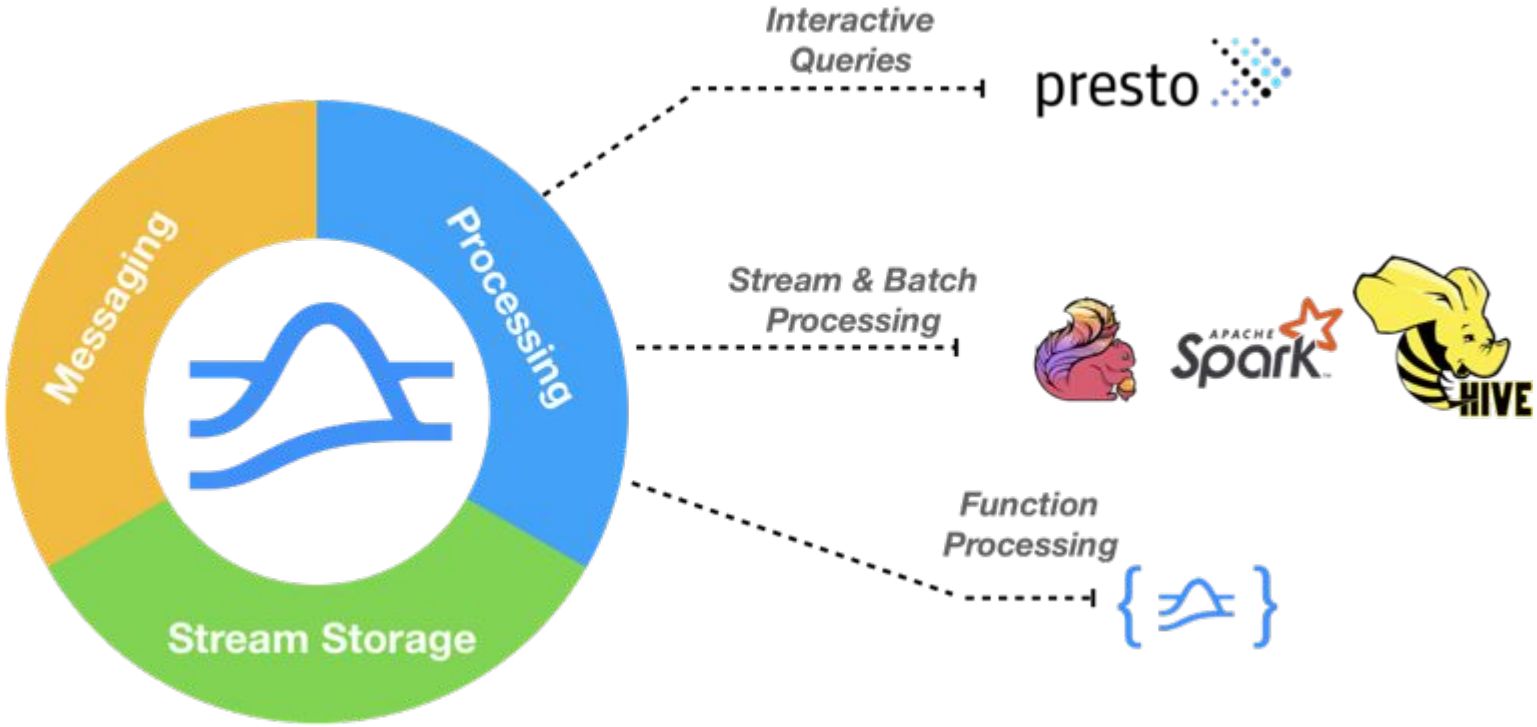
Function Mesh

Complex Streaming Jobs In A Simple Way

Neng Lu

Data Processing With Apache Pulsar



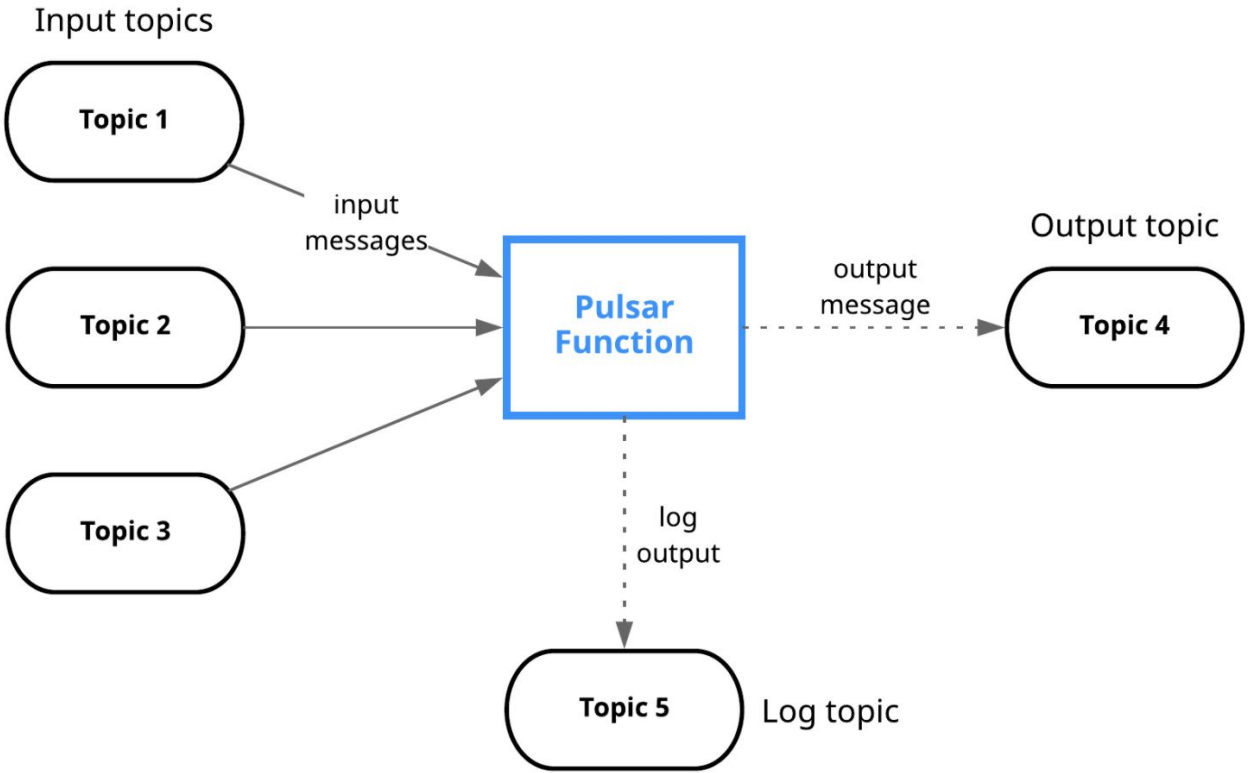
Pulsar Functions

Pulsar Functions are lightweight compute processes that:

- consume messages from Pulsar topics
- apply a **user-supplied** processing logic to each message
- publish results to another Pulsar topic

Pulsar Functions

Overview



Pulsar Functions

IS & IS NOT

Pulsar Functions **IS NOT**:

- Another Full-Power Streaming Processing Engine
- A New Computation Abstraction Layer

Pulsar Functions **IS**:

- Lambda-style functions that are specifically designed to integrate with Pulsar

Pulsar Functions

Use Case

- ETL Jobs
- Real-time Aggregation
- Microservices
- Reactive Services
- Event Routing
- ...

Pulsar Functions

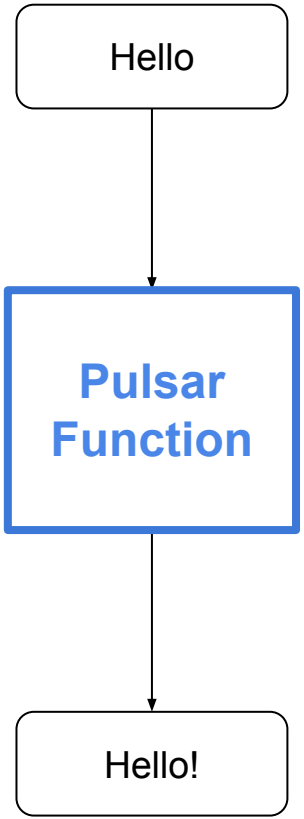
API

```java
public class ExclamationFunction implements Function<String, String> {
    @Override
    public String process(String input, Context context) {
        return String.format("%s!", input);
    }
}
```

```go
func HandleExclamation(ctx context.Context, in []byte) ([]byte, error) {
        return []byte(string(in) + "!"), nil
}
```

```python
class ExclamationFunction(Function):
  def __init__(self):
    pass

  def process(self, input, context):
    return input + '!'
```



Pulsar Functions

Semantics

- ATMOST_ONCE
  - Message is ACKed to Pulsar once received
- ATLEAST_ONCE
  - Message is ACKed to Pulsar after the function completes -- Default
- EFFECTIVELY_ONCE
  - Utilizes Pulsar’s Effectively Once Semantics

Pulsar Functions

State

- Built-In State Management
  - Provide `Context` object for users to access State
  - Stores state in Bookkeeper
  - Support Server-Side Operations like Counters

```
public class WordCountFunction implements Function<String, Void> {
    @Override
    public Void process(String input, Context context) throws Exception {
        Arrays.asList(input.split("\\.")).forEach(word -> context.incrCounter(word, 1));
        return null;
    }
}
```

Pulsar Functions

CLI

```
$ ./pulsar-admin functions
Usage: pulsar-admin functions [options] [command] [command options]
Commands:
localrun        Run a Pulsar Function locally, rather than deploy to a Pulsar cluster)
create          Create a Pulsar Function in cluster mode (deploy it on a Pulsar cluster)
delete          Delete a Pulsar Function that is running on a Pulsar cluster
update          Update a Pulsar Function that has been deployed to a Pulsar cluster
get             Fetch information about a Pulsar Function
restart         Restart function instance
stop            Stops function instance
start           Starts a stopped function instance
status          Check the current status of a Pulsar Function
stats           Get the current stats of a Pulsar Function
list            List all Pulsar Functions running under a specific tenant and namespace
querystate      Fetch the current state associated with a Pulsar Function
putstate        Put the state associated with a Pulsar Function
trigger         Trigger the specified Pulsar Function with a supplied value
```

Pulsar Functions

Summary

- Developer productivity
  - Intuitive API: `func apply(input) output {}`
  - Multiple Language Support: Java, Python, Golang
- Operational simplicity
  - Fully Integrated with Pulsar
  - No Extra System/Service Setup Needed
- Easy troubleshooting
  - Convenient Local Runtime
  - Easy to Use log topics

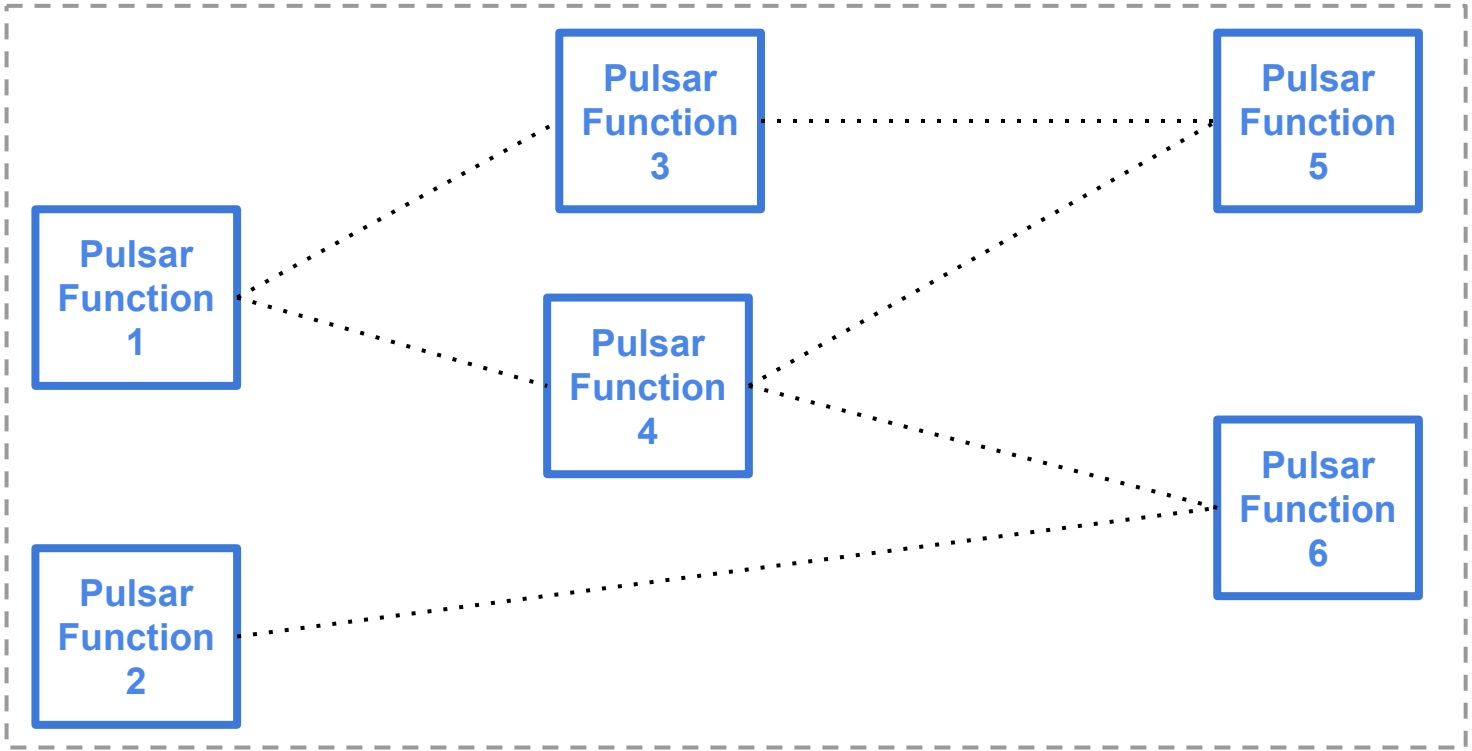
Function Mesh

Function Mesh is a collection of functions collaborate together to:

- Accomplish a final data processing goal
- With clearly defined stages

Function Mesh **IS NOT** full power Streaming Engines

Function Mesh

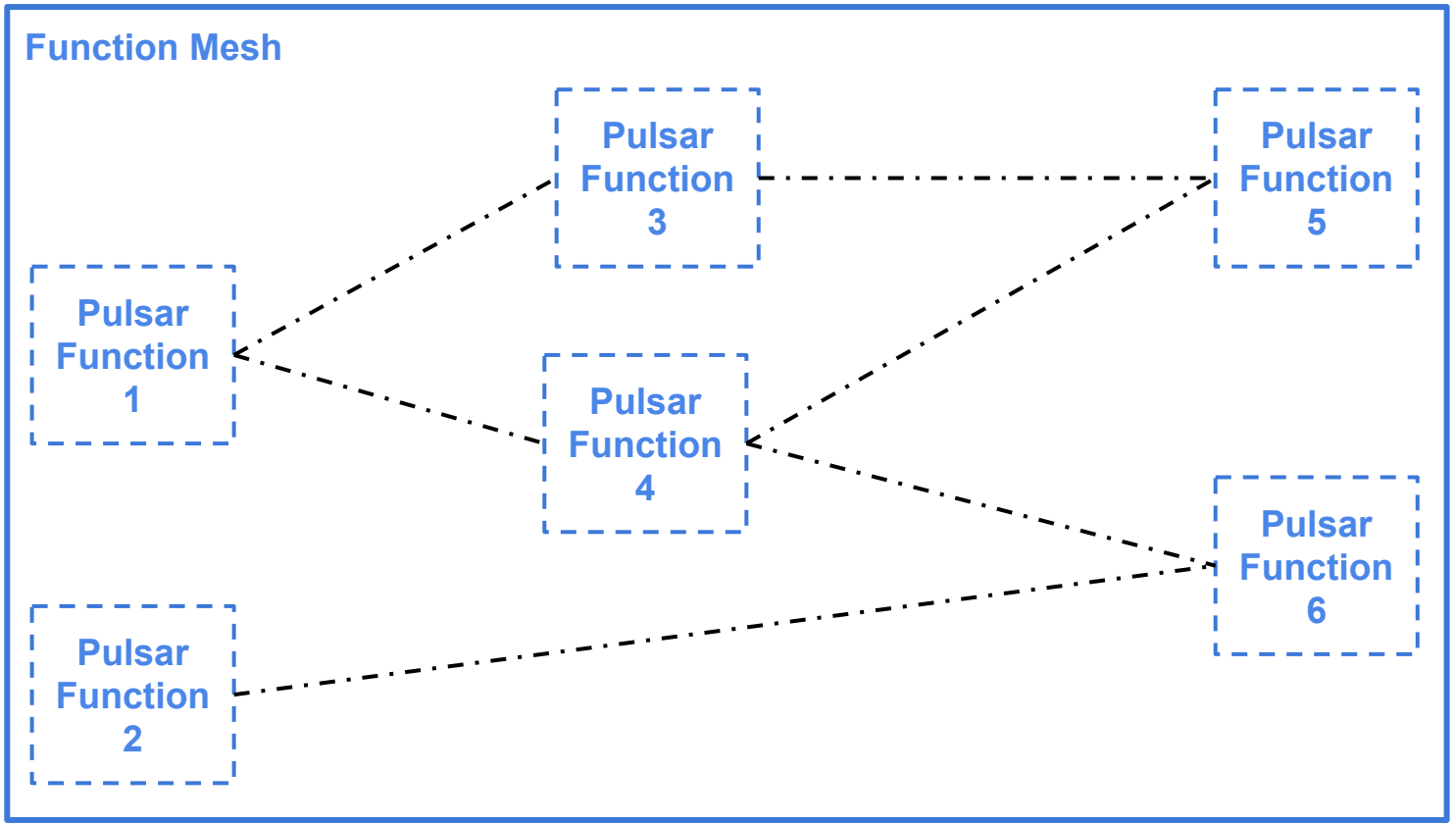


Function Mesh -- BUT

```
$ ./pulsar-admin functions create f1 ...
....
$ ./pulsar-admin functions create f2 ...
....
$ ./pulsar-admin functions create f3 ...
....
$ ./pulsar-admin functions create f4 ...
....
$ ./pulsar-admin functions create f5 ...
....
$ ./pulsar-admin functions create f6 ...
....
```

- Redundant to manage
- Hard to track as an integrity
- Really difficulty to know upstream/downstream functions

Function Mesh



Pulsar Native Function Mesh

PIP-66

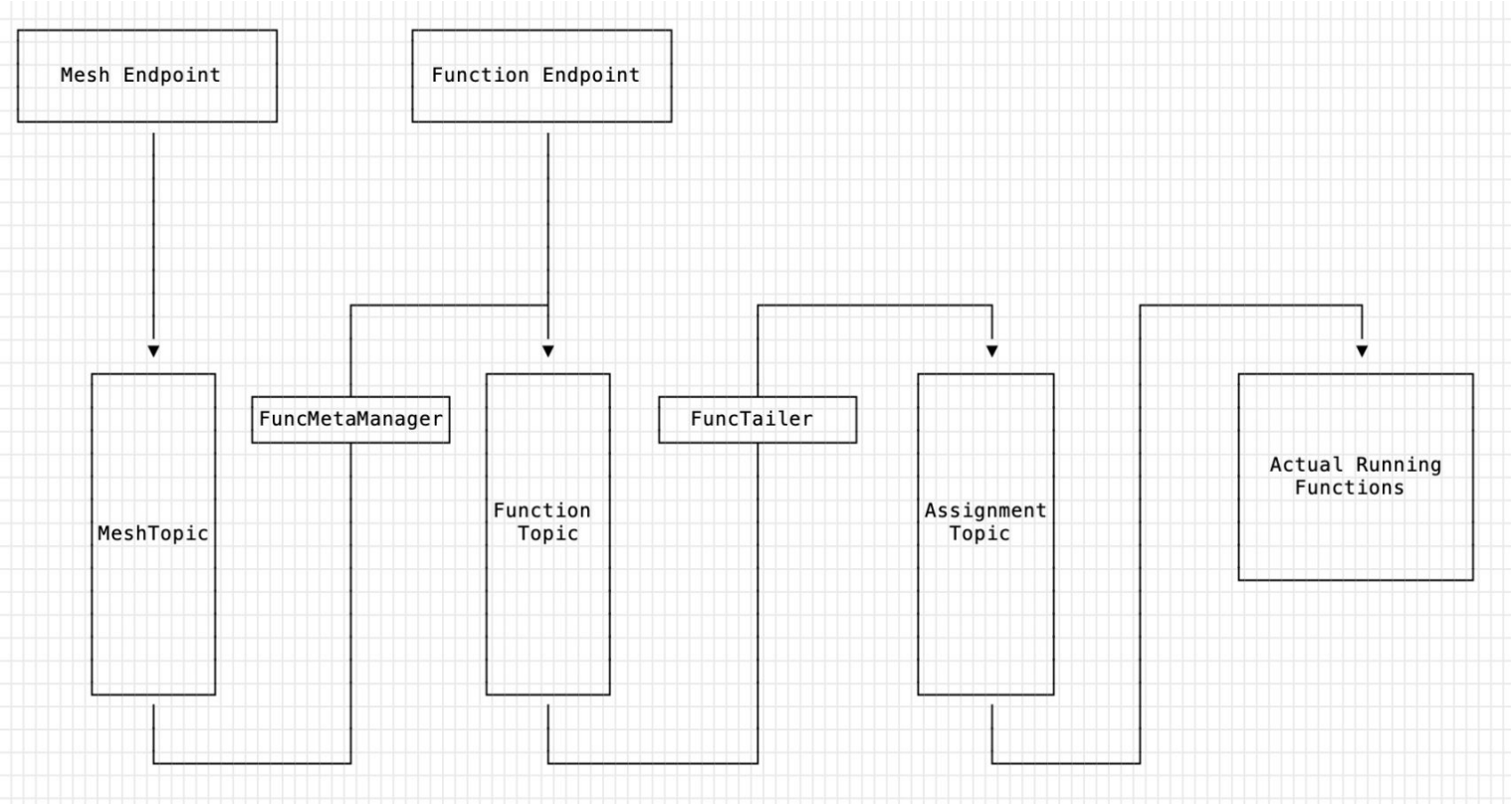
```
bin/pulsar-admin function-mesh create -f mesh.yaml
```

```yaml
# Metadata
name: PIP_Mesh
namespace: PIP_Namespace
tenant: PIP_Tenant

# Function Mesh configs
jarFile: /local/jar/files/example.jar

# Functions
functionInfos:
  - name: Func1
    classname: org.apache.pulsar.functions.api.examples.ExclamationFunction
    replicas: 1
    inputs:
    - pulsar_topic_sourcce
      output:
      - pulsar_topic_1
  - name: Func2
    classname: org.apache.pulsar.functions.api.examples.ExclamationFunction
    replicas: 1
    inputs:
    - pulsar_topic_1
      output:
      - pulsar_topic_result
```

Function Mesh -- Scheduling



Kubernetes Function Mesh

```
$ kubectl apply -f function-mesh.yaml
....
```

```yaml
apiVersion: cloud.streamnative.io/v1alpha1
kind: FunctionMesh
metadata:
  name: functionmesh-sample
spec:
  functions:
    - name: f1
      ...
    - name: f2
      ...
    - name: f3
      ...
    - name: f4
      ...
    - name: f5
      ...
    - name: f6
      ...
```

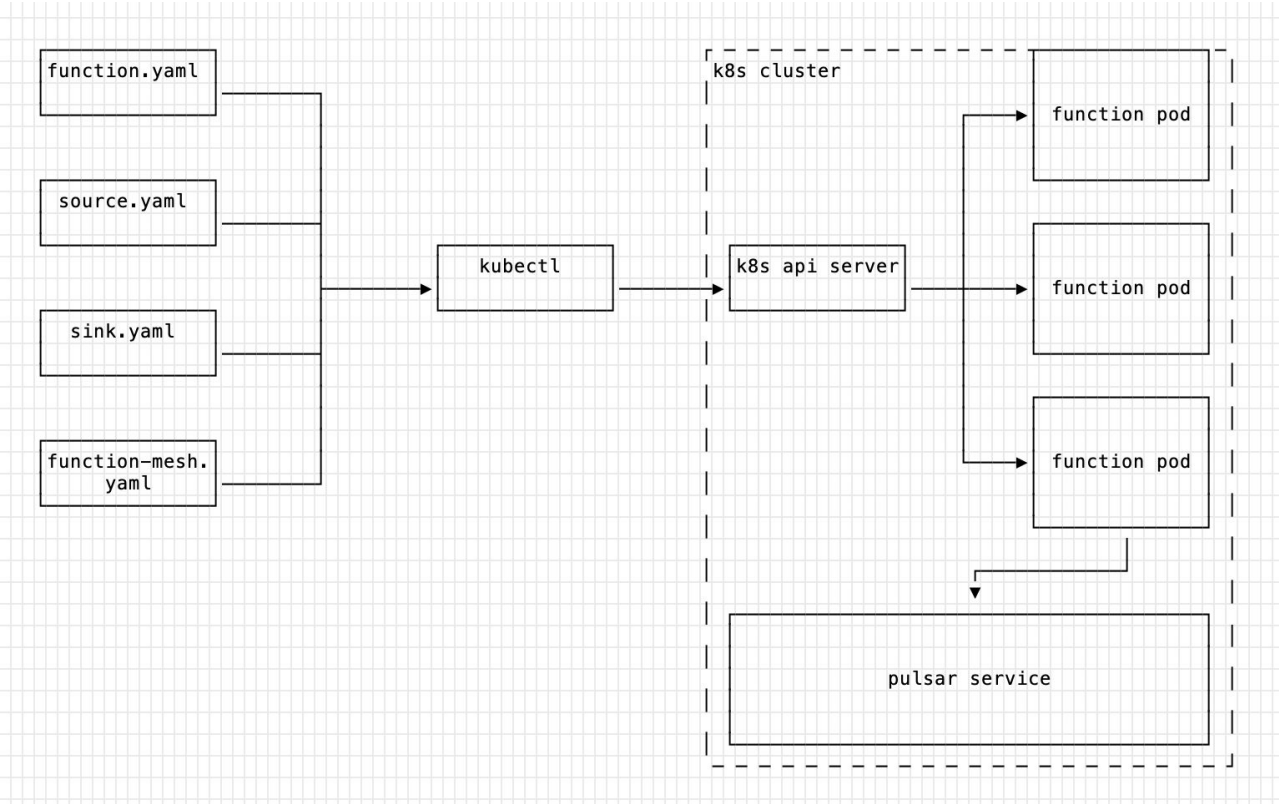
Kubernetes Function Mesh

- Custom Resource
  - Function
  - Mesh
  - Source
  - Sink

```
type FunctionSpec struct {
        Name                string   `json:"name,omitempty"`
        ClusterName         string   `json:"clusterName,omitempty"`
        Tenant              string   `json:"tenant,omitempty"`
        Namespace           string   `json:"namespace,omitempty"`
        ClassName           string   `json:"className,omitempty"`
        SourceTypeClass     string   `json:"sourceTypeClass,omitempty"`
        SinkTypeClass       string   `json:"sinkTypeClass,omitempty"`
        Replicas            int32    `json:"replicas,omitempty"`
        Sources             []string `json:"sources,omitempty"`
        Sink                string   `json:"sink,omitempty"`
        LogTopic            string   `json:"logTopic,omitempty"`
        FunctionPackage     string   `json:"functionPackage,omitempty"`
        PackageDownloadPath string   `json:"packageDownloadPath,omitempty"`
        Runtime             `json:",inline"`
}
```

```
type FunctionMeshSpec struct {
        Sources   []SourceSpec   `json:"sources,omitempty"`
        Sinks     []SinkSpec     `json:"sinks,omitempty"`
        Functions []FunctionSpec `json:"functions,omitempty"`
}
```

Kubernetes Function Mesh -- Scheduling

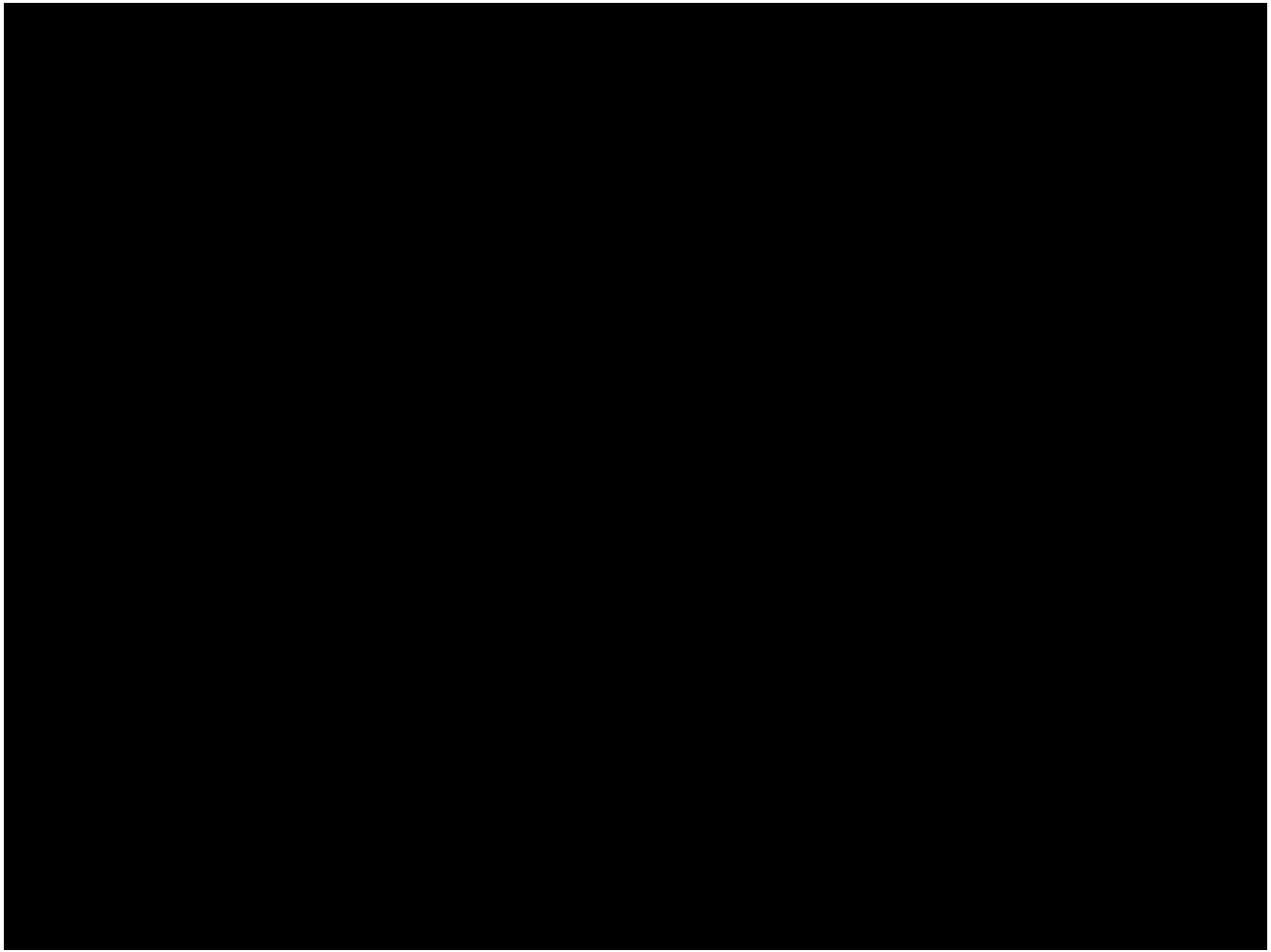


Pulsar v.s. Kubernetes

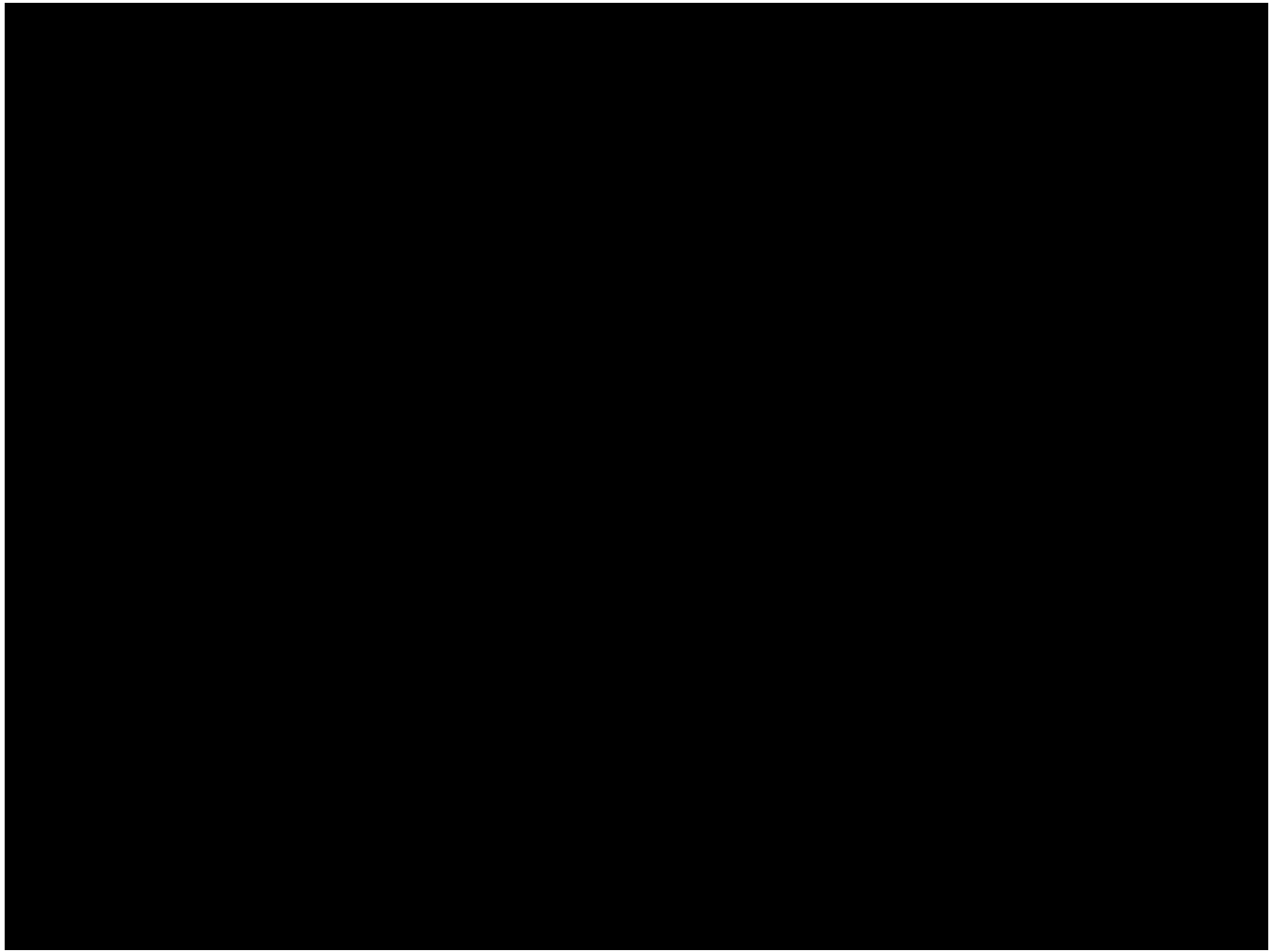
- Utilizes Kubernetes’ Scheduling Power
- Function as a First Class Citizen in Cloud Environment
- Open the potential talking to different messaging system

Demo

Function & Function Mesh



Connector - Source



Future Planning

- Cloud Native Support
- Self-Contained Function Runtime
- Function registry for reusing function unit
- Better Tools/Frontend to Manage & Inspect Function Meshes
- Smartly Group Function Unit Together If No Shuffle Used
- Auto-scaling based on Pulsar metrics

Thank You!