Modularity in the Era of Cloud Computing

Olivier Tardieu, Dave Grove

IBM Research

Modularity

\+ Cloud

?

Modularity

Modularity

Why?

- separation of concerns
- incremental construction
- complexity and scale
- reuse
- substitution
- distribution
- ecosystem

How?

- programming language
  - unit
  - capabilities and requirements
  - references and resolution
  - control and data flow
- beyond the language
  - distribution
  - discovery
  - configuration

Example: Node.js

loglog.js

```
let log = require('log')

function loglog (x) {
  return log(log(x))
}

module.exports = loglog
```

package.json

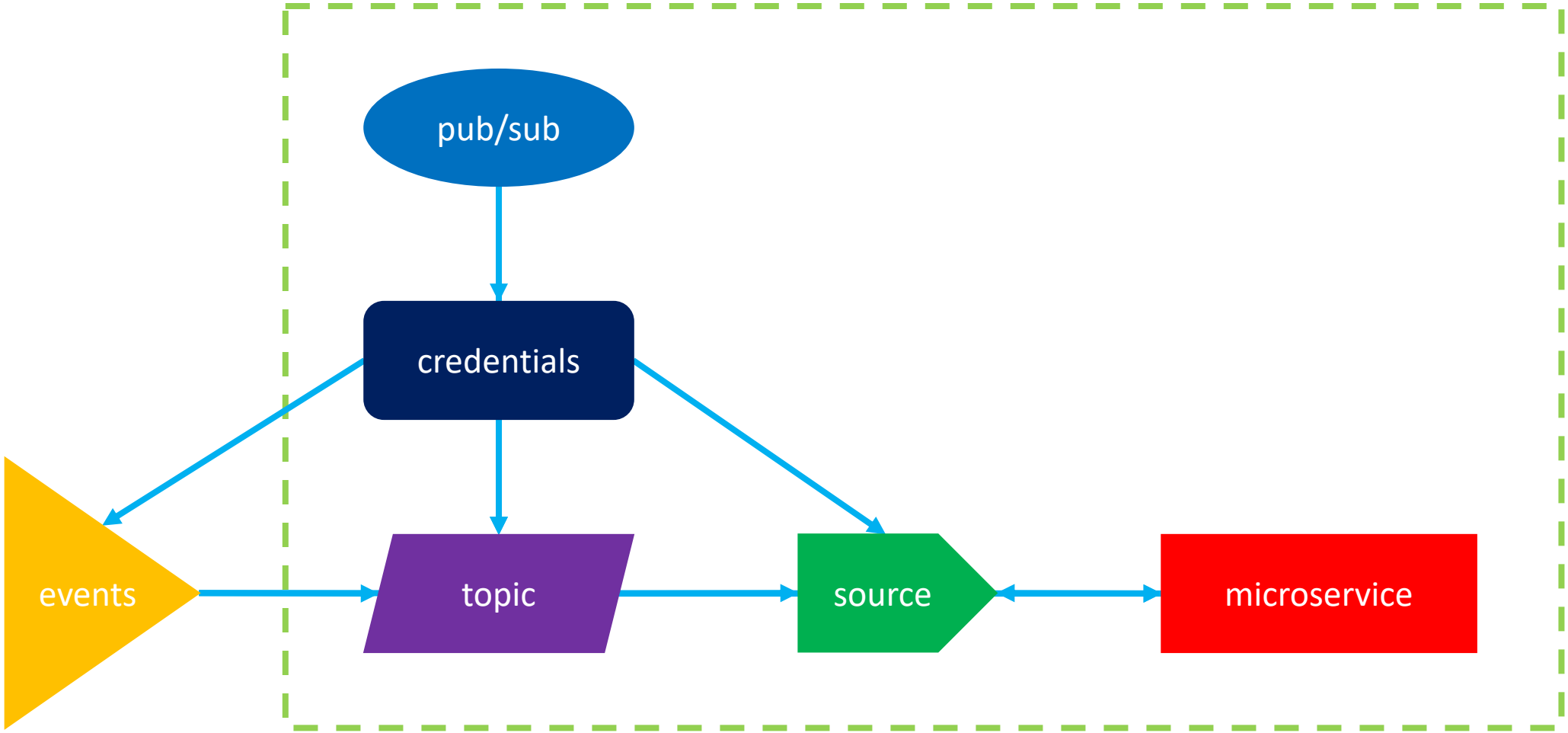
```
{
  name: 'loglog',
  version: '0.1.0',
  main: 'loglog.js',
  dependencies: {
    log: '^0.1.0'
  }
}
```

RUNNING

WIRING

Cloud

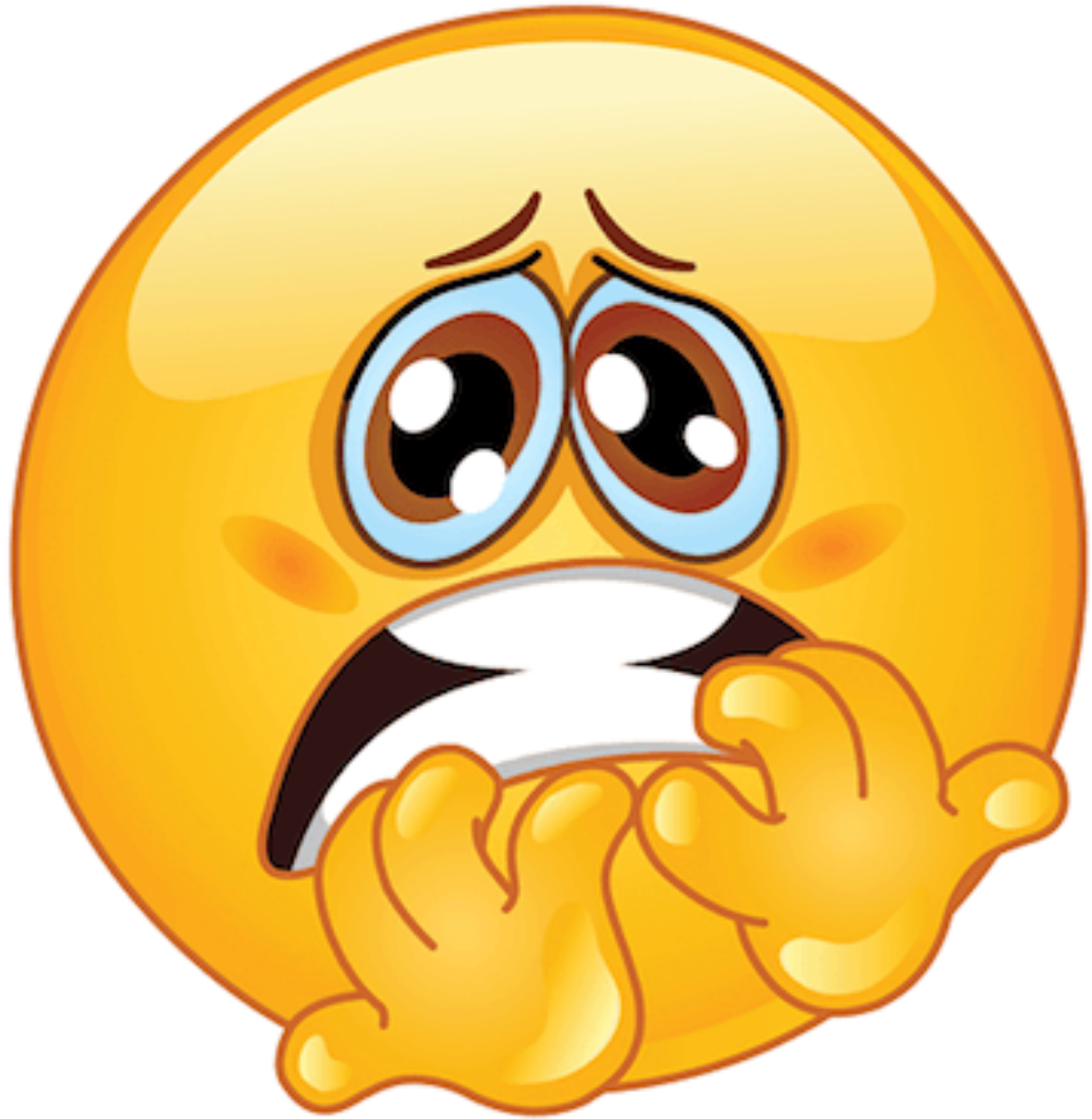
Example Cloud Composition



Kubernetes YAML

```yaml
apiVersion: ibmcloud.ibm.com/v1alpha1
kind: Service
metadata:
  name: my-kafka
spec:
  plan: standard
  serviceClass: messagehub
---
apiVersion: ibmcloud.ibm.com/v1alpha1
kind: Binding
metadata:
  name: my-kafka-credentials
spec:
  serviceName: my-kafka
---
apiVersion: ibmcloud.ibm.com/v1alpha1
kind: Topic
metadata:
  name: my-topic
spec:
  bindingFrom:
    name: my-kafka-credentials
  topicName: MyTopic
---
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  labels:
    serving.knative.dev/visibility: cluster-local
  name: my-sink
spec:
  template:
    spec:
      containers:
      - image: event_display
---
```

```yaml
apiVersion: ibmcloud.ibm.com/v1alpha1
kind: Composable
metadata:
  name: my-source
spec:
  template:
    apiVersion: sources.eventing.knative.dev/v1alpha1
    kind: KafkaSource
    metadata:
      name: my-source
    spec:
      bootstrapServers:
        getValueFrom:
          format-transformers:
          - Base64ToString
          - JsonToObject
          - ArrayToCSString
          kind: Secret
          name: my-kafka-credentials
          path: '{.data.kafka_brokers_sasl}'
      consumerGroup: my-source
      net:
        sasl:
          enable: true
          password:
            secretKeyRef:
              key: password
              name: my-kafka-credentials
          user:
            secretKeyRef:
              key: user
              name: my-kafka-credentials
        tls:
          enable: true
      sink:
        apiVersion: serving.knative.dev/v1alpha1
        kind: Service
        name: my-sink
      topics: MyTopic
```



Kubernetes YAML

- Kubernetes is an open-source system for automating deployment, scaling, and management of *cloud* applications.

The Good

- uniform syntax
- de facto standard
- extensible
- declarative
  - assuming composable operators

The Bad

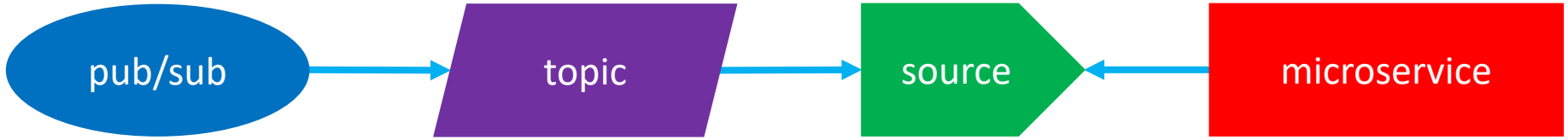
- low-level
- no semantics
- monolithic
  - no hierarchical composition
  - wiring and configuration combined

Modularity

\+ Cloud

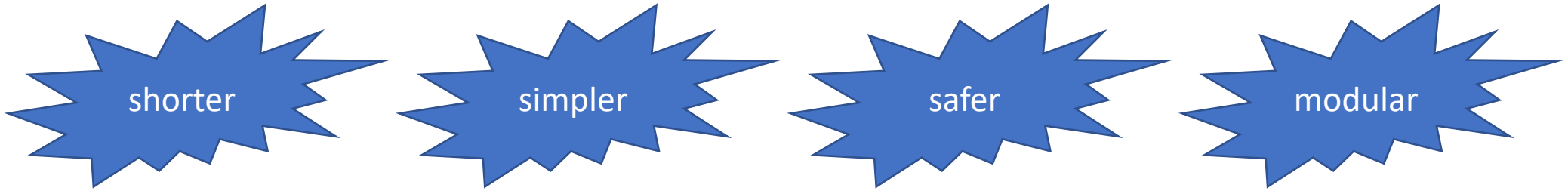
?

SolSA: Architecture as Code for the Cloud




```
import { Bundle, EventStreams, KnativeService } from 'solsa'

export = function ({ pubSubName, topicName, imageName }) {
  let pubSub = new EventStreams({ name: pubSubName, plan: 'standard' })
  let topic = pubSub.getTopic({ name: 'my-topic', topicName: topicName })
  let microservice = new KnativeService({ name: 'my-sink', image: imageName })
  let source = topic.getSource({ name: 'my-source', sink: microservice })
  return new Bundle({ pubSub, topic, source, microservice })
}
```



A Point of View

- Kubernetes YAML is an assembly language
- Kubernetes YAML does not support modular design

- A module system for the cloud needs to
  - feel familiar to developers
  - support defining reusable modules
  - handle heterogeneity: containers, VMs, functions, managed services...
  - capture capabilities and requirements
  - express sharing and understand lifecycles
  - permit secure compositions

Resources

SolSA

- Blog
  - https://medium.com/solsa
- NPM package
  - https://www.npmjs.com/package/solsa
- Source code
  - https://github.com/IBM/solsa
- Tutorial and examples
  - https://github.com/IBM/sosla-examples

Related Work

- Pulumi Kubernetes Extensions (Pulumi)
- Helm (CNCF)
- Halkyon (Red Hat)
- KubeDirector (HPE)
- CUE (Google)
- Open Application Model (Microsoft and Alibaba)