Producer - method:

```
  /* async uploadS3EnvelopeTimed(businessEventInstanceId, clientId, envelope)
{
    this.logger.debug('uploading envelope');

    const { isClientSharded } = await DBShards.getShard(clientId);
    const keysArray = [];
    const redisClient = AWSResource.getRedisClient();
    const redisKey = `bus:${keyPrefix}${clientId}:migration`;
    const findKey = await redisClient.findKeys(redisKey);
    console.log('kkkkkkkkkk', findKey, isClientSharded, redisKey);
    if (isClientSharded && findKey.length) {
      console.log('cccccccc');
      keysArray.push(`${clientId}/${INSTANCE_PREFIX}/$
{businessEventInstanceId}`, businessEventInstanceId);
      console.log('aaaaaaa', keysArray);
      keysArray.forEach(async (key) => { //mapseries
        this.logger.info('Uploading envelop with key: ', key);
        const body = JSON.stringify(envelope);
        try {
          const result = await this.s3Client.upload({
            key,
            body,
            contentType: 'application/json',
            tags: { clientId },
          });
          return result;
        } catch (err) {
          statsDClient.increment('error_uploading_to_s3_count');
          this.logger.error('failed to upload envelope');
          this.logger.sensitive.error('failed to upload envelope details', err);
          throw err;
        }
      });
    }

    // const key = (isClientSharded && !findkey)
    //   ? `${clientId}/${INSTANCE_PREFIX}/${businessEventInstanceId}`
    //   : businessEventInstanceId;
  } */
```

Now :

```
 async uploadS3EnvelopeTimed(businessEventInstanceId, clientId, envelope) {
   this.logger.debug('uploading envelope');

   const { isClientSharded } = await DBShards.getShard(clientId);
   // const isClientSharded = true;
   const redisClient = AWSResource.getRedisClient();
   const redisKey = `bus:${keyPrefix}${clientId}:migration`;
   const findKey = await redisClient.findKeys(redisKey);
   console.log('sssssssssss', isClientSharded, redisKey, findKey);
   if (isClientSharded && findKey.length) {
     console.log('111111111');
     const result = await Promise.all([this.random(`${clientId}/$
{INSTANCE_PREFIX}/${businessEventInstanceId}`, envelope, clientId),
this.random(businessEventInstanceId, envelope, clientId)]);
     console.log('222222', result);
     return result;
   }
   const key = isClientSharded
     ? `${clientId}/${INSTANCE_PREFIX}/${businessEventInstanceId}`
     : businessEventInstanceId;
   console.log('3333333', key);
   const result = await this.random(key, envelope, clientId);
   console.log('4444444', result);
   return result;
 }

 async random(key, envelope, clientId) {
   this.logger.info('Uploading envelop with key: ', key);
   const body = JSON.stringify(envelope);
   try {
     console.log('7777777', key);
     const result = await this.s3Client.upload({
       key,
       body,
       contentType: 'application/json',
       tags: { clientId },
     });
     return result;
   } catch (err) {
     statsDClient.increment('error_uploading_to_s3_count');
     this.logger.error('failed to upload envelope');
```

```
    this.logger.sensitive.error('failed to upload envelope details', err);
    throw err;
  }
}
```