# servicecomb-kie Documentation

The Apache Software Foundation

Feb 18, 2021

# User Documentations


**1 Introductions** **1**
- 1.1 What is ServiceComb kie . . . 1
- 1.2 Why use ServiceComb kie . . . 1
- 1.3 Concepts . . . 1
  - 1.3.1 labels . . . 1
  - 1.3.2 key value . . . 2
  - 1.3.3 revision . . . 2

**2 Get started** **3**
- 2.1 Quick start . . . 3
  - 2.1.1 With docker . . . 3
  - 2.1.2 Run locally with Docker compose . . . 4
  - 2.1.3 Run on kubernetes . . . 4
- 2.2 After running . . . 4

**3 User guides** **7**
- 3.1 Storage . . . 7
  - 3.1.1 Options . . . 7
  - 3.1.2 Example . . . 7
- 3.2 Use discovery service . . . 8
  - 3.2.1 Options . . . 8
  - 3.2.2 Example . . . 8
- 3.3 Long polling . . . 8
  - 3.3.1 event payload and trigger condition . . . 9

**4 Development guide** **11**
- 4.1 How to Build . . . 11

CHAPTER 1

# Introductions

## 1.1 What is ServiceComb kie

ServiceComb kie is a config server which manage configurations in a distributed system. It is also a micro service in ServiceComb ecosystem and developed by go-chassis we call it ServiceComb Native application.

## 1.2 Why use ServiceComb kie

Since it is a ServiceComb Native application, the user of ServiceComb is able to integrate with it easily, or even replace some implements of kie to develop a config server on-premise.

## 1.3 Concepts

### 1.3.1 labels

key value must belong to an identical label, a labels is equivalent to a map, it is represent as a json object

```
{
"app": "some_app",
"service": "web",
"environment": "production",
"version": "1.0.1"
}
```

for each unique label map, kie will generate a label id for it and produce a db record.

### 1.3.2 key value

A key value is usually a snippet configuration for your component, let's say a web UI widget should be enabled or not. But usually, a component has different version and deployed in different environments. the labels is indicates where this component located in. below the labels indicates it is located in production environment,the version is 1.0.1

```
{
 "id": "05529229-efc3-49ca-a765-05759b23ab28",
 "key": "enable_a_function",
 "value": "true",
 "value_type": "text",
 "create_revision": 13,
 "update_revision": 13,
 "labels": {
     "app": "some_app",
     "service": "web",
     "environment": "production",
     "version": "1.0.1"
  }
}
```

### 1.3.3 revision

kie holds a global revision number it starts from 1, each creation or update action of key value record will cause the increasing of this revision number, key value has a immutable attribute "create_revision" when first created. after each modify, "update_revision" will increase. kie leverage this revision to reduce network cost, when client polling for key values, it can give a revision number "?revision=200" in query parameter, server side compare current revision with it , if they are the same, server will only return http status 304 to client. at each query server will return current revision to client with response header "X-Kie-Revision"

CHAPTER 2

# Get started

## 2.1 Quick start

### 2.1.1 With docker

Run mongodb server

use the db init script

```
sudo docker run --name mongo -d \
    -e "MONGO_INITDB_DATABASE=kie" \
    -e "MONGO_INITDB_ROOT_USERNAME=root" \
    -e "MONGO_INITDB_ROOT_PASSWORD=root" \
    -p 27017:27017 \
    -v ./deployments/db.js:/docker-entrypoint-initdb.d/db.js:ro \
    mongo:4.0
```

```
export MONGO_IP=`sudo docker inspect --format '{{ .NetworkSettings.IPAddress }}'␣
↪mongo`
```

Run kie server

```
sudo docker run --name kie-server -d \
    -e "MONGODB_ADDR=${MONGO_IP}:27017" \
    -e "MONGODB_USER=root" \
    -e "MONGODB_PWD=root" \
    -p 30110:30110 \
    servicecomb/kie
```

### 2.1.2 Run locally with Docker compose

```
git clone git@github.com:apache/servicecomb-kie.git
cd servicecomb-kie/deployments/docker
sudo docker-compose up
```

it will launch 3 components

- mongodb: 127.0.0.1:27017
- mongodb UI:http://127.0.0.1:8081
- servicecomb-kie: http://127.0.0.1:30110

### 2.1.3 Run on kubernetes

```
kubectl apply -f https://raw.githubusercontent.com/apache/servicecomb-kie/master/
↪deployments/kuberneetes/
```

it will launch 3 components, you can access them both in kubernetes and out of kubernetes. out of kubernetes:

- mongodb: ${ANY_NODE_HOST}:30112
- mongodb UI:http://${ANY_NODE_HOST}:30111
- servicecomb-kie: http://${ANY_NODE_HOST}:30110 in kubernetes:
- mongodb: servicecomb-kie-nodeport:27017
- mongodb UI: servicecomb-kie-nodeport:8081
- servicecomb-kie: servicecomb-kie-nodeport:30110

## 2.2 After running

Put a key

```
curl -X POST \
  http://127.0.0.1:30110/v1/default/kie/kv/ \
  -H 'Content-Type: application/json' \
  -d '{
    "key": "timeout",
    "value": "2s",
    "labels": {
        "service": "order"
    }
}'
```

response is

```
{
    "id": "b01ad993-2bad-4468-8a3c-5aa7ad54afea",
    "key": "timeout",
    "value": "2s",
    "value_type": "text",
    "create_revision": 2,
    "update_revision": 2,
```

(continues on next page)

(continued from previous page)

```
    "status": "disabled",
    "create_time": 1590802244,
    "update_time": 1590802244,
    "labels": {
        "service": "order"
    }
}
```

then get config list

```
curl -X GET http://127.0.0.1:30110/v1/default/kie/kv
```

response is

```
{
 "total": 1,
 "data": [
  {
   "id": "b01ad993-2bad-4468-8a3c-5aa7ad54afea",
   "key": "timeout",
   "value": "2s",
   "value_type": "text",
   "create_revision": 2,
   "update_revision": 2,
   "status": "disabled",
   "create_time": 1590802244,
   "update_time": 1590802244,
   "labels": {
    "service": "order"
   }
  }
 ]
}
```

Check open API doc

- the api doc mounted to http://127.0.0.1:30110/apidocs.json
- or see https://github.com/apache/servicecomb-kie/blob/master/docs/api.yaml

CHAPTER 3

# User guides

## 3.1 Storage

you can use mongo db as kie server storage to save configuration

### 3.1.1 Options

**uri**

*(required, string)* db uri

**timeout**

*(optional, string)* db operation timeout

**sslEnabled**

*(optional, bool)* enable TLS communication to mongodb server

**rootCAFile**

*(optional, string)* if sslEnabled is true, you must give a ca file

**verifyPeer**

*(optional, bool)* if verifyPeer is true, kie will verify database server's certificate, otherwise not

### 3.1.2 Example

```
db:
  uri: mongodb://kie:123@127.0.0.1:27017/kie
  poolSize: 10
  timeout:  5s
  sslEnabled: true
```

(continues on next page)

(continued from previous page)

```
  rootCAFile: /opt/kie/ca.crt
  verifyPeer: true
```

## 3.2 Use discovery service

kie server is able to register to a discovery service, so that other client is able to discover it registry service. by default registry feature is disabled.

```
cse:
  service:
    registry:
      disabled: true
```

For example, kie is able to register to ServiceComb service center.

### 3.2.1 Options

this feature is powered by go-chassis, refer to https://docs.go-chassis.com/user-guides/registry.html to know how to use

### 3.2.2 Example

Register to ServiceComb service center

```
cse:
  service:
    registry:
      address: http://127.0.0.1:30100
...
```

## 3.3 Long polling

*experimental*

Kie leverage gossip protocol to broad cast cluster events. if client use query parameter "?wait=5s" to poll key value, this polling will become long polling and if there is key value change events, server will response key values to client.

kie must join to a cluster and listen to peer events

start first node

```
./kie --name=kie0 --listen-peer-addr=10.1.1.11:5000
```

start another node

```
./kie --name=kie1 --listen-peer-addr=10.1.1.12:5000 --peer-addr=10.1.1.11:5000
```

### 3.3.1 event payload and trigger condition

condition: key value put or delete

payload:

```
{
  "Key": "timeout",
  "Action": "put",
  "Labels": {
    "app": "default",
    "service": "order"
  },
  "DomainID": "default",
  "Project": "default"
}
```

CHAPTER 4

# Development guide

## 4.1 How to Build

Download init and run mongodb as mentioned in get started section

Build serivcecomb-kie binary

```
cd ${project_root_path}
build/build_binary.sh
```

This will build 4 packages for different platforms in ${servicecomb_root_path}/release/kie directory. Choose the right one, unzip and run it. For Example:

```
cd ${servicecomb_root_path}/release/kie
tar -xzvf apache-servicecomb-kie--${platform}-amd64.tar.gz
cd apache-servicecomb-kie--${platform}-amd64
./kie --config conf/kie-conf.yaml
```