

1. Scheduler is Source to push the TaskLaunchRequest Payload which is pushed to AQMP

**Spring.cloud.stream.bindings.output.destination** defines the exchange which

Produces the payload.

1. **EnableTaskLauncher** TaskLauncherSink is the listener to AQMP, Launches a task upon the receipt of a valid TaskLaunchRequest.

**Spring.cloud.stream.bindings.input.destination** defines the intake queue.

**spring.cloud.stream.bindings.input.group**

**Spring.cloud.stream.bindings.input.consumer.concurrency,** defines the number of

Concurrent consumers of TaskLauncherSink.

1. @EnableTask to define the short lived individual microservice. Spring Cloud Task will contain the list of jobs which need to be performed by the task which are managed via JobFactory.
2. Spring cloud dataflow server UI display the Task and Jobs which are executed.

Download spring-cloud-dataflow-server-local-1.3.1.RELEASE.jar or any latest

Version of jar.

Run the command, to display the Spring cloud dataflow server UI:

**java -jar spring-cloud-dataflow-server-local-1.3.1.RELEASE.jar**

**--spring.datasource.url="jdbc:mysql://localhost/springcloud" --spring.datasource.username="root" --spring.datasource.password="root" --spring.datasource.driver-class-name="org.mariadb.jdbc.Driver"**

1. spring.cloud.deployer.memory for specifying "container" memory in megabytes - 1024 or 1024m are both acceptable formats
2. spring.cloud.deployer.cpu for specifying number of cpu cores.
3. **TaskConfigurer** is used to customize the way components of Spring Cloud Task are configured. By default, the DefaultTaskConfigurer provides logical defaults (Map based in memory components useful for development if no DataSource is provided and JDBC based components if there is a DataSource available.

Create database **springTask** which will stores the information regarding the Spring

Cloud task.

Create another database **test** which will stores the information regarding the application.

**Concerns:**

1. .
2. .
3. .