1. We need to first Install RabbitMQ in our local system(you can use Docker for rabbitmq installation as well) , below are the steps to install it.

1) Download and Install Erlang (OTP\_win32\_21.0.1)

2)Download and Install RabbitMQ(3.7.7)

3) Once installed rabbitmq it will be start by default.

To check rabbitmq management console use below command to install plugin in rabbitmq’s command prompt.

**rabbitmq-plugins enable rabbitmq-management**

4)You can use below url to access rabbitmq management UI

<http://localhost:15672/>

username:guest

password:guest

II)To make the Cloud Config Server application as Cloud Bus we need to add following dependencies in Config Server and Client Application.

1. Config Server

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-stream-rabbit</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-config-monitor</artifactId>

</dependency>

1. Config Client

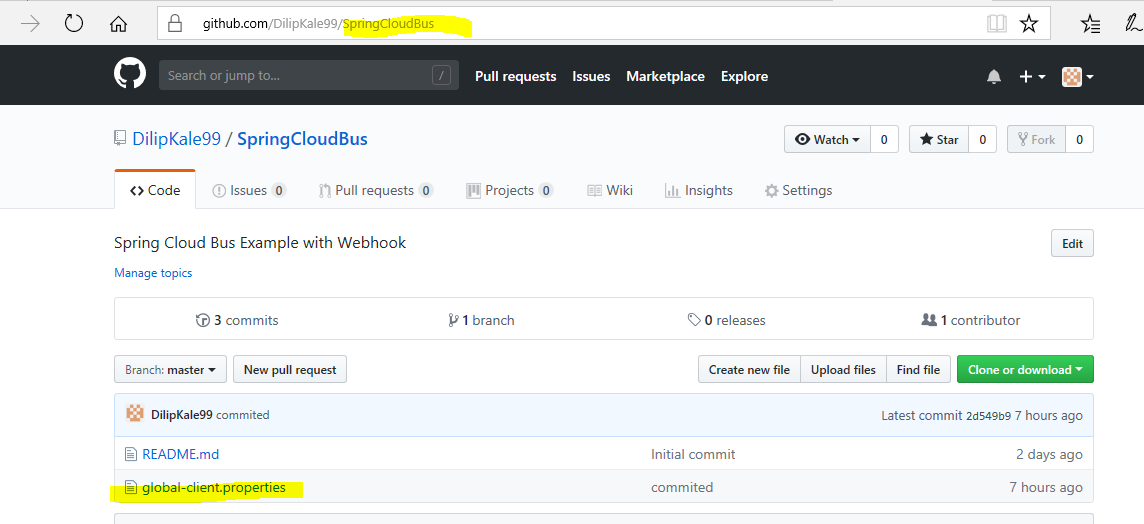
<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-bus-amqp</artifactId>

</dependency>

III) We need to add global-client.properties file in GitHub Repository First , So that we can access property file from GITHUB.



IV) We need to make add few details in properties files of Server and Client Application

1)Config Server(application.properties)

spring.rabbitmq.host=localhost

spring.rabbitmq.port=5672

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

spring.cloud.config.server.monitor.github.enabled=true

spring.cloud.config.server.git.uri=https://github.com/DilipKale99/SpringCloudBus

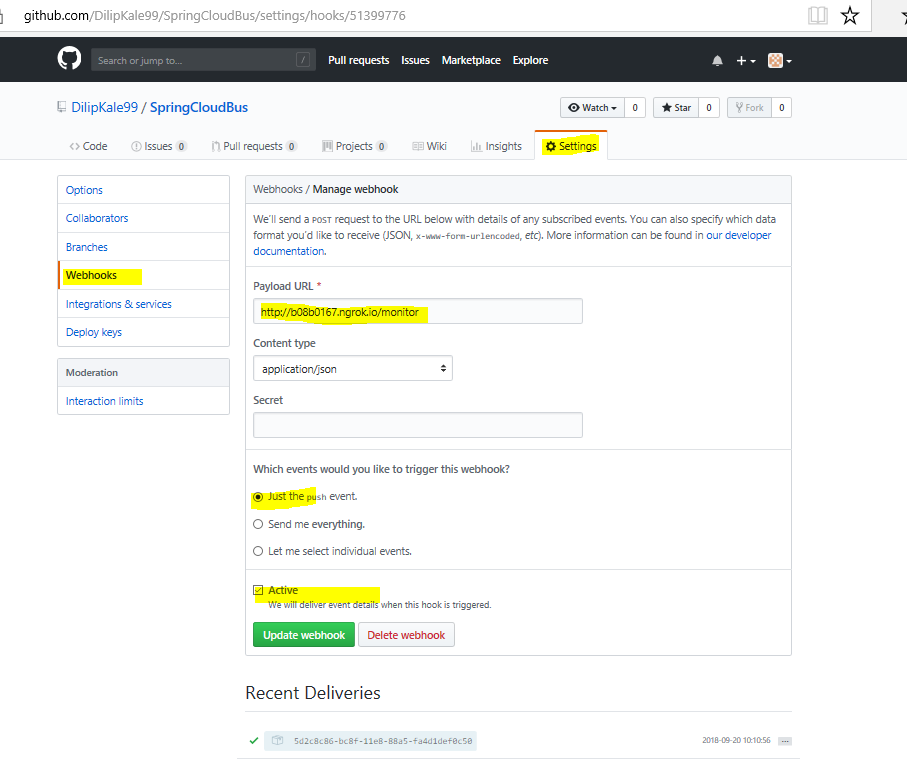
Properties file URI must change from local file repo to Github file repo.

V) Our Config Server application in running on 8888 port in local system , To access that from GITHUB we need use tool called ngrok. You can download that from url <https://ngrok.com/>

Unzip the downloaded file and click on .exe file which will open command prompt.  
trigger below command on that

**ngrok http 8888**

It will create a public URL which we can use to setup our webhook in Github like this:



Payload url created by ngrok so that we can access our localhost::8888 application from GITHUB/Outside world. We need to add **/monitor** endpoint in the payload url i.e

<http://b08b0167.ngrok.io/monitor>

VI) We can now start Server application and Client application.

And trigger Client application using url <http://localhost:8091/message>

This will show current content from Github property file.  
Now change the file content and commit the code in Github.

Trigger client application once again we can see updated value on browser.

This happened without triggering any /refresh endpoint.

We can create multiple client instances to check whether it’s reflecting on all instances.

Thanks