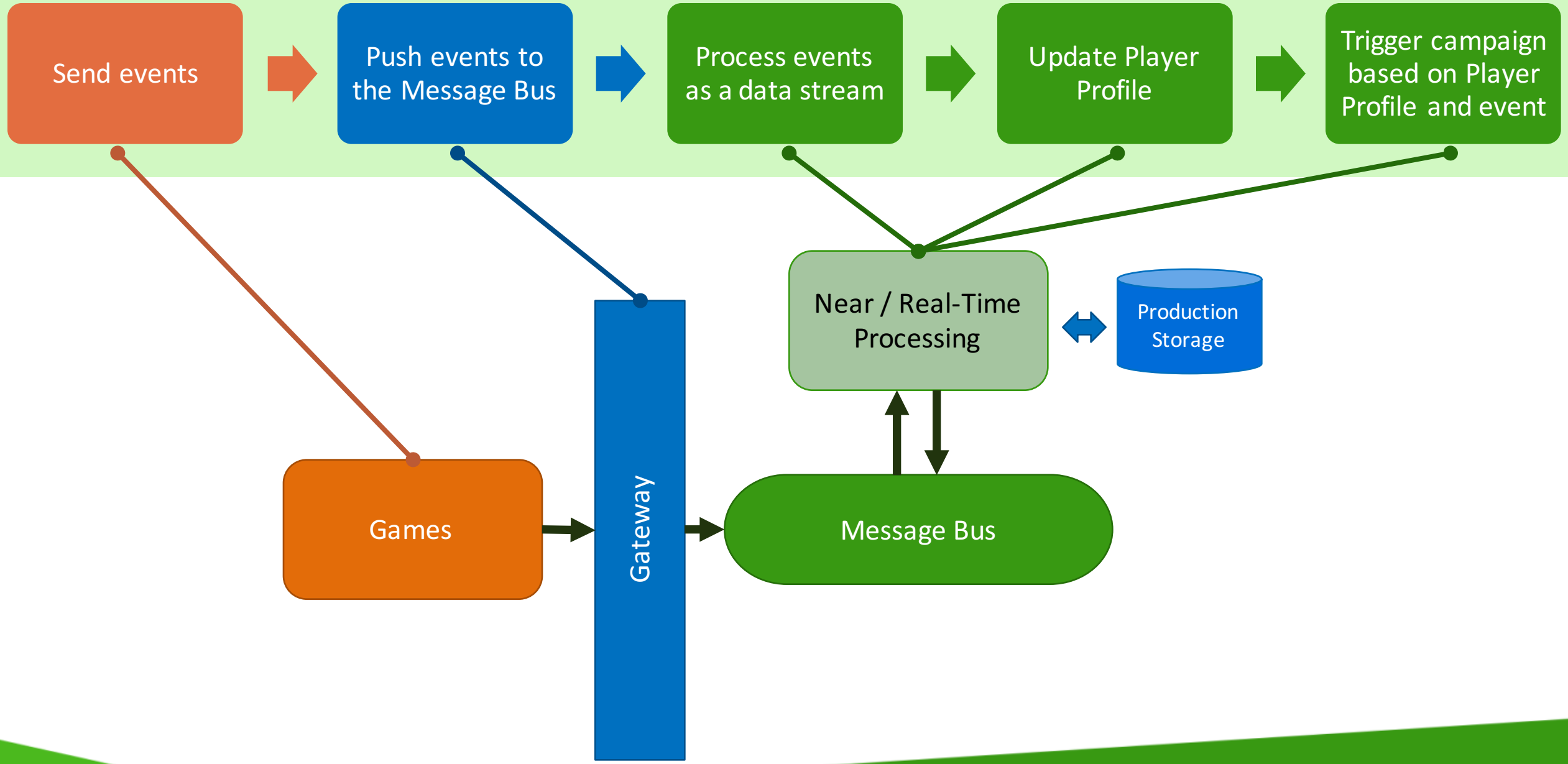


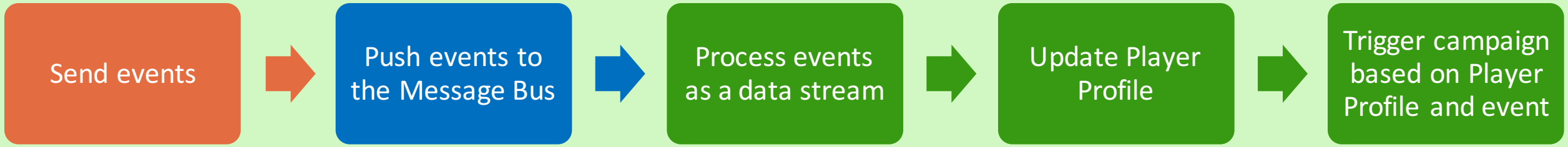
DATA PIPELINE

DATA PLATFORM



Real-Time CRM is use case of the Data Pipeline


Send events
Push events to the Message Bus
Process events as a data stream
Update Player Profile
Trigger campaign based on Player Profile and event
Near / Real-Time Processing
Production Storage
Games
Gateway
Message Bus



Key technologies:

kafka

APACHE STORM™
Distributed • Resilient • Real-time

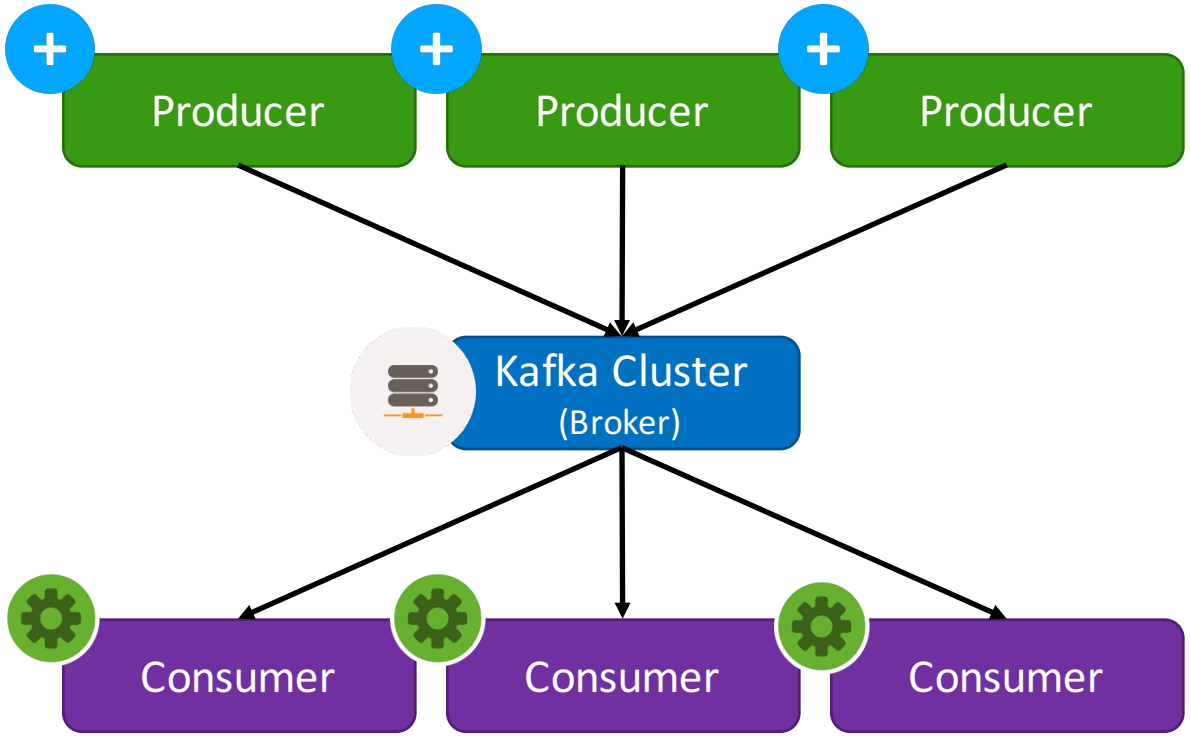
cassandra

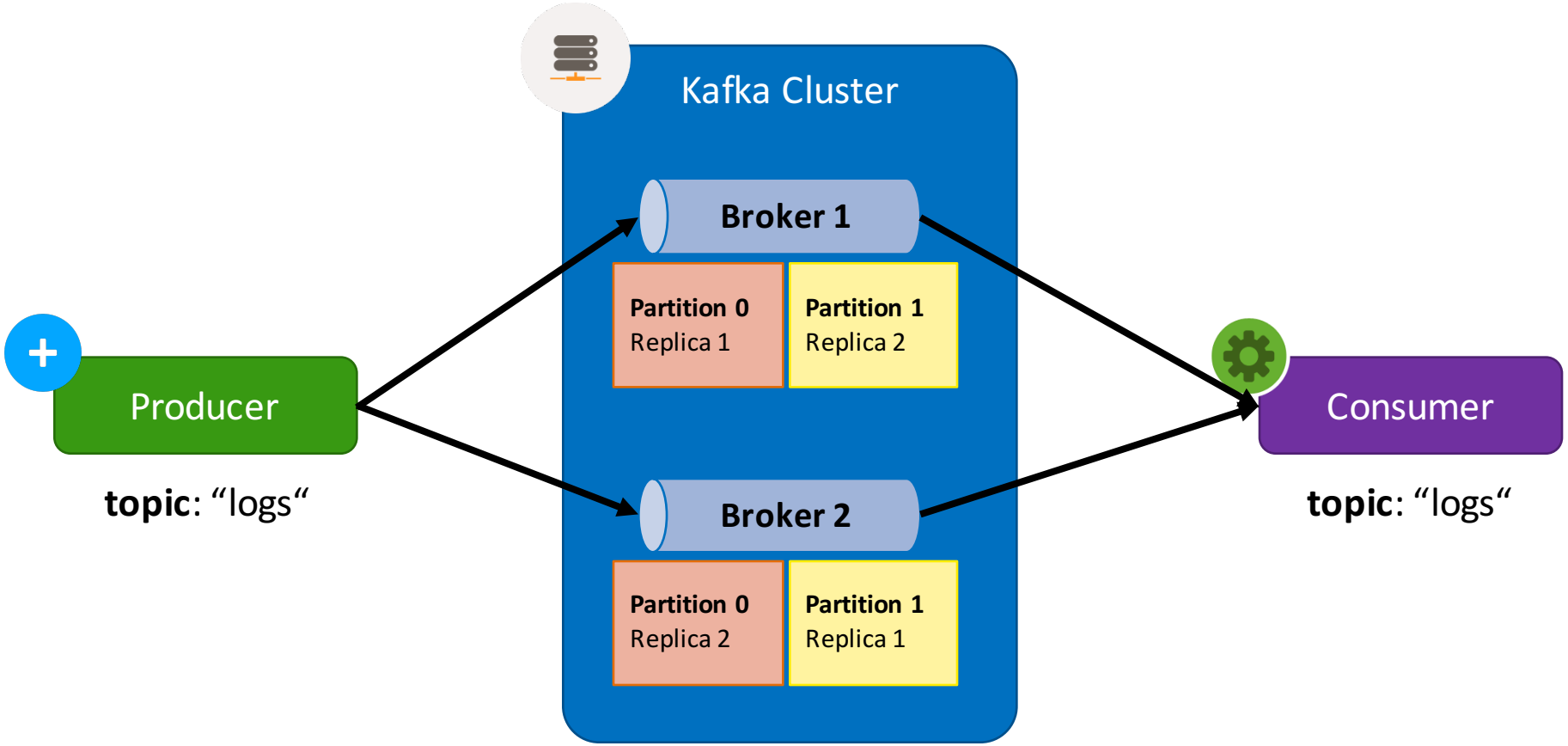
APACHE KAFKA AS A GENERIC MESSAGE BUS

http://kafka.apache.org/

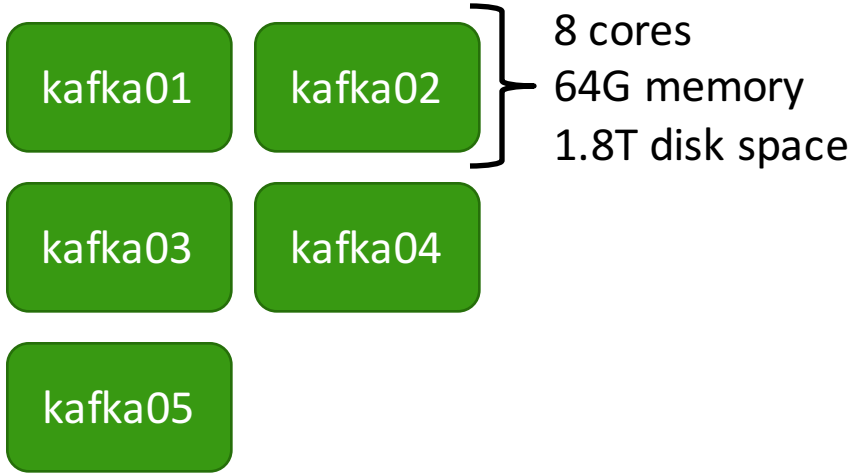
Kafka is a distributed, partitioned, replicated commit log service. It provides the functionality of a messaging system, but with a unique design.

- kafka.apache.org




Kafka Cluster
Broker 1
Partition 0
Replica 1
Partition 1
Replica 2
Broker 2
Partition 0
Replica 2
Partition 1
Replica 1
Producer
topic: “logs“
Consumer
topic: “logs“

kafka @ InnoGames



server.properties (version: 0.8.2.2):

```
…
num.replica.fetchers=4
controller.message.queue.size=10
auto.leader.rebalance.enable=true
controlled.shutdown.enable=true

num.partitions=8
log.retention.hours=168
log.flush.interval.ms=10000
log.flush.interval.messages=20000
log.flush.scheduler.interval.ms=2000
log.roll.hours=48
log.retention.check.interval.ms=300000
log.segment.bytes=1073741824

num.network.threads=8
socket.request.max.bytes=104857600
socket.receive.buffer.bytes=1048576
socket.send.buffer.bytes=1048576
queued.max.requests=16
fetch.purgatory.purge.interval.requests=100
producer.purgatory.purge.interval.requests=100
…
```


```
kafka-topics.sh --list --zookeeper x.x.x.x:2181

crm_trigger
maintenance
raw

/opt/kafka/app/bin/kafka-topics.sh --zookeeper x.x.x.x:2181 --describe --topic raw

Topic:raw   PartitionCount:16       ReplicationFactor:2     Configs:retention.ms=345600000
            Topic: raw  Partition: 0           Leader: 3   Replicas: 3,4           Isr: 3,4
            Topic: raw  Partition: 1           Leader: 4   Replicas: 4,5           Isr: 4,5
            Topic: raw  Partition: 2           Leader: 5   Replicas: 5,1           Isr: 5,1
            Topic: raw  Partition: 3           Leader: 1   Replicas: 1,2           Isr: 1,2
            Topic: raw  Partition: 4           Leader: 2   Replicas: 2,3           Isr: 2,3
            Topic: raw  Partition: 5           Leader: 3   Replicas: 3,5           Isr: 3,5
            Topic: raw  Partition: 6           Leader: 4   Replicas: 4,1           Isr: 1,4
            Topic: raw  Partition: 7           Leader: 5   Replicas: 5,2           Isr: 2,5
            Topic: raw  Partition: 8           Leader: 1   Replicas: 1,3           Isr: 1,3
            Topic: raw  Partition: 9           Leader: 2   Replicas: 2,4           Isr: 2,4
            Topic: raw  Partition: 10          Leader: 3   Replicas: 3,1           Isr: 1,3
            Topic: raw  Partition: 11          Leader: 4   Replicas: 4,2           Isr: 2,4
            Topic: raw  Partition: 12          Leader: 5   Replicas: 5,3           Isr: 3,5
            Topic: raw  Partition: 13          Leader: 1   Replicas: 1,4           Isr: 1,4
            Topic: raw  Partition: 14          Leader: 2   Replicas: 2,5           Isr: 2,5
            Topic: raw  Partition: 15          Leader: 3   Replicas: 3,2           Isr: 2,3
```

APACHE STORM FOR REAL-TIME DATA PROCESSING

http://storm.apache.org/

Apache Storm is a free and open source distributed realtime computation system.

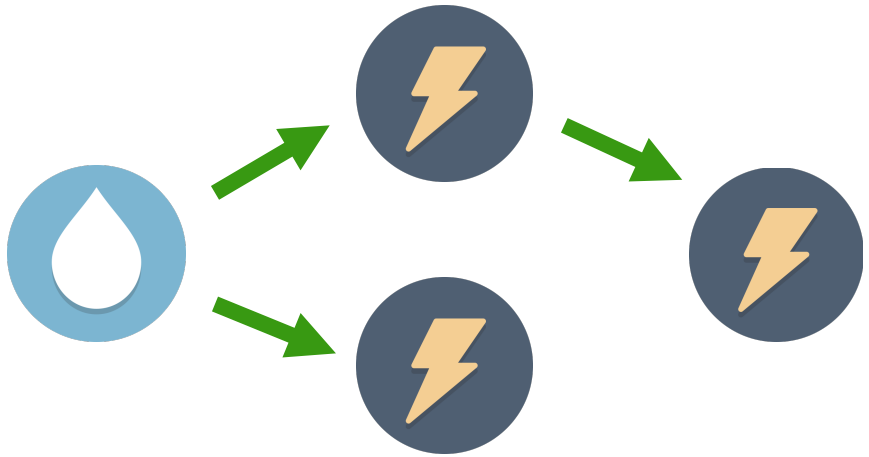
- storm.apache.org

SPOUT

Data stream sources

BOLT

Data stream processing

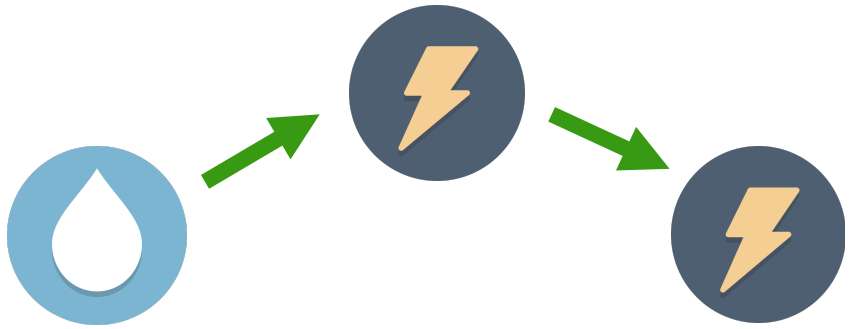


TOPOLOGY

Composition of Spouts and Bolts

FilterBolt
Checks if the temperature value is higher than a specific threshold – only in this case it sends out a tuple

TemperatureBolt
Reads the current temperature from a sensor every few milliseconds



WarnBolt
If it receives a tuple it raises an alarm

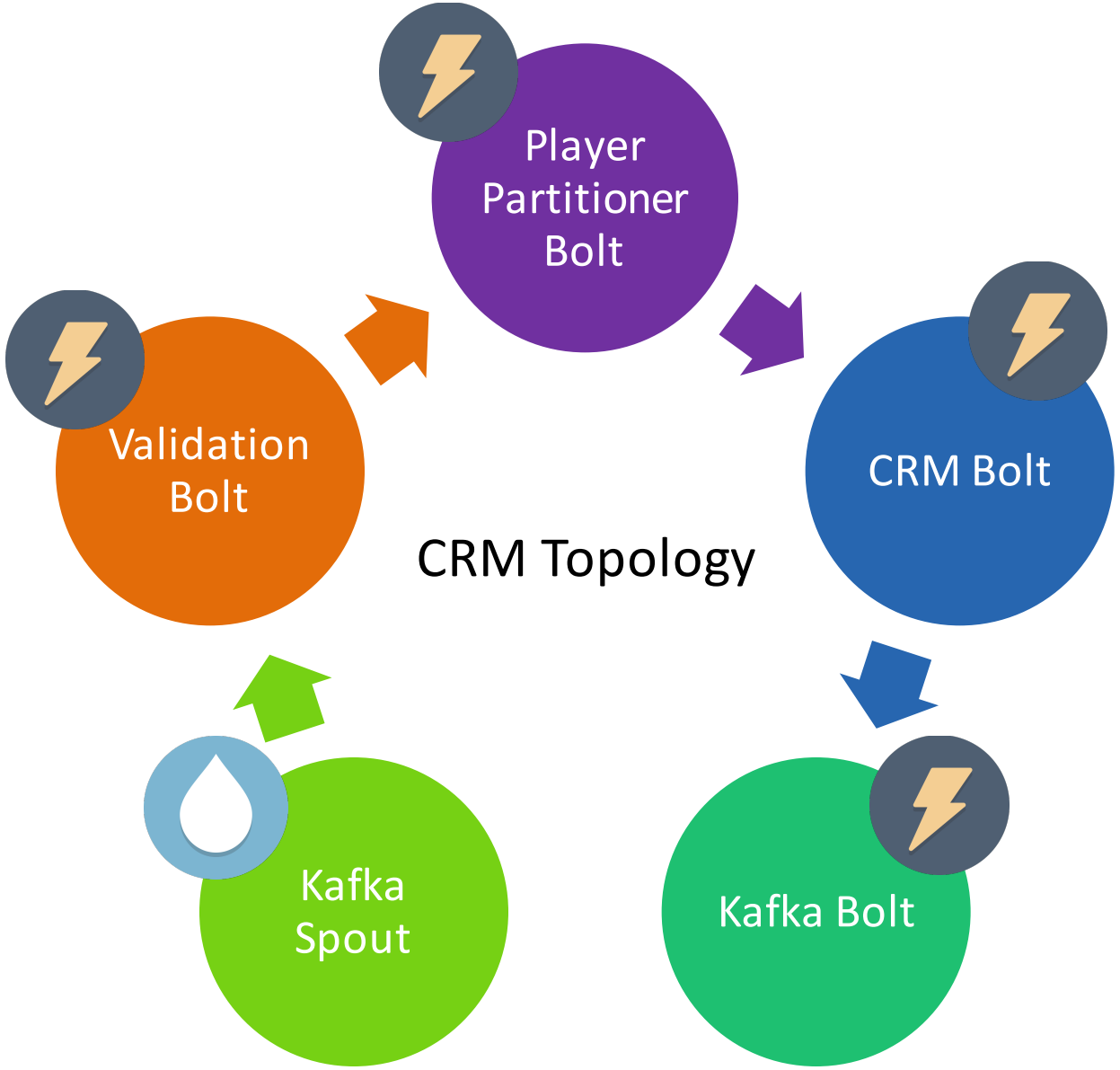
EXAMPLE

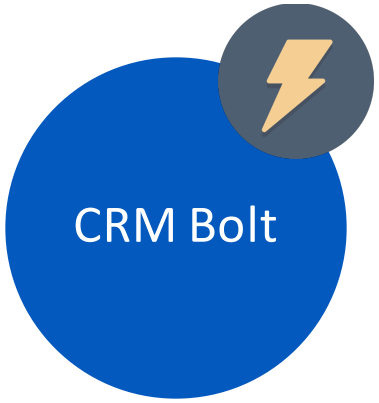
Temperature alerting

FilterBolt

Checks if the temperature value is higher than a specific threshold – only in this case it sends out a tuple

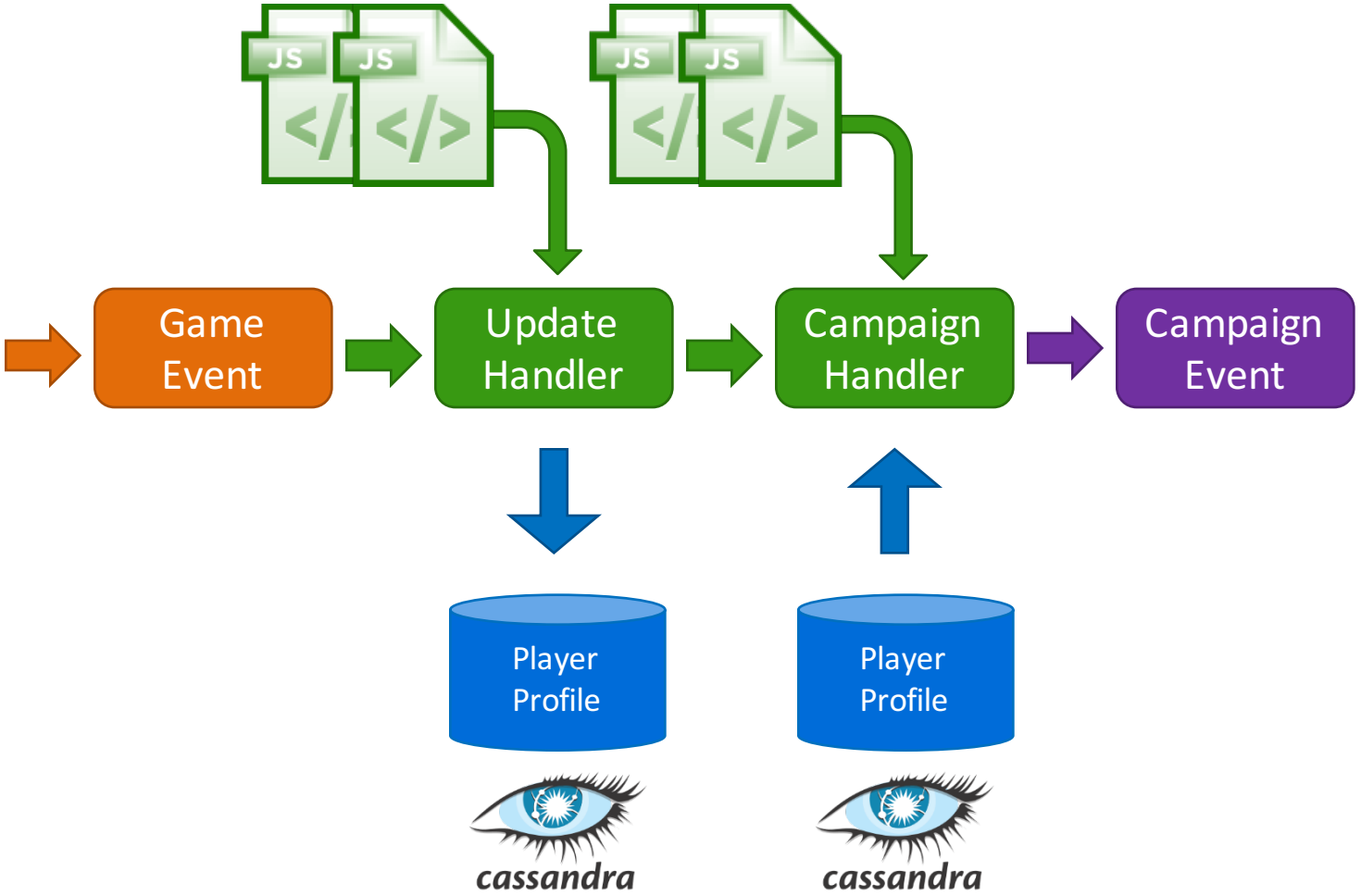
```
public class FilterBolt extends BaseRichBolt {
    private OutputCollectorBase collector;

    @Override
    public void prepare(Map conf, TopologyContext ctx, OutputCollectorBase collector) {
        this.collector = collector;
    }

    @Override
    public void execute(Tuple input) {
        int temperature = input.getInteger(0);

        if(temperature >= 50) collector.emit(new Values(temperature));
        collector.ack(input);
    }

    @Override
    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("temperature"));
    }
}
```

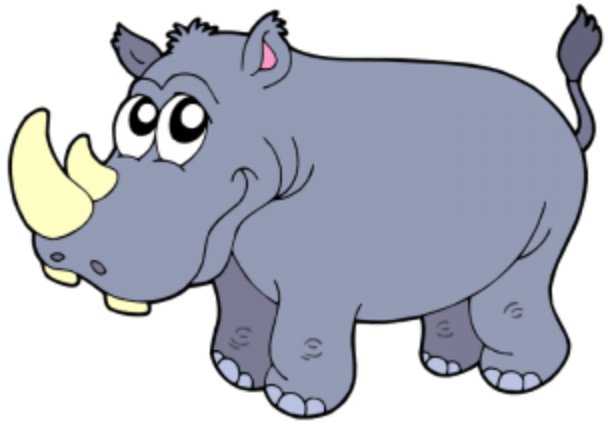

Player Partitioner Bolt
CRM Bolt
Kafka Bolt
Kafka Spout
Validation Bolt
CRM Topology



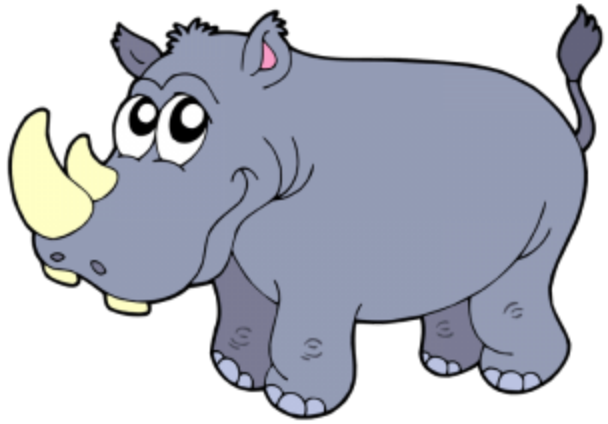
1. Execute update handlers that are defined for the incoming event.
2. Execute campaign handlers that are defined for the incoming event.

Handlers are JavaScript snippets that are processed by the **Nashorn engine**.





```
public class JsEngine {

    private final ScriptEngine engine;

    public JsEngine(final EventHandlerRegistry registry, final Storage storage) {
        final NashornScriptEngineFactory nashornScriptEngineFactory = new NashornScriptEngineFactory();
        engine = nashornScriptEngineFactory.getScriptEngine(
            new String[] {"-doe", "-strict"},
            Thread.currentThread().getContextClassLoader(),
            new JsClassFilter()
        );

        put("registry", registry);
        put("storage", storage);
        put("objectMapper", new ObjectMapper());
    }

    public void put(final String key, final Object value) {
        engine.put(key, value);
    }

    public Object eval(final String string) throws ScriptException {
        return engine.eval(string);
    }

    public Object get(final String key) {
        return engine.get(key);
    }

}
```



Updater:

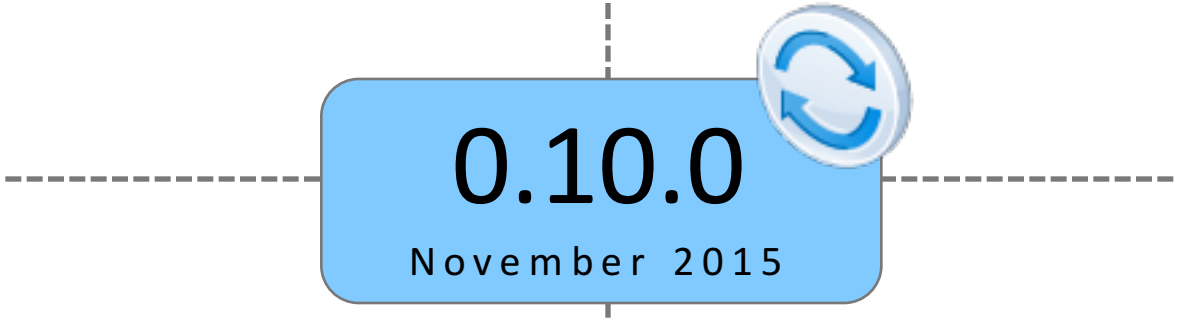
```
var identifier = "count-login-events"

var filter = new EventFilter("login")

var action = new EventAction(function(event) {
    framework.actions.countEvent(event)
})

framework.registerUpdater(identifier, new EventHandler(filter, action))
```

Campaign:

```
var content = {"id": 3, "game": "foe", "markets": ["de"]}

var trigger = {"eventType": "fight", "fn": function(event) {
    return (event.getData().result == "loss")
}}

var segment = function(player) {
    return (
        player.isPayer() == false &&
        player.getIngameLevel() < 100 &&
        player.fightsLoss() > 2
    )
}

framework.registerCampaign(content, trigger, segment)
```

Flux

Rolling Upgrades

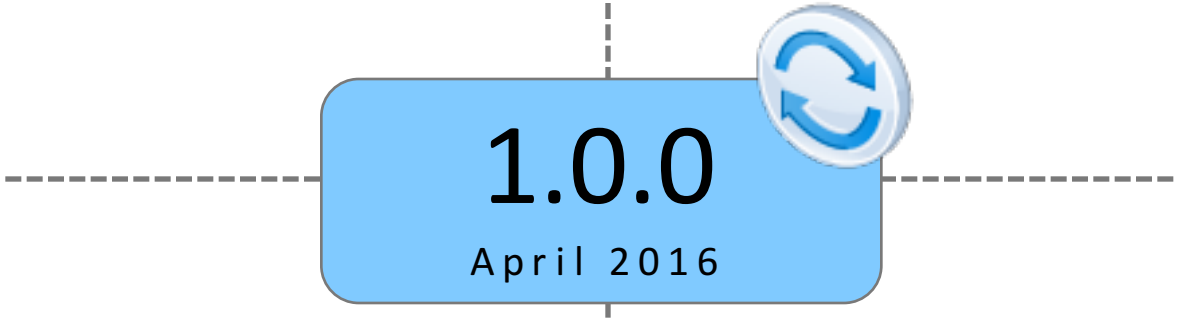


Security

Improved Logging

Improved Performance

Pacemaker

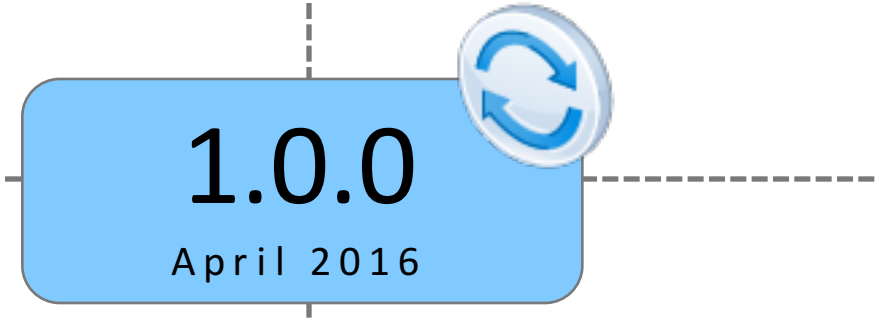


Distributed Cache

State Management

Automatic Backpressure

Resource Aware Scheduler



Native Windowing API

Debugging / Profiling

Alternatives?

Kafka Streams

Spark Streaming

Flink

Check out:

The Stream Processor as a Database: Building Online Applications directly on Streams with Apache Flink and Apache Kafka
Stephan Ewen
Berlin Buzzwords 2016

Introducing Kafka Streams, the new stream processing library of Apache Kafka
Michael Noll
Berlin Buzzwords 2016

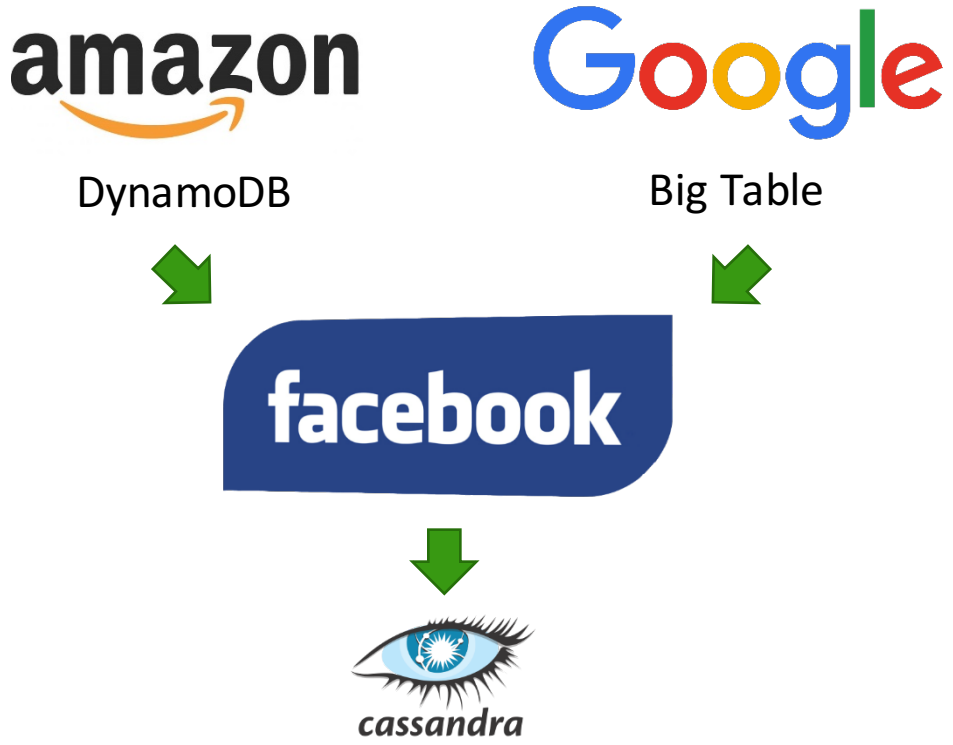
The Future of Apache Storm
P. Taylor Goetz
Hadoop Summit Dublin 2016

APACHE CASSANDRA AS A STORAGE FOR THE PLAYER PROFILE

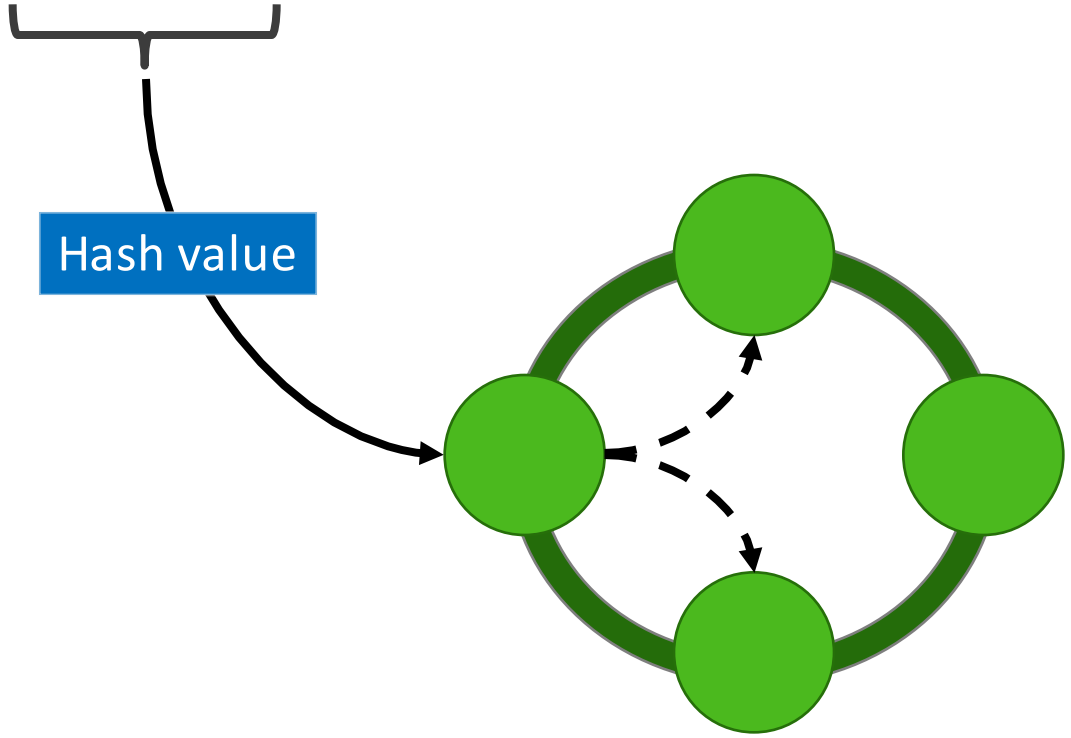
http://cassandra.apache.org/

Apache Cassandra is a massively scalable open source non-relational database that offers continuous availability, linear scale performance, operational simplicity and easy data distribution across multiple data centers and cloud availability zones.

- Datastax



SortedMap<RowKey, SortedMap<ColumnKey, ColumnValue>>



Bad model

ddmmyyhh	Timestamp 1	Timestamp 2
	Payload 1	Payload 2

Good model

ddmmyyhh player_id	Time UUID 1	Time UUID 2
	Payload 1	Payload 2

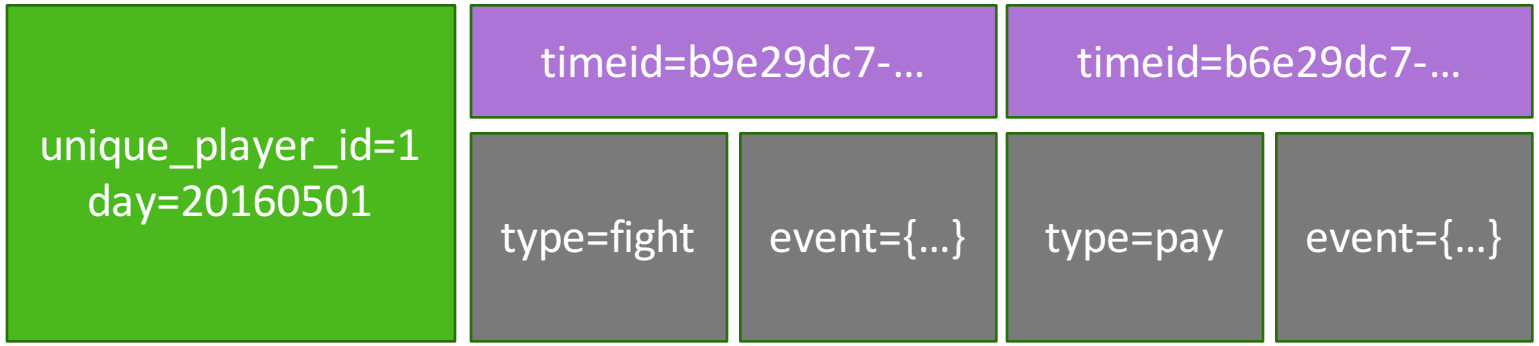
Non-goals

- Minimize number of writes.
- Minimize data duplication.

Goals

- Spread data evenly across the cluster.
- Minimize number of partitions read.
- Ensure that row and column keys are unique.
- Model column families around query patterns.
- Denormalize and duplicate data for read performance.

```
CREATE TABLE IF NOT EXISTS player_profile.event (
    unique_player_id bigint,
    day int, # yyyymmdd
    timeid timeuuid,
    type text,
    event text,
    PRIMARY KEY ((unique_player_id, day), timeid)
);
```

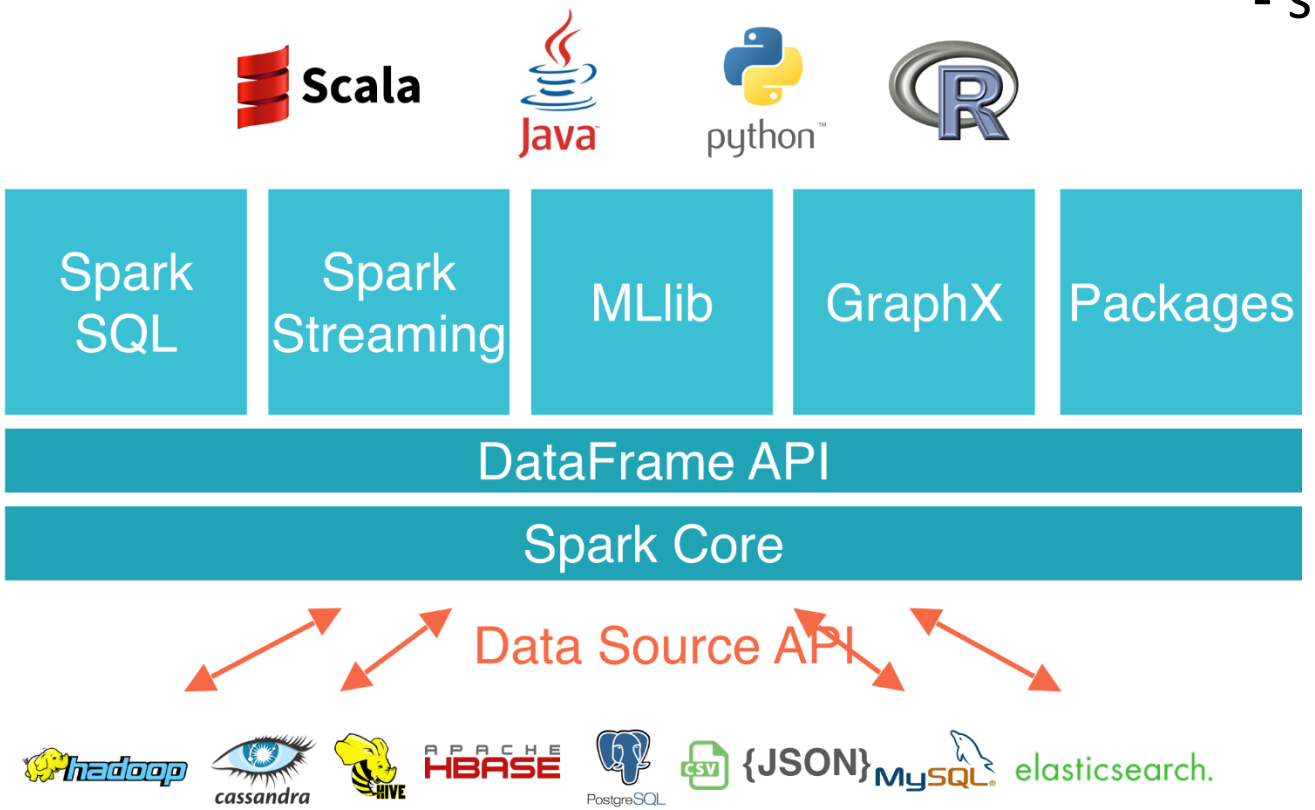


APACHE SPARK FOR ANALYTICAL QUERIES

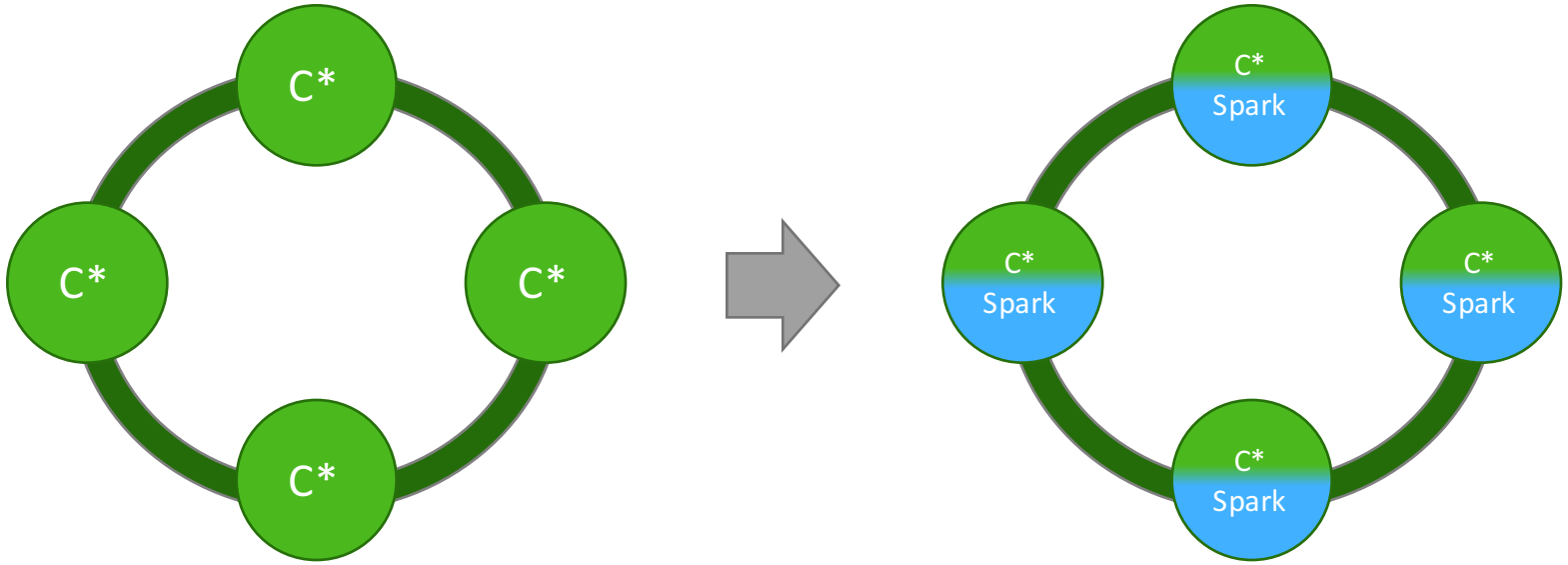
http://spark.apache.org/

Apache Spark is a fast and general-purpose cluster computing system.

- spark.apache.org



databricks



https://github.com/datastax/spark-cassandra-connector

+

Data Locality

=

Analytical queries on Cassandra data

Cached Player Profile using a shared Spark context:

```
import com.innogames.spark.jobserver.extras.JavaSparkCassandraJob;
import com.typesafe.config.Config;
import org.apache.spark.sql.cassandra.CassandraSQLContext;

public class App extends JavaSparkCassandraJob {

    public static final String CASSANDRA_TABLE = "event";

    @Override
    public Object runJob(Object sc, Config jobConfig) {
        final CassandraSQLContext cassandraSqlContext = (CassandraSQLContext) sc;
        cassandraSqlContext.setKeyspace("player_profile");

        if(!cassandraSqlContext.isCached(CASSANDRA_TABLE)) {
            cassandraSqlContext.cacheTable(CASSANDRA_TABLE);
        }

        return cassandraSqlContext.table(CASSANDRA_TABLE).javaRDD().filter(
            row -> row.getInt(1) % 2 == 0
        ).count();
    }

}
```

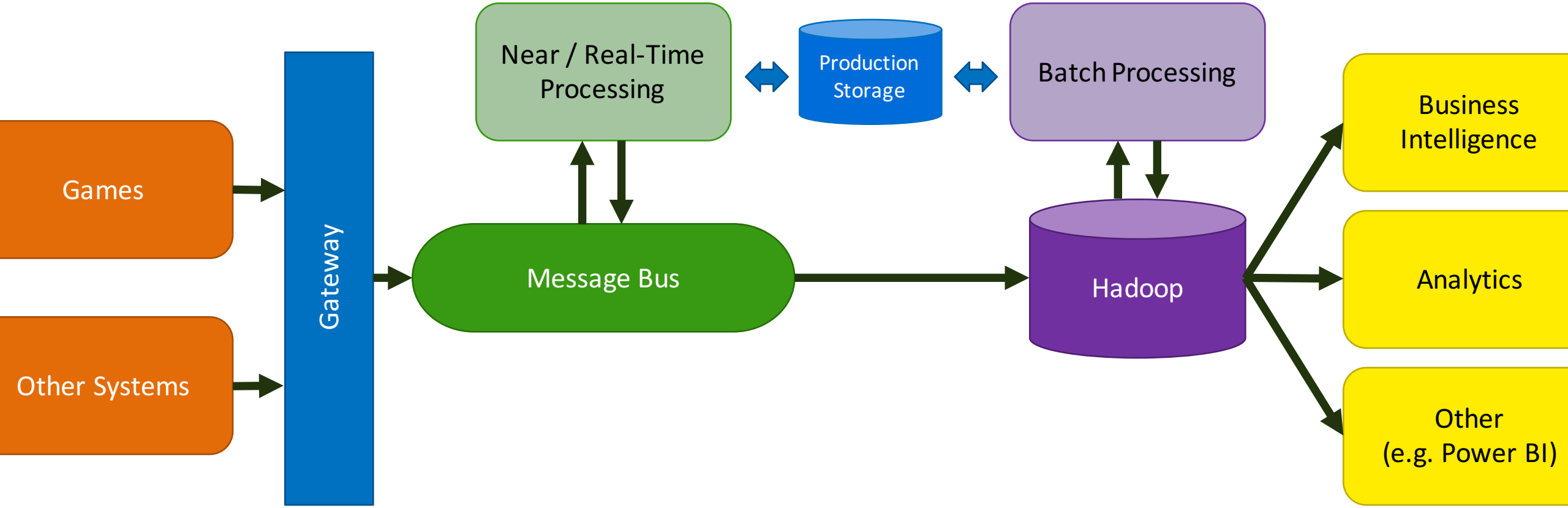
https://github.com/spark-jobserver/spark-jobserver

CONCLUSION: REAL-TIME CRM

seconds

milliseconds

hours



DATA PIPELINE

DATA PLATFORM

- Kafka is awesome.
- Storm makes it easy to implement real-time applications but...
- ...we had performance issues and defining topologies was not developer friendly.
- These disadvantages and issues were resolved with Storm 0.10.0 and 1.0.0.
- Reacting to user behavior in real-time is not only a challenge for data engineers but also for game designers.
- Combining Storm with Nashorn was a great idea that enables us to define processing logic with JavaScript that can be changed at runtime.

InnoGames

THX

</>