

# Polly: A Language-Based Approach for Change Detection in Web Service Data

**Elyas Ben Hadj Yahia**

*CProDirect*

**Jean-Rémy Falleri**

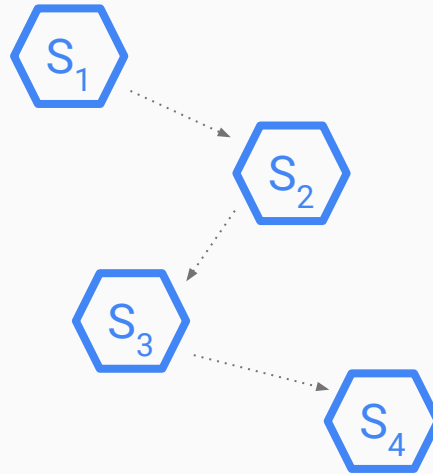
*University of Bordeaux*

**Laurent Réveillère**

*University of Bordeaux*

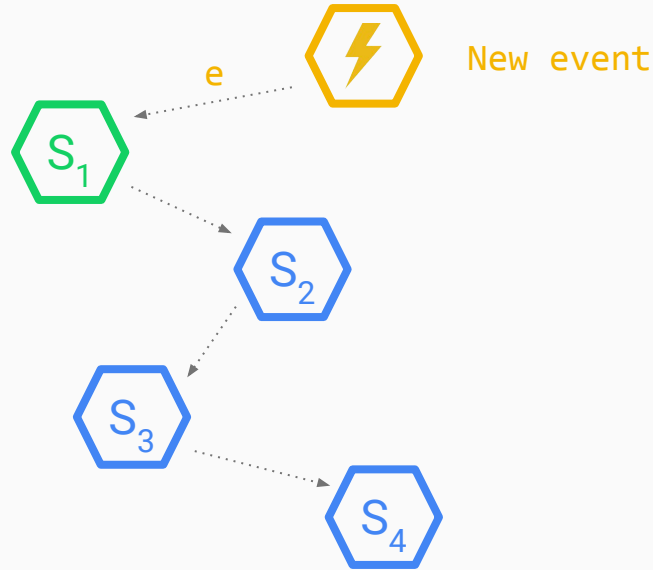
16/11/17 - ICSOC - Málaga

# Service composition



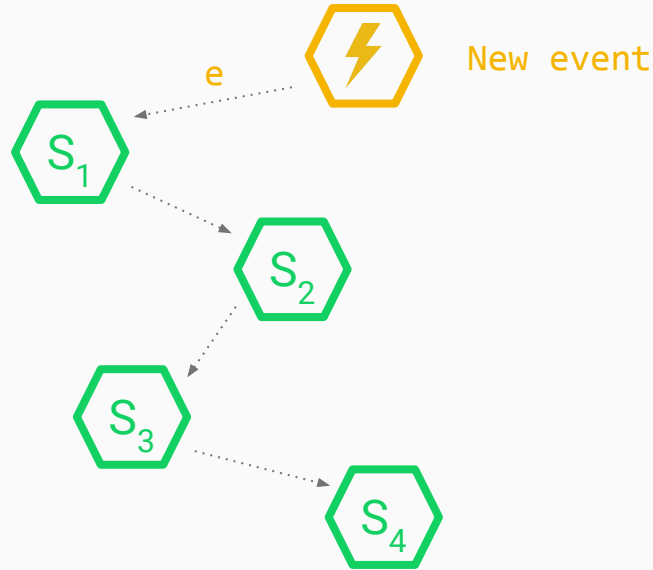
Orchestrating applications and services together to automate a **business process**

# Service composition



Orchestrating applications and services together to automate a **business process**

# Service composition



Orchestrating applications and services together to automate a **business process**

# Motivation



**Customizable** events

# Motivation



**Customizable** events

- “**New** questions on Stackoverflow”



# Motivation



## Customizable events

- “New questions on Stackoverflow”
  - “... about Angular...”

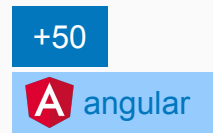


# Motivation



## Customizable events

- “**New** questions on Stackoverflow”
  - “... about **Angular**...”
  - “... with over **50+ bounty points**”





# Motivation



## Customizable events

- “**New** questions on Stackoverflow”
  - “... about **Angular**...”
  - “... with over **50+ bounty points**”
- “A stock market share price **decreased by 5+%**”



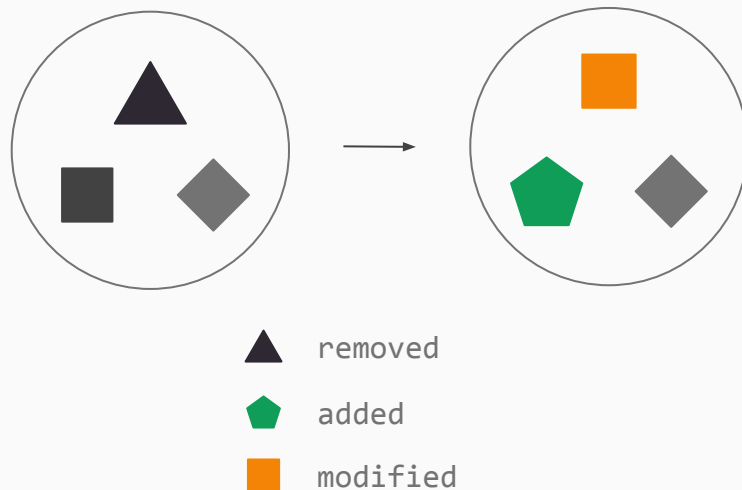
100\$ → 90\$ (-10%)

# Motivation



## Customizable events

- “**New** questions on Stackoverflow”
  - “... about **Angular**...”
  - “... with over **50+ bounty points**”
- “A stock market share price **decreased by 5+%**”
- “A **change** in the top 3 best-selling products”

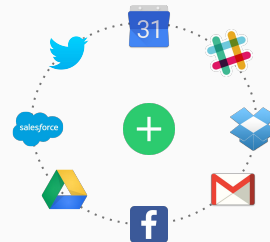


# Motivation



## Customizable events

- “**New** questions on Stackoverflow”
  - “... about **Angular**...”
  - “... with over **50+ bounty points**”
- “A stock market share price **decreased by 5+%**”
- “A **change** in the top 3 best-selling products”



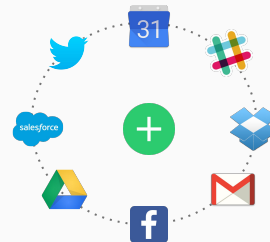
## Fast integration of new service providers

# Motivation



## Customizable events

- “**New** questions on Stackoverflow”
  - “... about **Angular**...”
  - “... with over **50+ bounty points**”
- “A stock market share price **decreased by 5+%**”
- “A **change** in the top 3 best-selling products”



## Fast integration of new service providers

- Rapidly support **any** RESTful web API
- Enable custom **change detection strategies**
- **High-level abstraction** of engineering concerns:
  - Data collection
  - Security, authentication
  - Maintenance, evolution

# A simple composition

- Automatic photo backup

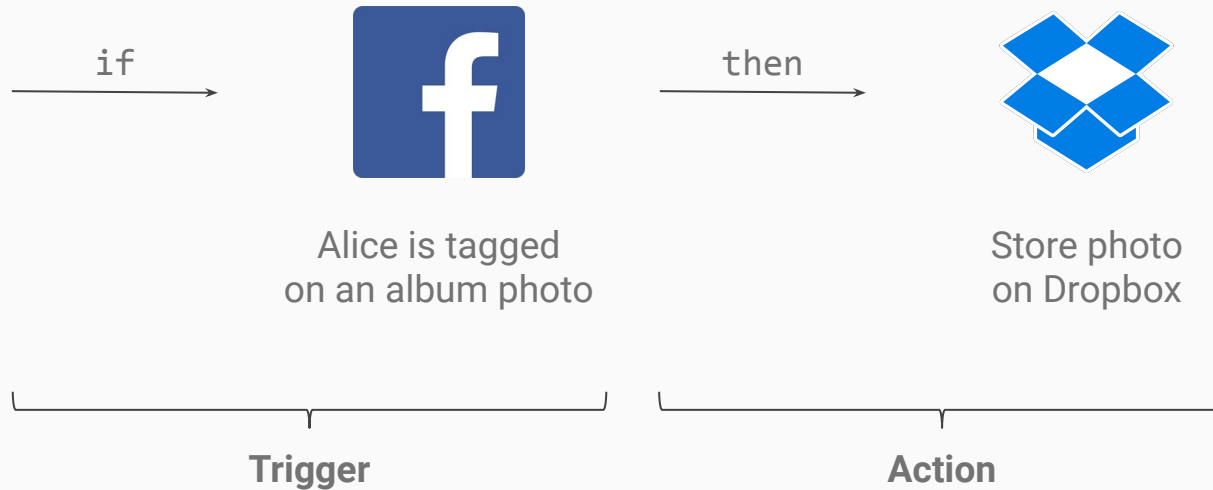
# A simple composition

- Automatic photo backup



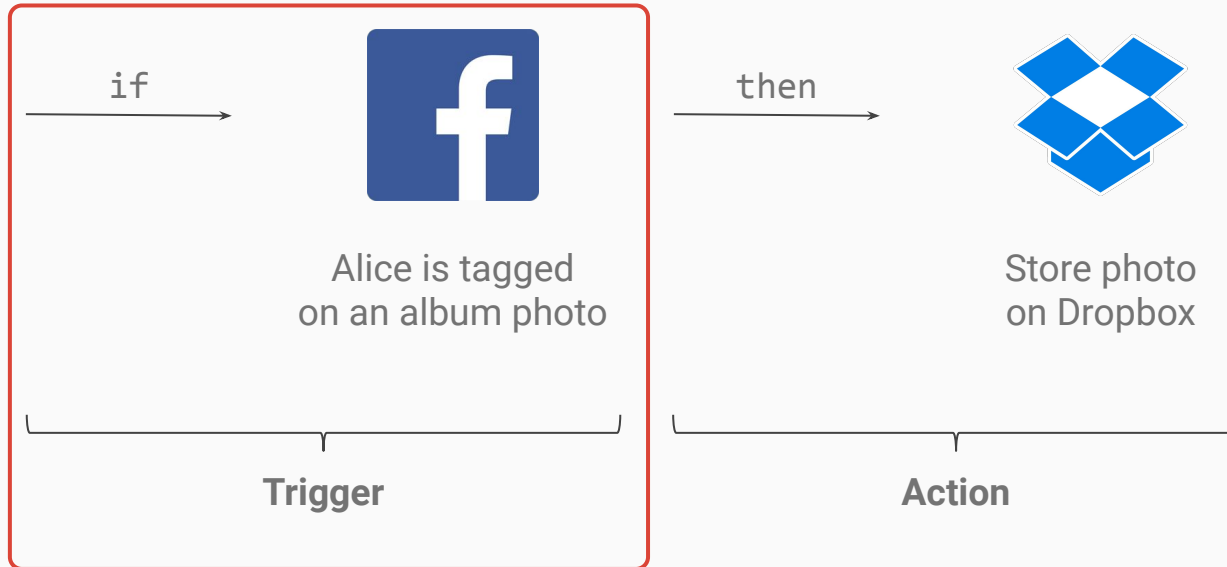
# A simple composition

- Automatic photo backup



# A simple composition

- Automatic photo backup





How to implement such triggers?

# State construction

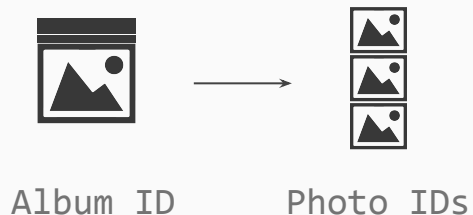
- Service providers only expose raw data (i.e. the state of a resource at a given time)
  - Example: Facebook only provides the following two API endpoints:

# State construction

- Service providers only expose raw data (i.e. the state of a resource at a given time)
  - Example: Facebook only provides the following two API endpoints:

GET `/:albumId/photos`

*Retrieves the photos of an album on Facebook.*

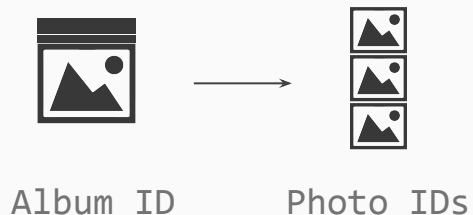


# State construction

- Service providers only expose raw data (i.e. the state of a resource at a given time)
  - Example: Facebook only provides the following two API endpoints:

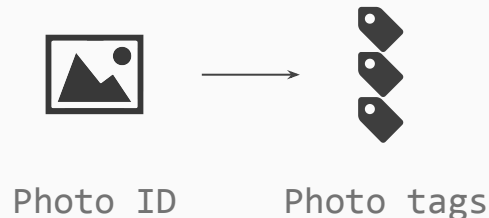
GET `/:albumId/photos`

*Retrieves the photos of an album on Facebook.*



GET `/:photoId/tags`

*Retrieves the users tagged in the photo.*



# State construction

GET /:albumId/photos

```
{
  "data": [
    {
      "created_time": "2016-05-20T12:28:57",
      "updated_time": "2016-05-20T12:26:57",
      "id": "1106290499393017"
    }
  ],
  "paging": {
    "next": "https://graph.facebook.com/..."
  }
}
```



Album ID

# State construction

GET `/:albumId/photos`

```
{
  "data": [
    {
      "created_time": "2016-05-20T12:28:57",
      "updated_time": "2016-05-20T12:26:57",
      "id": "1106290499393017"
    }
  ],
  "paging": {
    "next": "https://graph.facebook.com/..."
  }
}
```



Album ID



Photo IDs

# State construction

GET /:albumId/photos

```
{
  "data": [
    {
      "created_time": "2016-05-20T12:28:57",
      "updated_time": "2016-05-20T12:26:57",
      "id": "1106290499393017"
    }
  ],
  "paging": {
    "next": "https://graph.facebook.com/..."
  }
}
```



Album ID

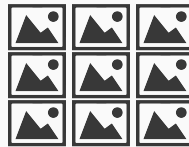


Photo IDs

# State construction

GET /:albumId/photos

```
{
  "data": [
    {
      "created_time": "2016-05-20T12:28:57",
      "updated_time": "2016-05-20T12:26:57",
      "id": "1106290499393017"
    }
  ],
  "paging": {
    "next": "https://graph.facebook.com/..."
  }
}
```



Album ID

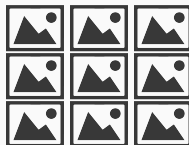


Photo IDs

GET /:photoId/tags

```
[
  {
    "created_time": "2016-05-20T12:26:57",
    "updated_time": "2016-05-20T12:28:57",
    "id": "1106290499393017",
    "data": [
      {
        "id": "10203528656797589",
        "name": "Bob",
        "x": 73.684210526316,
        "y": 74.865350089767
      }
    ]
  }
]
```



Photo tags



# Change detection

```
[
  {
    "created_time": "2016-05-20T12:26:57",
    "updated_time": "2016-05-20T12:28:57",
    "id": "1106290499393017",
    "data": [
      {
        "id": "10203528656797589",
        "name": "Bob",
        "x": 73.684210526316,
        "y": 74.865350089767
      }
    ]
  }
]
```

Initial state ( $t_0$ )

# Change detection

```
[
  {
    "created_time": "2016-05-20T12:26:57",
    "updated_time": "2016-05-20T12:28:57",
    "id": "1106290499393017",
    "data": [
      {
        "id": "10203528656797589",
        "name": "Bob",
        "x": 73.684210526316,
        "y": 74.865350089767
      }
    ]
  }
]
```

Initial state ( $t_0$ )

```
[
  {
    "created_time": "2016-05-20T12:26:57",
    "updated_time": "2016-05-20T12:29:57",
    "id": "1106290499393017",
    "data": [
      {
        "id": "10203528656797589",
        "name": "Bob",
        "x": 76.684210526316,
        "y": 74.865350089767
      }
    ]
  },
  {
    "created_time": "2016-05-20T12:35:57",
    "id": "2206280499393006",
    "data": [
      {
        "id": "20406528656797578",
        "name": "Alice",
        "x": 63.684210526316,
        "y": 62.865350089767
      }
    ]
  }
]
```

Updated state ( $t_1$ )

# Change detection

```
[
  {
    "created_time": "2016-05-20T12:26:57",
    "updated_time": "2016-05-20T12:28:57",
    "id": "1106290499393017",
    "data": [
      {
        "id": "10203528656797589",
        "name": "Bob",
        "x": 73.684210526316,
        "y": 74.865350089767
      }
    ]
  }
]
```

Initial state ( $t_0$ )

```
[
  {
    "created_time": "2016-05-20T12:26:57",
    "updated_time": "2016-05-20T12:29:57",
    "id": "1106290499393017",
    "data": [
      {
        "id": "10203528656797589",
        "name": "Bob",
        "x": 76.684210526316,
        "y": 74.865350089767
      }
    ]
  },
  {
    "created_time": "2016-05-20T12:35:57",
    "id": "2206280499393006",
    "data": [
      {
        "id": "20406528656797578",
        "name": "Alice",
        "x": 63.684210526316,
        "y": 62.865350089767
      }
    ]
  }
]
```

Updated state ( $t_1$ )

# Change detection

- Resulting JSON diff

```
[
  {
    "op": "replace",
    "path": "/0/data/0/x",
    "value": 76.684210526316
  },
  {
    "op": "replace",
    "path": "/0/updated_time",
    "value": "2016-05-20T12:29:57"
  },
  {
    "op": "add",
    "path": "/1",
    "value": {
      "created_time": "2016-05-20T12:35:57",
      "id": "2206280499393006",
      "data": [
        {
          "id": "0406528656797578",
          "name": "Alice",
          "x": 63.684210526316,
          "y": 62.865350089767
        }
      ]
    }
  }
]
```

# Change detection

- Resulting JSON diff

```
[
  {
    "op": "replace",
    "path": "/0/data/0/x",
    "value": 76.684210526316
  },
  {
    "op": "replace",
    "path": "/0/updated_time",
    "value": "2016-05-20T12:29:57"
  },
  {
    "op": "add",
    "path": "/1",
    "value": {
      "created_time": "2016-05-20T12:35:57",
      "id": "2206280499393006",
      "data": [
        {
          "id": "0406528656797578",
          "name": "Alice",
          "x": 63.684210526316,
          "y": 62.865350089767
        }
      ]
    }
  }
]
```

Irrelevant changes  
(wrt. scenario)

# Change detection

- Resulting JSON diff

```
[
  {
    "op": "replace",
    "path": "/0/data/0/x",
    "value": 76.684210526316
  },
  {
    "op": "replace",
    "path": "/0/updated_time",
    "value": "2016-05-20T12:29:57"
  },
  {
    "op": "add",
    "path": "/1",
    "value": {
      "created_time": "2016-05-20T12:35:57",
      "id": "2206280499393006",
      "data": [
        {
          "id": "0406528656797578",
          "name": "Alice",
          "x": 63.684210526316,
          "y": 62.865350089767
        }
      ]
    }
  }
]
```

Irrelevant changes  
(wrt. scenario)

Relevant changes  
(wrt. scenario)

# Change detection

- Resulting JSON diff

## Developers need to:

- Post-process the diff output
- Identify relevant changes
- Extract and transform them into a usable format

```
[
  {
    "op": "replace",
    "path": "/0/data/0/x",
    "value": 76.684210526316
  },
  {
    "op": "replace",
    "path": "/0/updated_time",
    "value": "2016-05-20T12:29:57"
  },
  {
    "op": "add",
    "path": "/1",
    "value": {
      "created_time": "2016-05-20T12:35:57",
      "id": "2206280499393006",
      "data": [
        {
          "id": "0406528656797578",
          "name": "Alice",
          "x": 63.684210526316,
          "y": 62.865350089767
        }
      ]
    }
  }
]
```

Irrelevant changes  
(wrt. scenario)

Relevant changes  
(wrt. scenario)

# Challenges

- Challenges in polling for changes



## State construction

Chaining multiple endpoints and navigating through responses to construct a state



## Change detection

Identifying relevant changes, and ignoring irrelevant ones



Enter Polly

# Approach

- **Polly**, a declarative DSL for custom change detection in web service data

# Approach

- **Polly**, a declarative DSL for custom change detection in web service data



Language constructs for constructing a state from one or multiple API endpoints

# Approach

- **Polly**, a declarative DSL for custom change detection in web service data



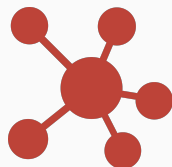
Language constructs for constructing a state from one or multiple API endpoints



Describing custom change detection strategies in JSON data from REST APIs

# Approach

- **Polly**, a declarative DSL for custom change detection in web service data



Language constructs for constructing a state from one or multiple API endpoints



Describing custom change detection strategies in JSON data from REST APIs



Compiles to efficient Node.js code

# Approach

- **Polly**, a declarative DSL for custom change detection in web service data



Language constructs for constructing a state from one or multiple API endpoints



Describing custom change detection strategies in JSON data from REST APIs



Compiles to efficient Node.js code

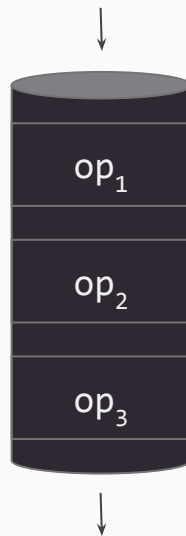


Validated on several real use-cases

# Architecture

- Processing pipelines for defining custom change detectors
  - Expressed as a series of transformation operations on successive sets of data
  - Data and operations on it are independent from each other
  - Each operation produces a JSON document that is passed as input for the following operation

```
pipeline:  
- operation: op1  
  definition: [...]  
  
- operation: op2  
  definition: [...]  
  
- operation: op3  
  definition: [...]
```



# Language operators - fetch (1/2)

- Collecting data from a set of API endpoints

```
operation: fetch  
definition: # [...]
```



# Language operators - fetch (1/2)

- Collecting data from a set of API endpoints

```
operation: fetch
definition:
  request:
    url: https://graph.facebook.com/v2.9/:albumId/photos
    params:
      albumId: 465607303461343
    query:
      access_token: XXXX
    headers:
      Accept: application/json
```

- Request object construction
  - URL parameters
  - Query parameters
  - HTTP headers

# Language operators - fetch (1/2)

- Collecting data from a set of API endpoints

```
operation: fetch
definition:
  request:
    url: https://graph.facebook.com/v2.9/:albumId/photos
    params:
      albumId: 465607303461343
    query:
      access_token: XXXX
    headers:
      Accept: application/json
    template: ~.data # ~ denotes the response body
```

- Request object construction
  - URL parameters
  - Query parameters
  - HTTP headers
- Custom templating
  - Transform response data
  - Include only relevant fields

# Language operators - fetch (1/2)

- Collecting data from a set of API endpoints

```
operation: fetch
definition:
  request:
    url: https://graph.facebook.com/v2.9/:albumId/photos
    params:
      albumId: 465607303461343
    query:
      access_token: XXXX
    headers:
      Accept: application/json
  template: ~.data # ~ denotes the response body
  pagination:
    next: ~.paging.next
```

- Request object construction
  - URL parameters
  - Query parameters
  - HTTP headers
- Custom templating
  - Transform response data
  - Include only relevant fields
- Automatic pagination handling
  - Recursively navigate to following pages

# Language operators - fetch (2/2)

- Fetching multiple resources in parallel

```
operation: fetch
definition:

request:
  url: https://graph.facebook.com/v2.9/:photoId/tags
  query:
    access_token: XXXX
  headers:
    Accept: application/json
  pagination:
    next: ~.paging.next
  template:
    photoId: ^.id
    tags: ~.data
```

# Language operators - fetch (2/2)

- Fetching multiple resources in parallel

```
operation: fetch
definition:
  repeat:
    forEach: _          # _ denotes the input object
    placeholders:
      photoId: ^.id     # ^ denotes the iteration cursor
  request:
    url: https://graph.facebook.com/v2.9/:photoId/tags
    query:
      access_token: XXXX
    headers:
      Accept: application/json
  pagination:
    next: ~.paging.next
  template:
    photoId: ^.id
    tags: ~.data
```

- Multiple requests in parallel
  - Build a URL for each iteration cursor
  - Fetch resources asynchronously

# Language operators - filterArray

- Detecting custom changes in array structures

```
operation: filterArray  
definition: # [...]
```

# Language operators - filterArray

- Detecting custom changes in array structures

```
operation: filterArray
definition:
  identifiers:
    - ^.photoId
```

- Item identification
  - Specify how to uniquely identify an item within a collection

# Language operators - filterArray

- Detecting custom changes in array structures

```
operation: filterArray
definition:
  identifiers:
    - ^.photoId
  find:
    - addedItems
  output: &.addedItems # & denotes the output object
```

- Item identification
  - Specify how to uniquely identify an item within a collection
- Change detection strategy
  - Change types to watch for
  - Leverage predefined change types



# Language operators - filterCustom

- Custom filtering strategy

```
operation: filterCustom
definition:
  function: !!js/function >
    function (oldState, newState) {
      const result = [];
      # Filter the input, keeping only photos
      # where Alice was newly tagged.
      return result;
    }
```

# Language operators - filterCustom

- Custom filtering strategy

```
operation: filterCustom
definition:
  function: !!js/function >
    function (oldState, newState) {
      const result = [];
      # Filter the input, keeping only photos
      # where Alice was newly tagged.
      return result;
    }
```

- User-defined logic
  - Javascript hook implementation
  - Runs in secure isolated sandbox

# Evaluation

# Evaluation

- Six use cases provided by our industrial partner

# Evaluation

- Six use cases provided by our industrial partner



**ElasticSearch**

Notify when the top 5 best-selling products change in ranking order

# Evaluation

- Six use cases provided by our industrial partner



**ElasticSearch**

Notify when the top 5 best-selling products change in ranking order



**Facebook**

Notify when users X and Y are tagged together

# Evaluation

- Six use cases provided by our industrial partner



**ElasticSearch**

Notify when the top 5 best-selling products change in ranking order



**Facebook**

Notify when users X and Y are tagged together



**GitHub**

Notify about new Go projects with over 2,000 stars

# Evaluation

- Six use cases provided by our industrial partner



**ElasticSearch**

Notify when the top 5 best-selling products change in ranking order



**Facebook**

Notify when users X and Y are tagged together



**GitHub**

Notify about new Go projects with over 2,000 stars



**StackOverflow**

Notify about new JS questions with bounty > 100 reputation points



# Evaluation

- Six use cases provided by our industrial partner



**ElasticSearch**

Notify when the top 5 best-selling products change in ranking order



**Facebook**

Notify when users X and Y are tagged together



**GitHub**

Notify about new Go projects with over 2,000 stars



**StackOverflow**

Notify about new JS questions with bounty > 100 reputation points



**Transport for London**

Notify when status of Victoria subway line changes

# Evaluation

- Six use cases provided by our industrial partner



**ElasticSearch**

Notify when the top 5 best-selling products change in ranking order



**Facebook**

Notify when users X and Y are tagged together



**GitHub**

Notify about new Go projects with over 2,000 stars



**StackOverflow**

Notify about new JS questions with bounty > 100 reputation points



**Transport for London**

Notify when status of Victoria subway line changes



**Twitter**

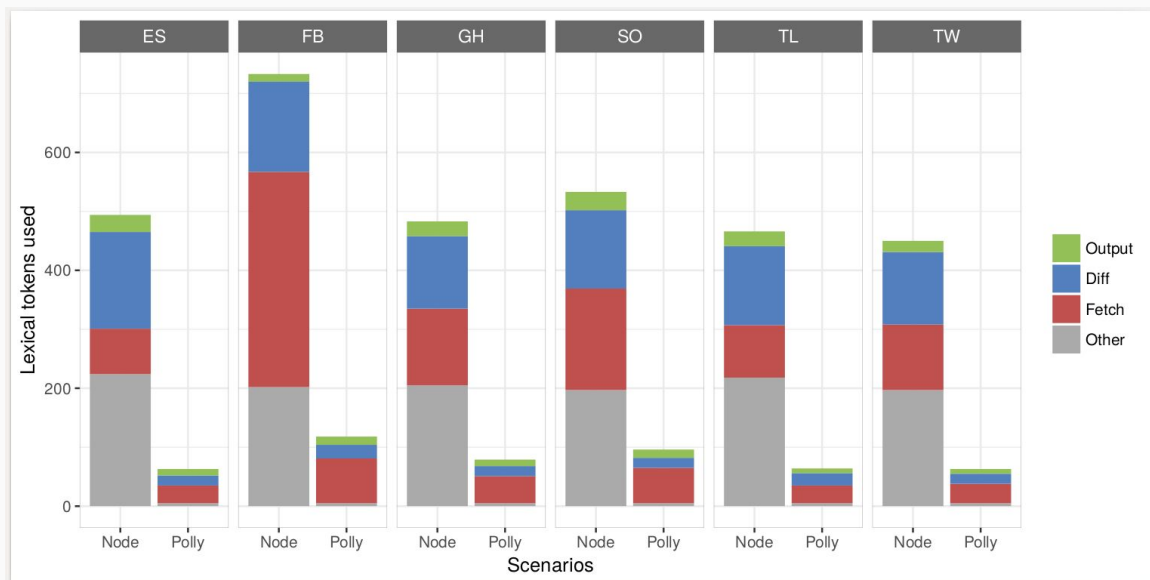
Notify whenever the official *Bordeaux* account has new followers

# Evaluation

- Analysis of abstraction level
  - Verbosity measured by number of lexical tokens used to express a scenario
  - Comparison of handwritten Node.js implementation vs. [Polly](#)

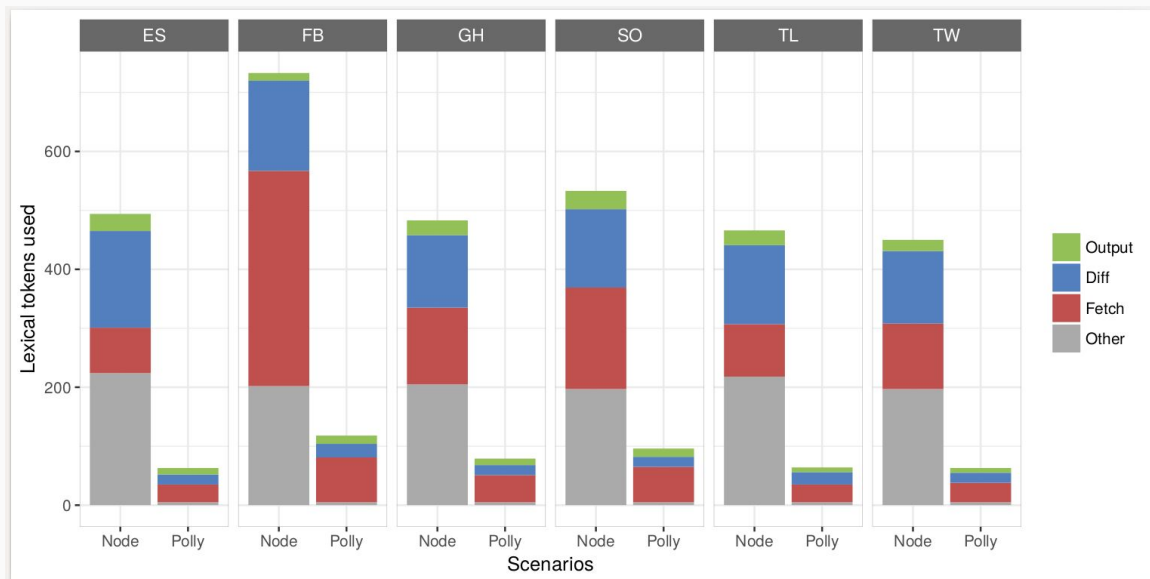
# Evaluation

- Analysis of abstraction level
  - Verbosity measured by number of lexical tokens used to express a scenario
  - Comparison of handwritten Node.js implementation vs. **Polly**



# Evaluation

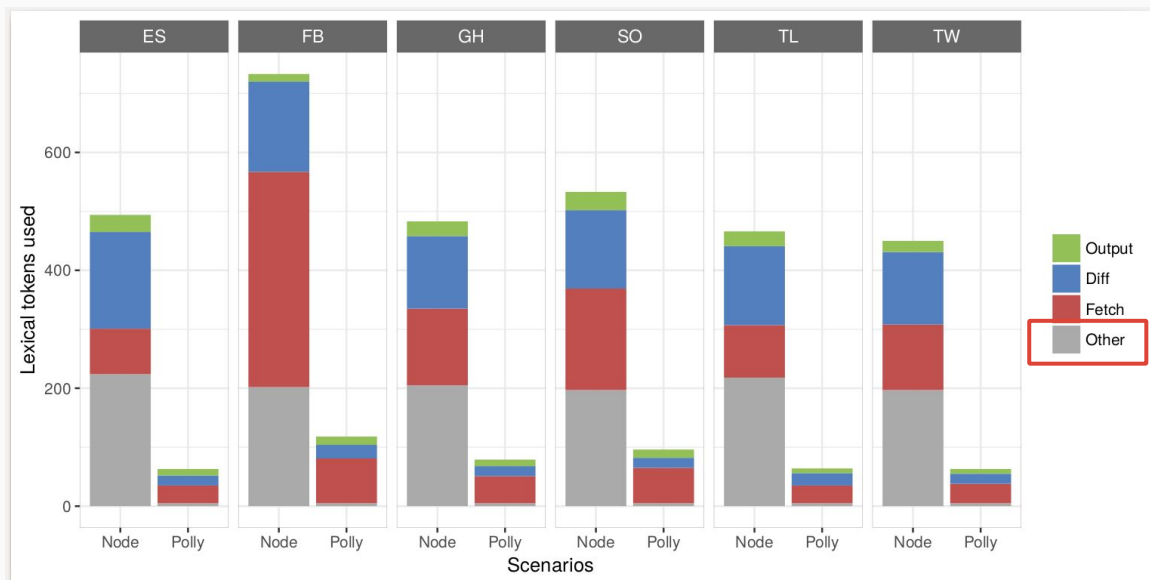
- Analysis of abstraction level
  - Verbosity measured by number of lexical tokens used to express a scenario
  - Comparison of handwritten Node.js implementation vs. **Polly**



- Programs written in **Polly** **5.5 to 8 times smaller** than Node.js

# Evaluation

- Analysis of abstraction level
  - Verbosity measured by number of lexical tokens used to express a scenario
  - Comparison of handwritten Node.js implementation vs. **Polly**



- Programs written in **Polly** **5.5 to 8 times smaller** than Node.js
- Requires a lot less boilerplate code

# Evaluation

- Performance analysis
  - Comparison against the top performing JSON differencing tool: **JDR**
  - Analyze differencing time and output size

# Evaluation

- Performance analysis
  - Comparison against the top performing JSON differencing tool: **JDR**
  - Analyze differencing time and output size
- Experimental setup



Collect real data from services  
*(snapshot every 5 min for 48 hours = 576 snapshots)*



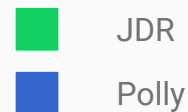
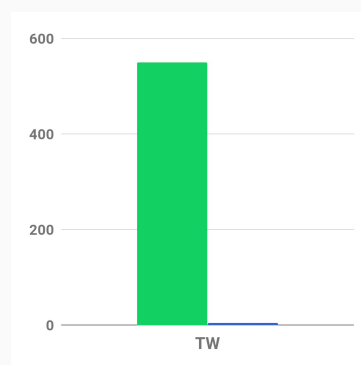
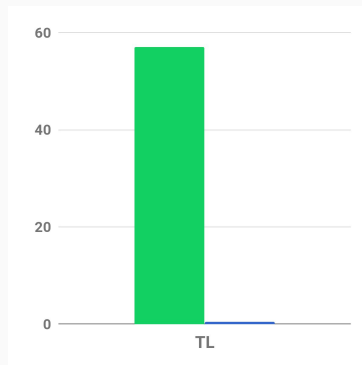
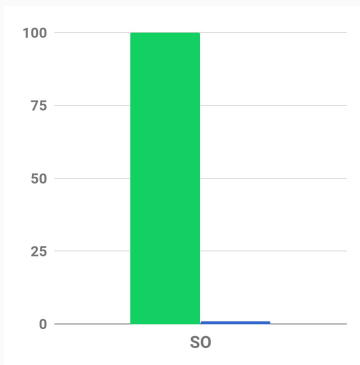
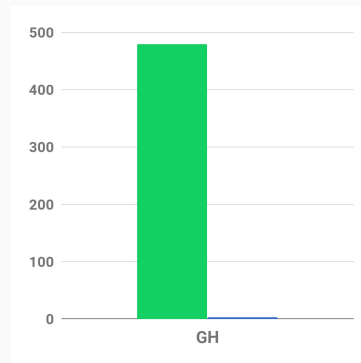
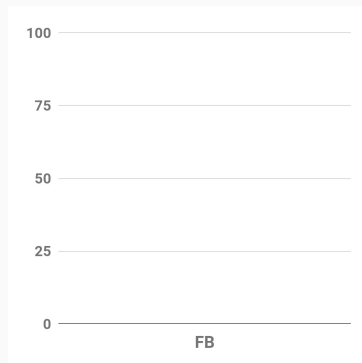
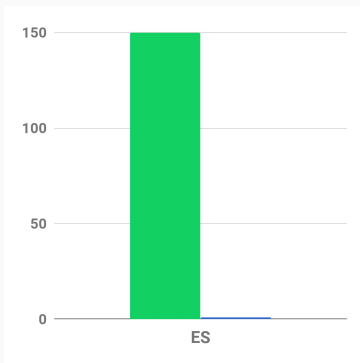
Mock server  
*Serve collected data*



Run scenarios and measure metrics

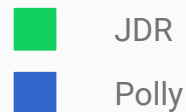
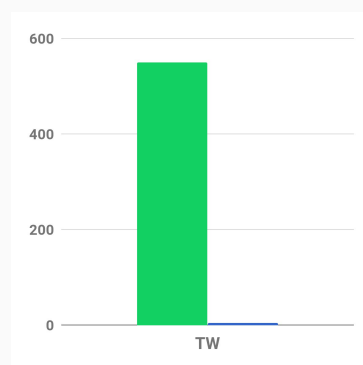
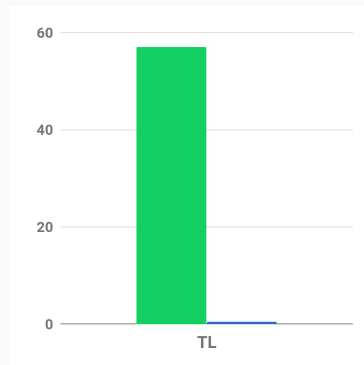
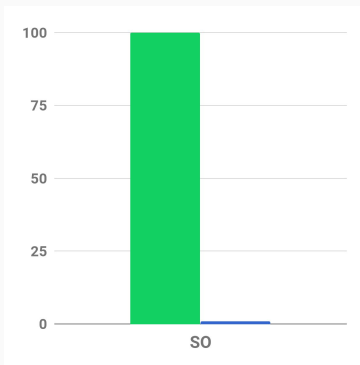
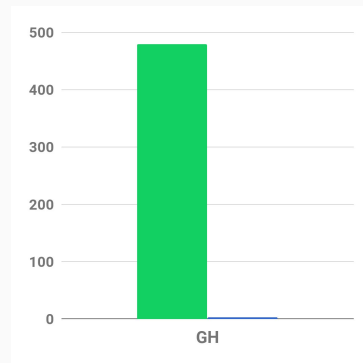
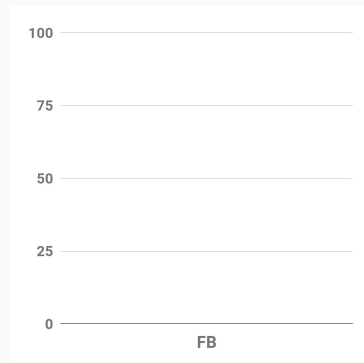
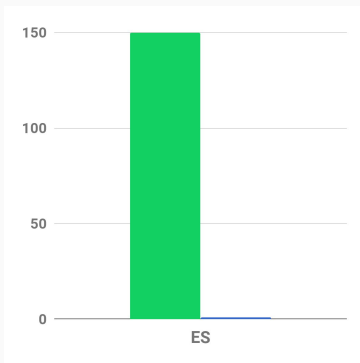


# Evaluation - Output size



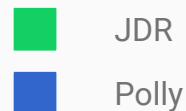
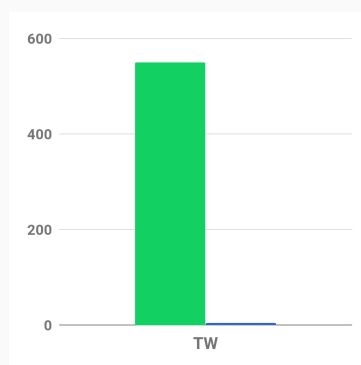
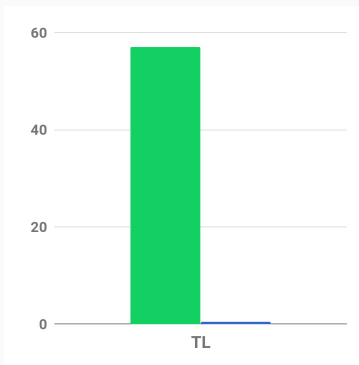
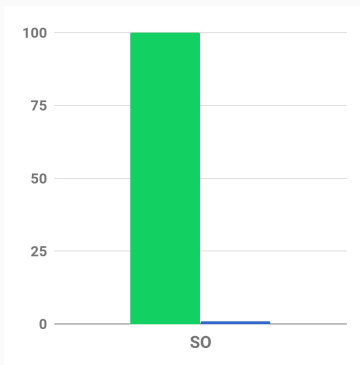
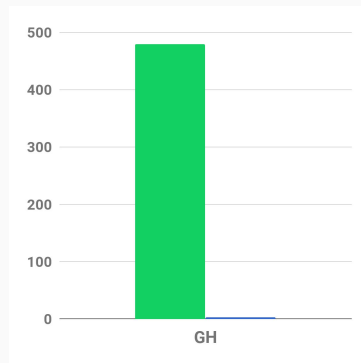
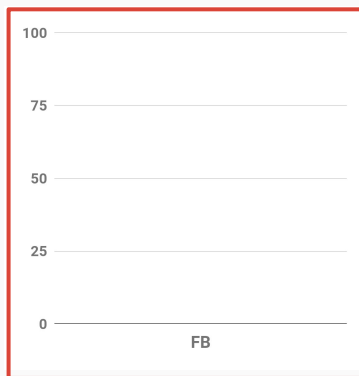
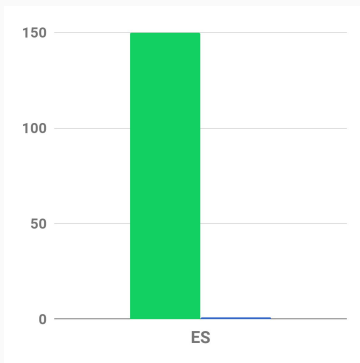
- Median values of output sizes (in bytes) per case

# Evaluation - Output size



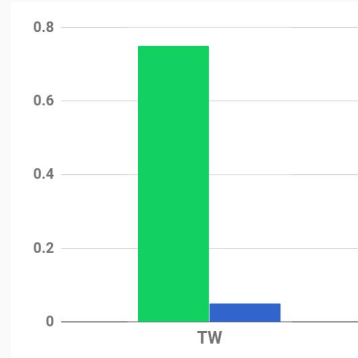
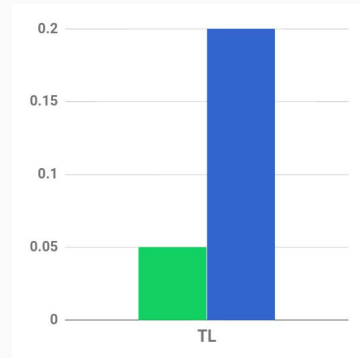
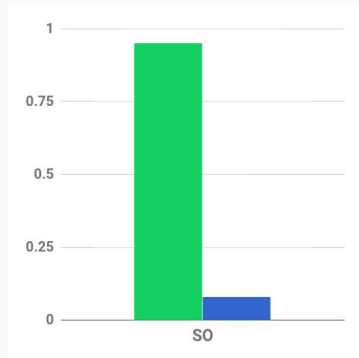
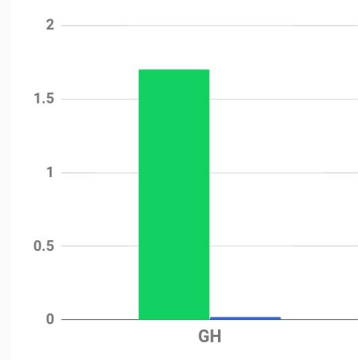
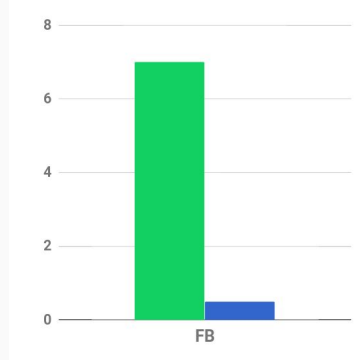
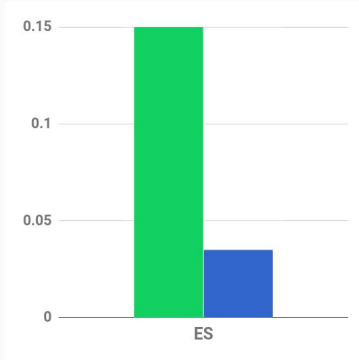
- Median values of output sizes (in bytes) per case
- **Polly** outperforms **JDR** in all cases

# Evaluation - Output size



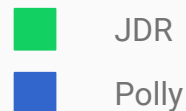
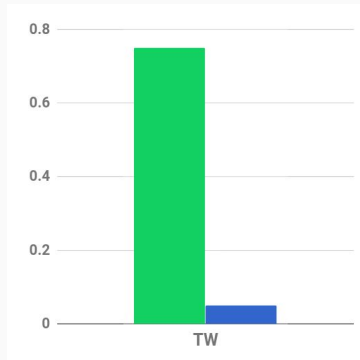
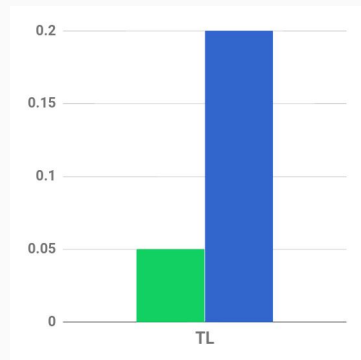
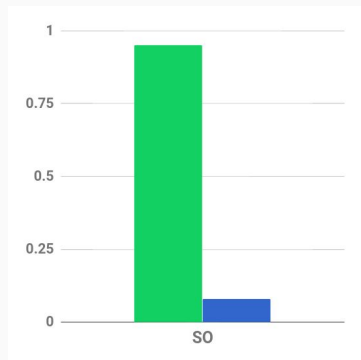
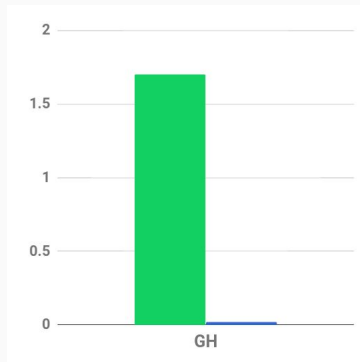
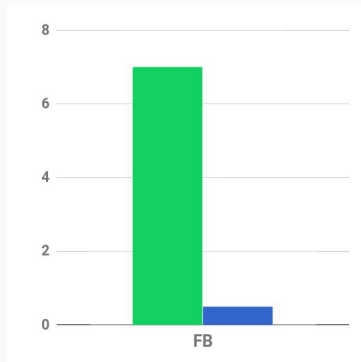
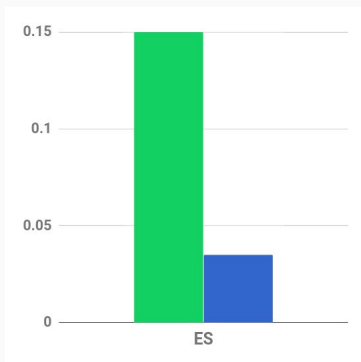
- Median values of output sizes (in bytes) per case
- **Polly** outperforms **JDR** in all cases
- **FB** scenario: **no changes** during polling period

# Evaluation - Differencing time



- Median values of diff time (in milliseconds) per case







# Evaluation - Differencing time



- Median values of diff time (in milliseconds) per case
- **Polly** outperforms JDR in almost all cases (**0.15 ms slower** for TL)







# Statistical significance tests

- One-tailed paired Wilcoxon rank test
- Cohen's  $d$  level reported on Cohen's standard scale

Scenarios	Detection time (effect size)	Output size (effect size)
 ES	Large	Medium
 FB	Large	NA
 GH	Large	Large
 SO	Large	Medium
 TL	Large	Large
 TW	Large	Large

# Statistical significance tests







- One-tailed paired Wilcoxon rank test
- Cohen's  $d$  level reported on Cohen's standard scale

Scenarios	Detection time (effect size)	Output size (effect size)
 ES	Large	Medium
 FB	Large	NA
 GH	Large	Large
 SO	Large	Medium
 TL	Large	Large
 TW	Large	Large

- **Polly** results in a significantly improved outcome compared to JDR

# Statistical significance tests

- One-tailed paired Wilcoxon rank test
- Cohen's  $d$  level reported on Cohen's standard scale

Scenarios	Detection time (effect size)	Output size (effect size)
 ES	Large	Medium
 FB	Large	NA
 GH	Large	Large
 SO	Large	Medium
 TL	Large	Large
 TW	Large	Large

- **Polly** results in a significantly improved outcome compared to JDR
- **FB**: output size is equal to 0 for every polling step for both approaches.



# Conclusion

- **Polly**, a declarative DSL for custom change detection in web service data



Language constructs for state construction  
from multiple API endpoints



Fine-grained custom  
change detection strategies



Efficient implementation,  
outperforming existing solutions



Validated on several real use-cases

# Demo

- A demo site is available at the following address:

<https://demo.pollyapp.ml>

# Polly: A Language-Based Approach for Change Detection in Web Service Data

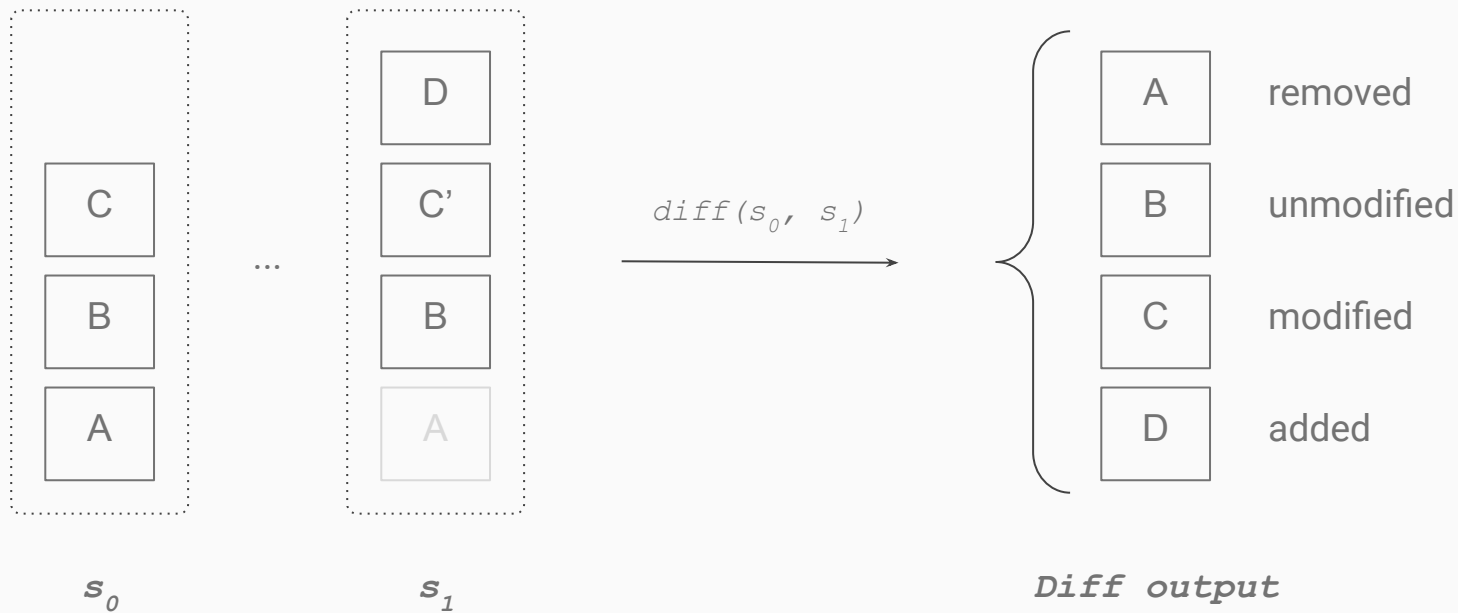
**Elyas Ben Hadj Yahia**

elyas.bhy@cprodirect.fr

16/11/17 - ICSOC - Málaga



# State differencing



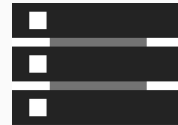
# Triggers - push mode

- Webhooks



# Triggers - push mode

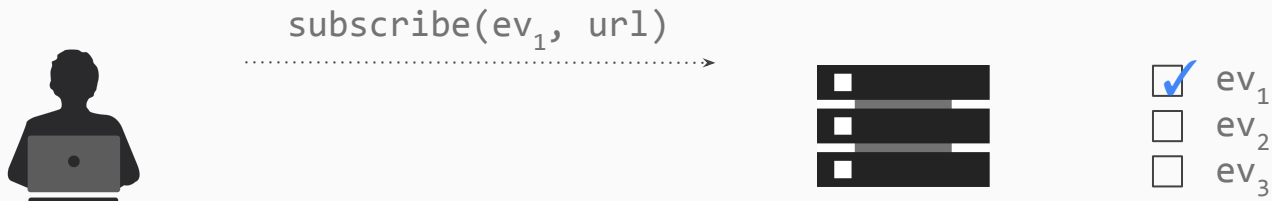
- Webhooks
  - Service providers exposes a set of predefined events



☐  $ev_1$   
☐  $ev_2$   
☐  $ev_3$

# Triggers - push mode

- Webhooks
  - Service providers exposes a set of predefined events
  - Client subscribes to the event(s) by providing a callback URL

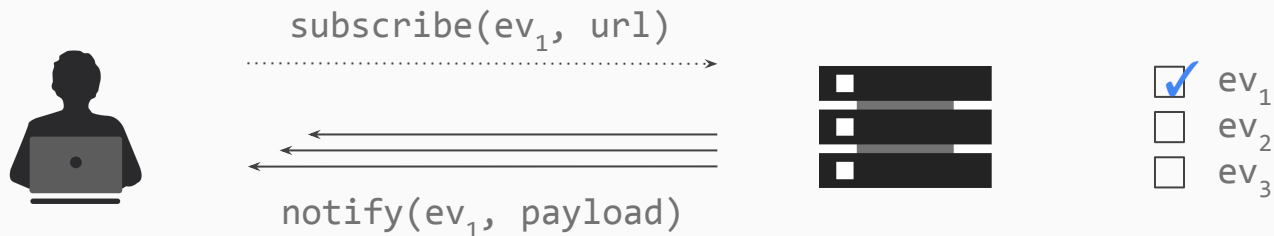




# Triggers - push mode

- Webhooks

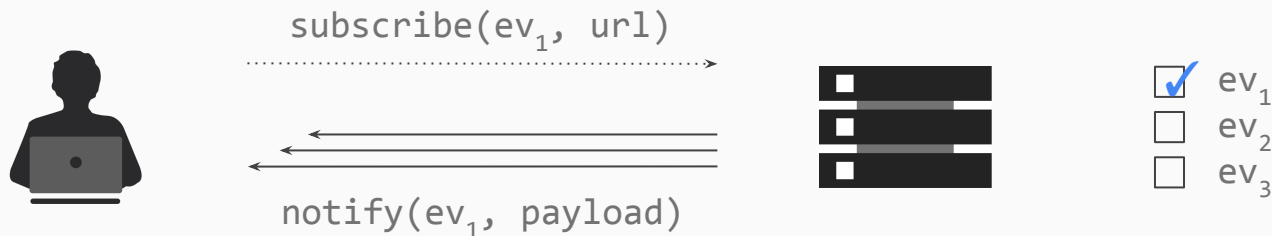
- Service providers exposes a set of predefined events
- Client subscribes to the event(s) by providing a callback URL
- Service provider calls the provided URL when the specified event occurs



# Triggers - push mode

- Webhooks

- Service providers exposes a set of predefined events
- Client subscribes to the event(s) by providing a callback URL
- Service provider calls the provided URL when the specified event occurs

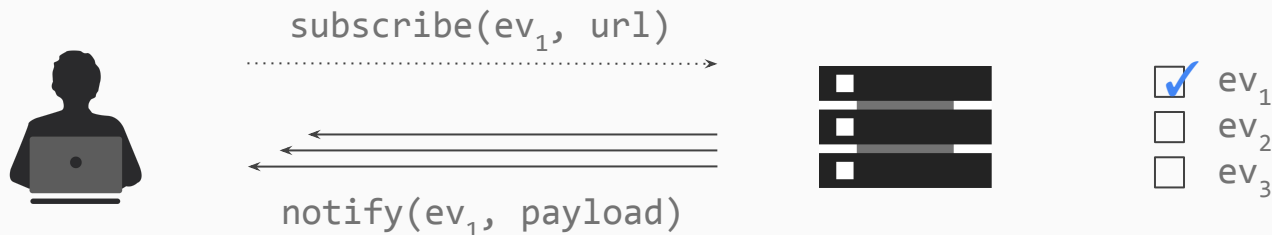


- + Realtime notification
- + No wasted network roundtrips
- + Easy to setup

# Triggers - push mode

- Webhooks

- Service providers exposes a set of predefined events
- Client subscribes to the event(s) by providing a callback URL
- Service provider calls the provided URL when the specified event occurs



- + Realtime notification
- + No wasted network roundtrips
- + Easy to setup

- Provided by very few services in practice
- Limited to predefined events

# Triggers - push mode

- Example: GitHub repository webhooks

Webhooks / **Add webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

**Payload URL \***

**Content type**

## Which events would you like to trigger this webhook?

- ☐ Just the push event.
- ☐ Send me **everything**.
- ☒ Let me select individual events.

☒ **Commit comment**  
Commit or diff commented on.

☐ **Create**  
Branch or tag created.

☐ **Delete**  
Branch or tag deleted.

☒ **Issue comment**  
Issue comment created, edited, or deleted.

☐ **Issues**  
Issue opened, edited, closed, reopened, assigned, unassigned, labeled, unlabeled, milestoned, or demilestoned.

☐ **Push**  
Git push to a repository.

# Triggers - push mode

- Example: GitHub repository webhooks

Webhooks / **Add webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

**Payload URL \***

**Content type**

## Which events would you like to trigger this webhook?

- ☐ Just the push event.
- ☐ Send me **everything**.
- ☒ Let me select individual events.

☒ **Commit comment**  
Commit or diff commented on.

☐ **Delete**  
Branch or tag deleted.

☐ **Issues**  
Issue opened, edited, closed, reopened, assigned, unassigned, labeled, unlabeled, milestoned, or demilestoned.

☐ **Create**  
Branch or tag created.

☒ **Issue comment**  
Issue comment created, edited, or deleted.

☐ **Push**  
Git push to a repository.

# Triggers - push mode

- Example: GitHub repository webhooks

Webhooks / **Add webhook**

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

**Payload URL \***

**Content type**

## Which events would you like to trigger this webhook?

- ☐ Just the push event.
- ☐ Send me **everything**.
- ☒ Let me select individual events.

☒ **Commit comment**  
Commit or diff commented on.

☐ **Delete**  
Branch or tag deleted.

☐ **Issues**  
Issue opened, edited, closed, reopened, assigned, unassigned, labeled, unlabeled, milestoned, or demilestoned.

☐ **Create**  
Branch or tag created.

☒ **Issue comment**  
Issue comment created, edited, or deleted.

☐ **Push**  
Git push to a repository.

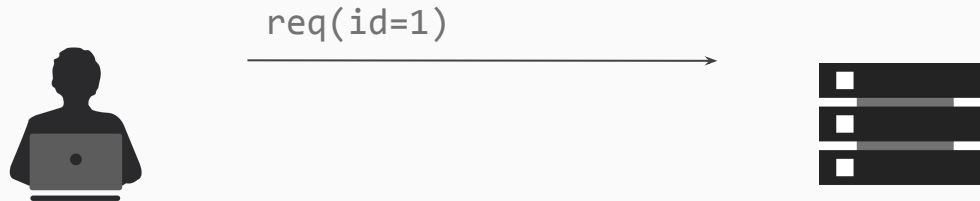
# Triggers

- Recurrent polling



# Triggers

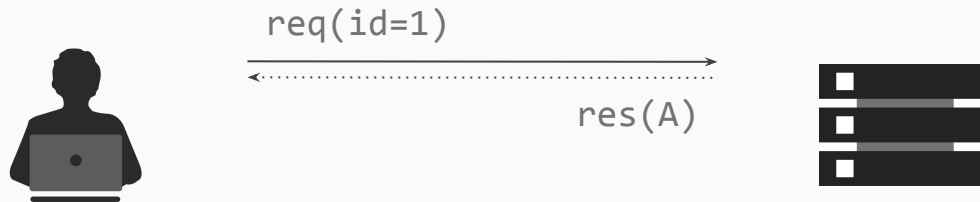
- Recurrent polling
  - Client polls a web service for a given resource





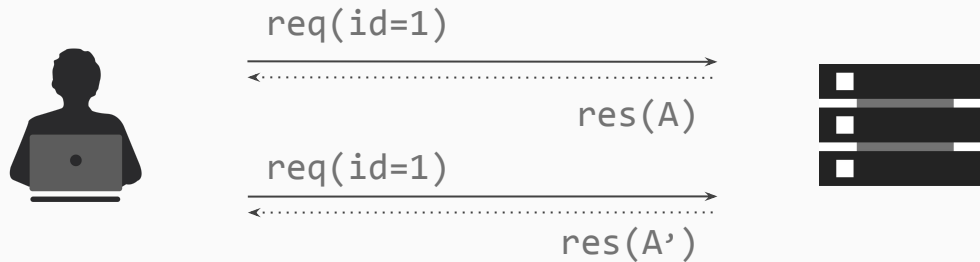
# Triggers

- Recurrent polling
  - Client polls a web service for a given resource
  - Service returns the current state of the requested resource



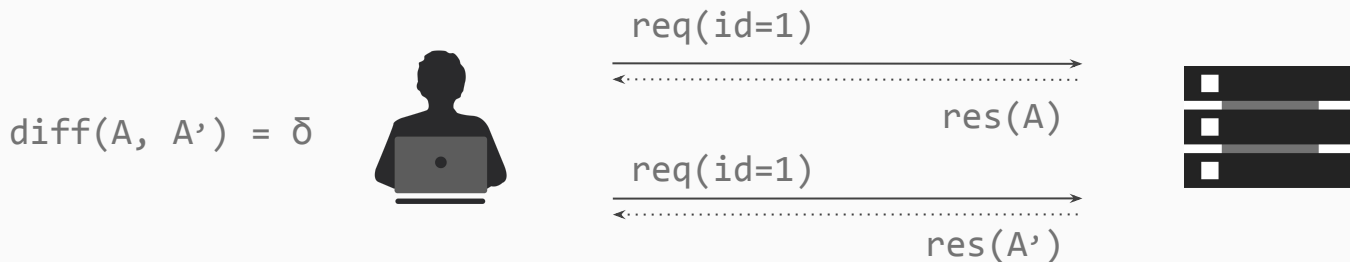
# Triggers

- Recurrent polling
  - Client polls a web service for a given resource
  - Service returns the current state of the requested resource
  - Client polls again later for the same resource



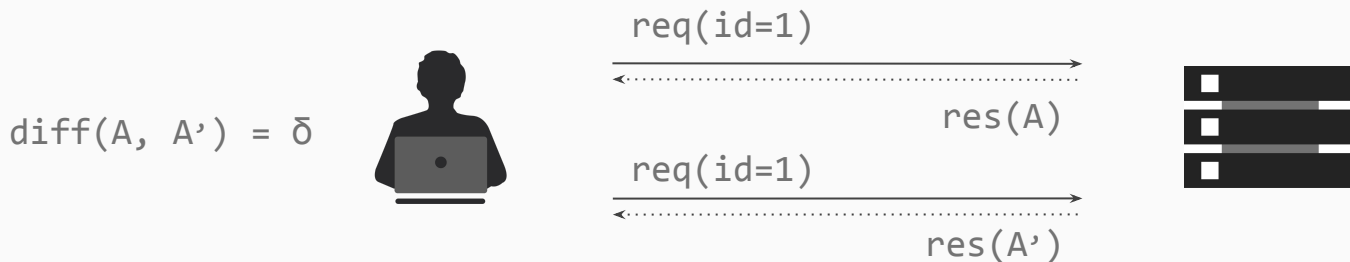
# Triggers

- Recurrent polling
  - Client polls a web service for a given resource
  - Service returns the current state of the requested resource
  - Client polls again later for the same resource
  - Client calculates the difference between both states to detect any changes



# Triggers

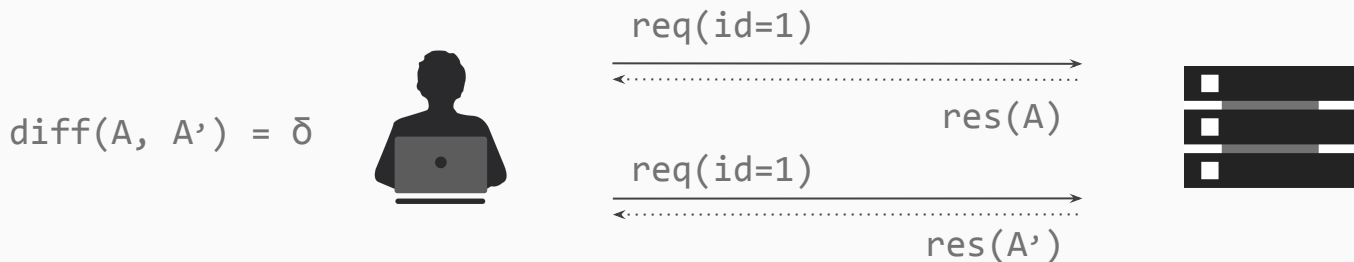
- Recurrent polling
  - Client polls a web service for a given resource
  - Service returns the current state of the requested resource
  - Client polls again later for the same resource
  - Client calculates the difference between both states to detect any changes



+ Universal approach

# Triggers

- Recurrent polling
  - Client polls a web service for a given resource
  - Service returns the current state of the requested resource
  - Client polls again later for the same resource
  - Client calculates the difference between both states to detect any changes



+ Universal approach

- Relatively complex to implement
- Tedious to support all event types