Dmaap adapter

Current Observations with Dmaap Adapter Scalability

- When multiple replicas of DmaapAdapter are created they register with ICS with same jobCallbackUrl, producerSupervisionCallbackUrl. Due to this, when a new job is created, ICS sends job creation messages to the dmaapAdapter, and the Kubernetes service, acting as a load balancer, distributes the calls to multiple replicas of the dmaapAdapter.
  This causes inconsistency of Job information in different replicas of dmaapAdapter.
  Some jobs may get created in dmaapAdapter-0 and some may be created in dmaapAdapter-1.

- Similarly, due to data inconsistencies ICS may send job deletion message to instance of dmaapAdapter which may not have job information.

- There is no common repository or DB where the job information is shared across dmaapAdapters

- In H-release, with the support of PM data PM files are stored in volumes. Currently DmaapAdapter is designed as a stateful deployment. So, the PM files may not be shared across all the replicas of DmaapAdapter

- Once error is encountered, Data consumer stops without retrying

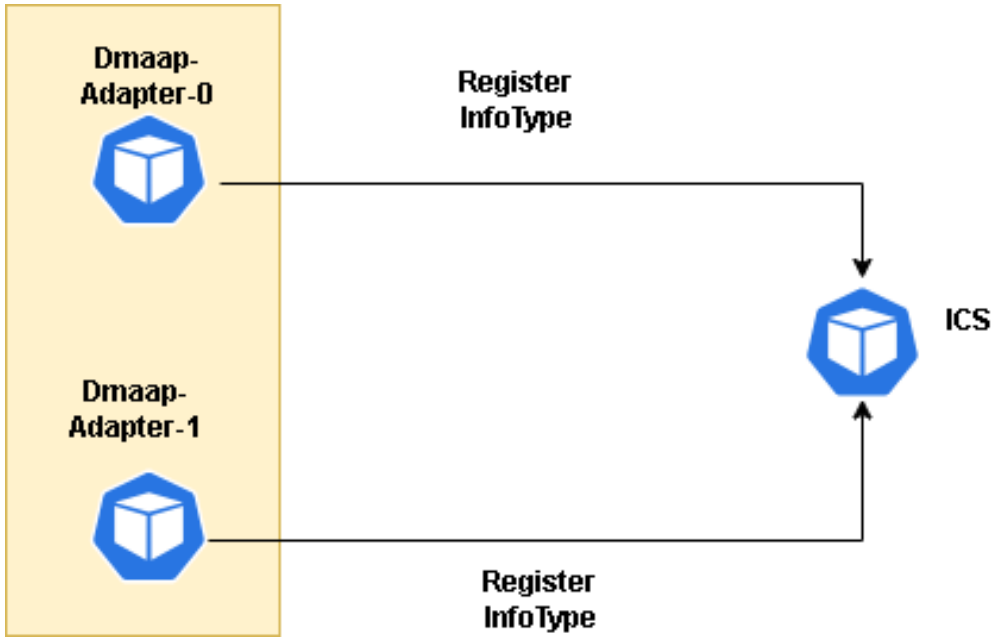
Dmaap-Adapter Scalability Solutions

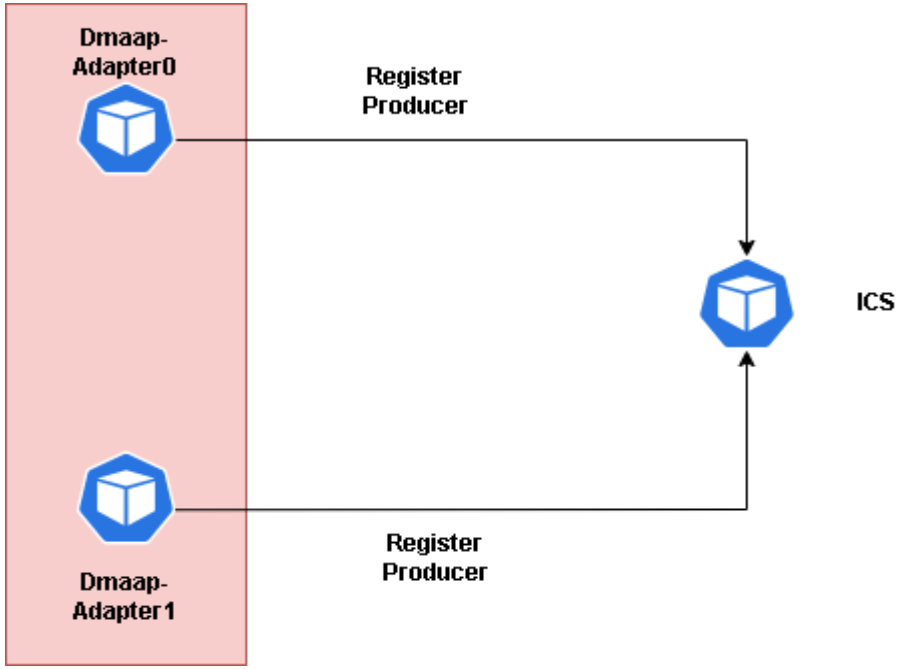
- Use Database to store job information.
- Sync Job Information between replicas (implementing DB trigger could be an option)

Queries on DmaapAdapter/PMProducer

- What is the purpose of jobgroup in code
- Why is Dmaap-Adapter having a Stateful-set chart when Dmaap-Adapter is stateless

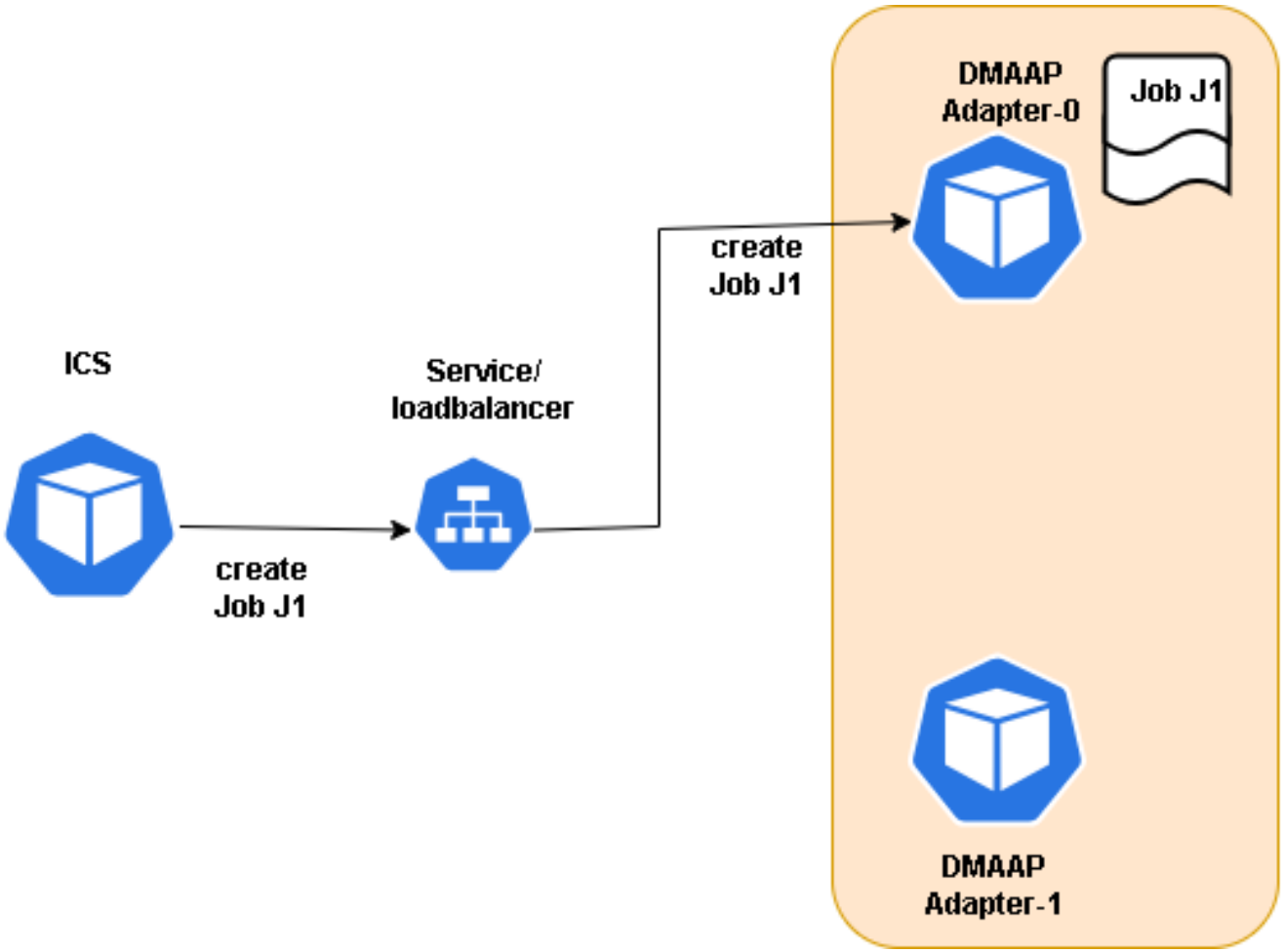
Scenario 1- Info-Type and Producer Registration with ICS





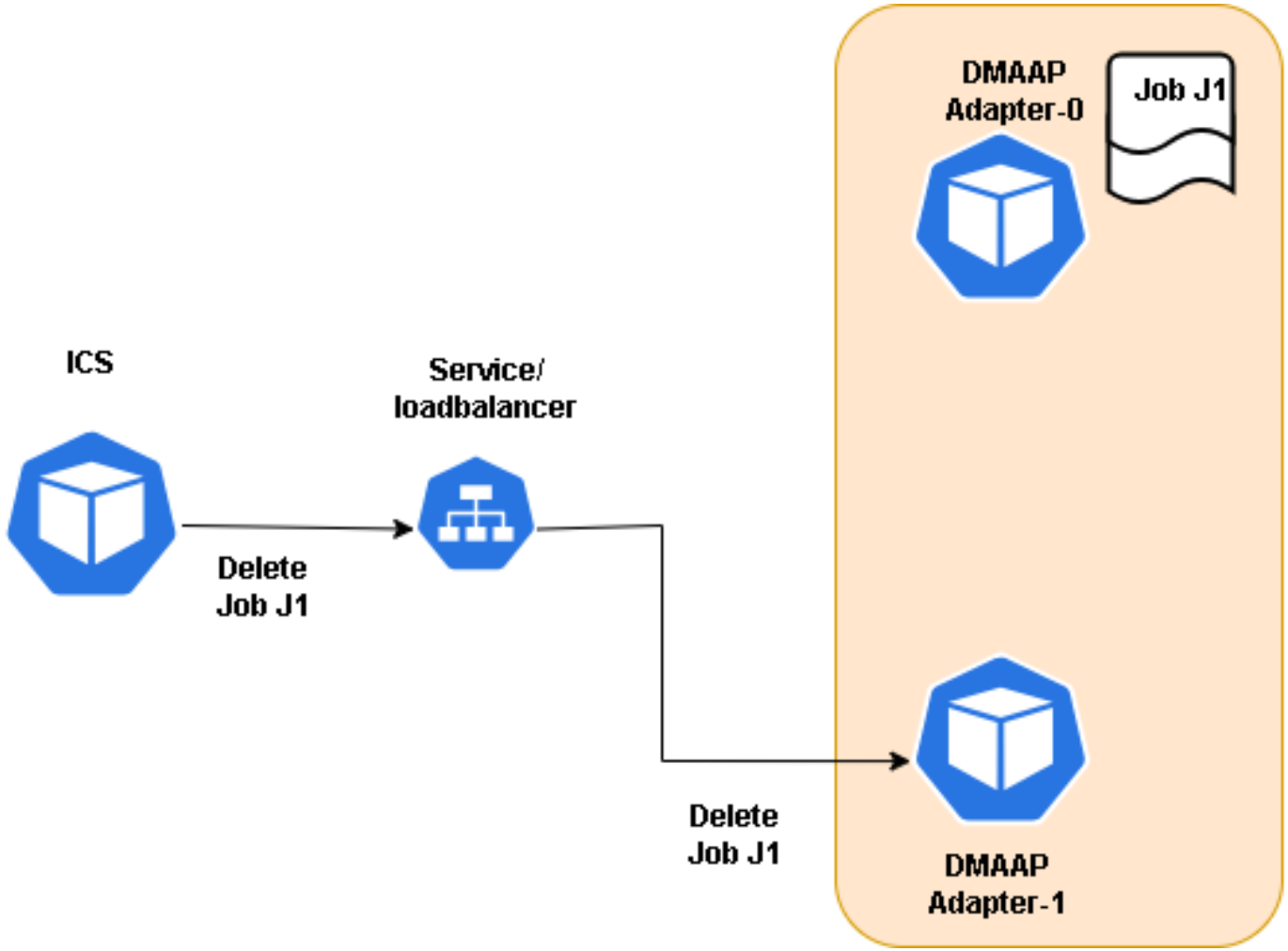
- When multiple replicas of DmaapAdapter are created they register with ICS with same jobCallbackUrl , producerSupervisionCallbackUrl.

Scenario-2: Data Inconsistency between Instances during Job Creation



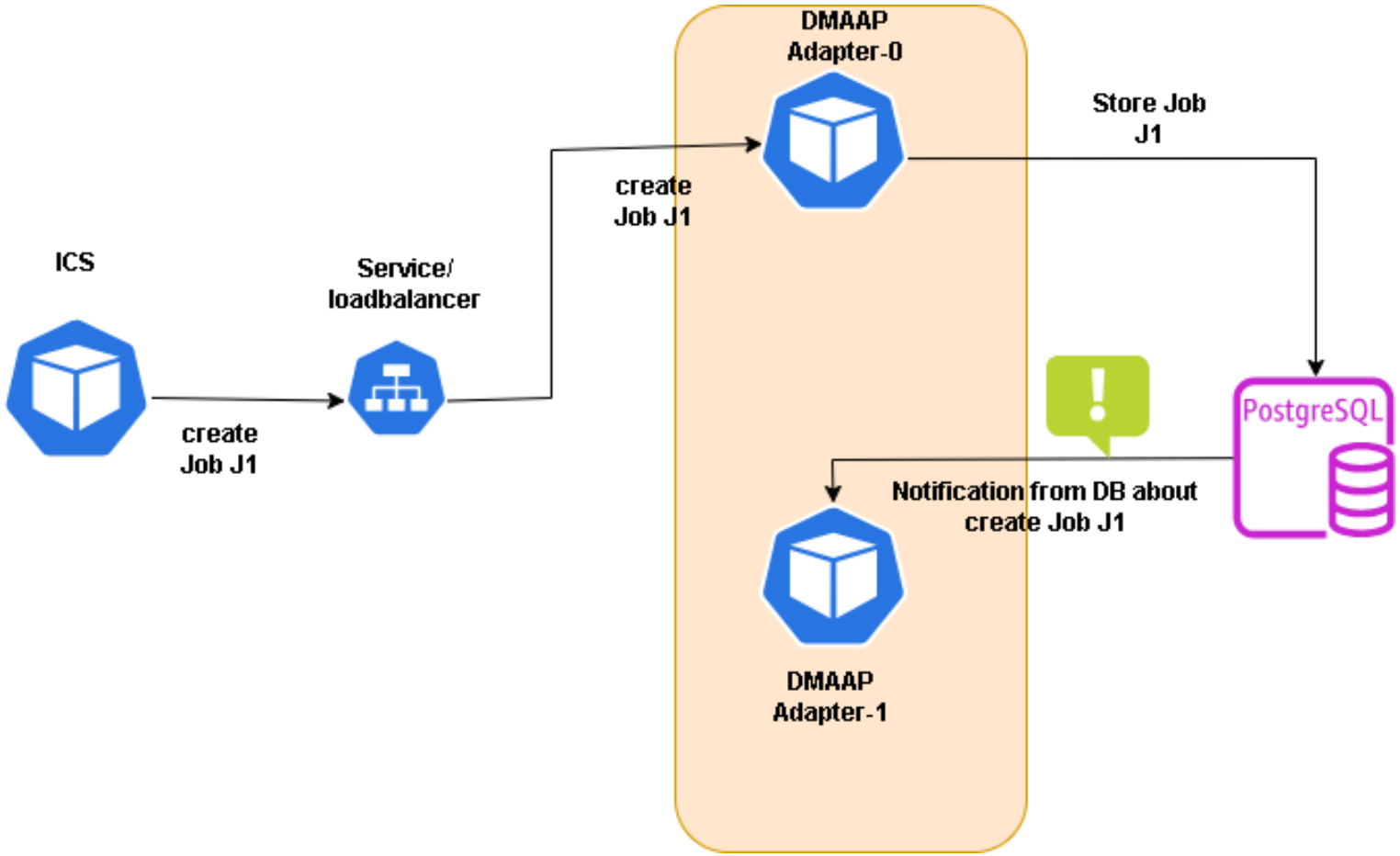
- ICS sends job creation messages to the Dmaap adapter, and the Kubernetes service, acting as a load balancer, distributes the calls to multiple replicas of the Dmaap adapter.
- Few jobs are created in Dmaap-adapter-0 and others in Dmaap-adapter1.

Scenario-3: Data Inconsistency between Instances during Job Deletion



- When remove job is called by ICS the message might be sent to a replica which is not having the job.

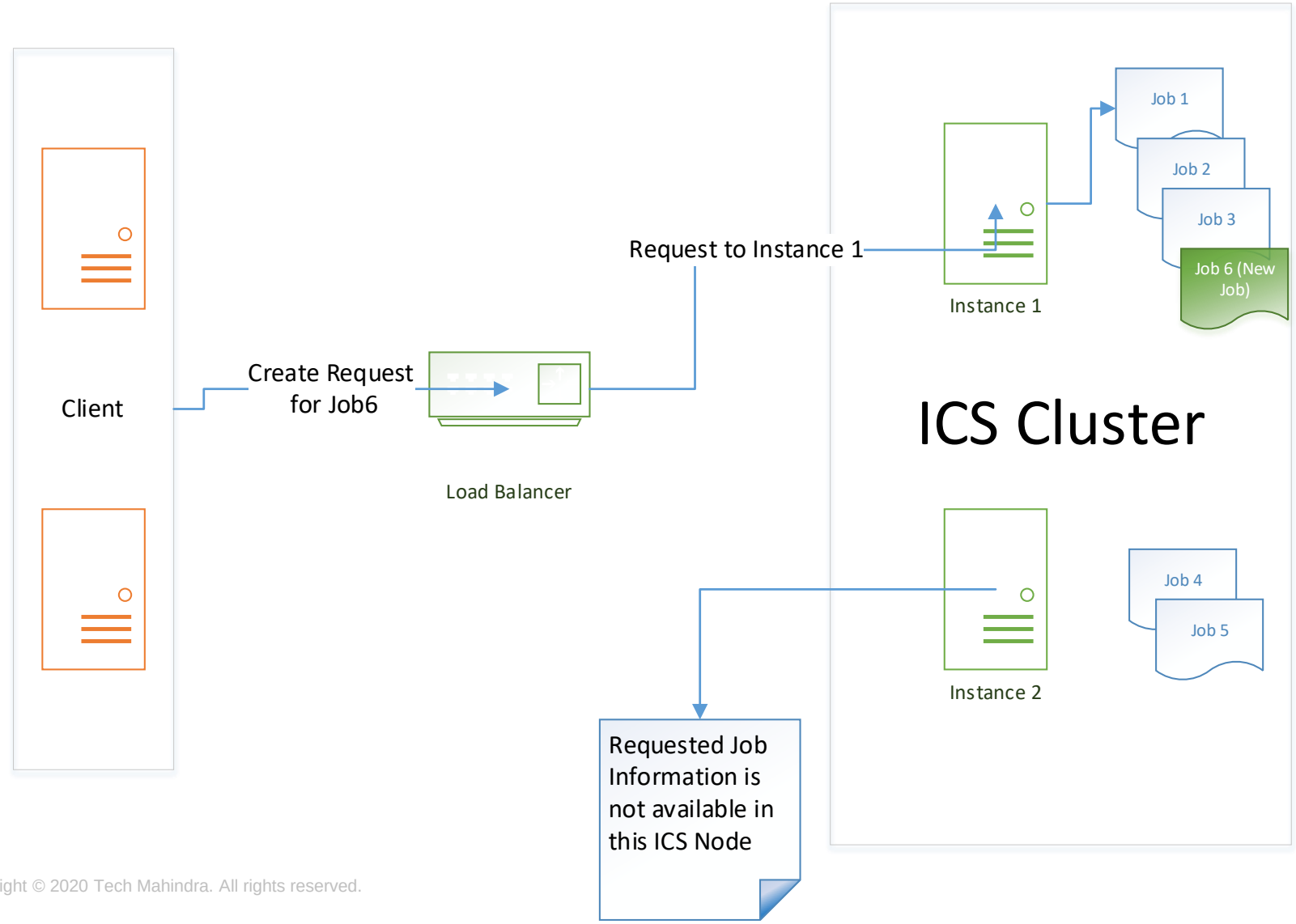
Solution



- Data storage in the form of DB can be implemented
- DB Trigger Notifications can be used to synchronize job information between the replicas

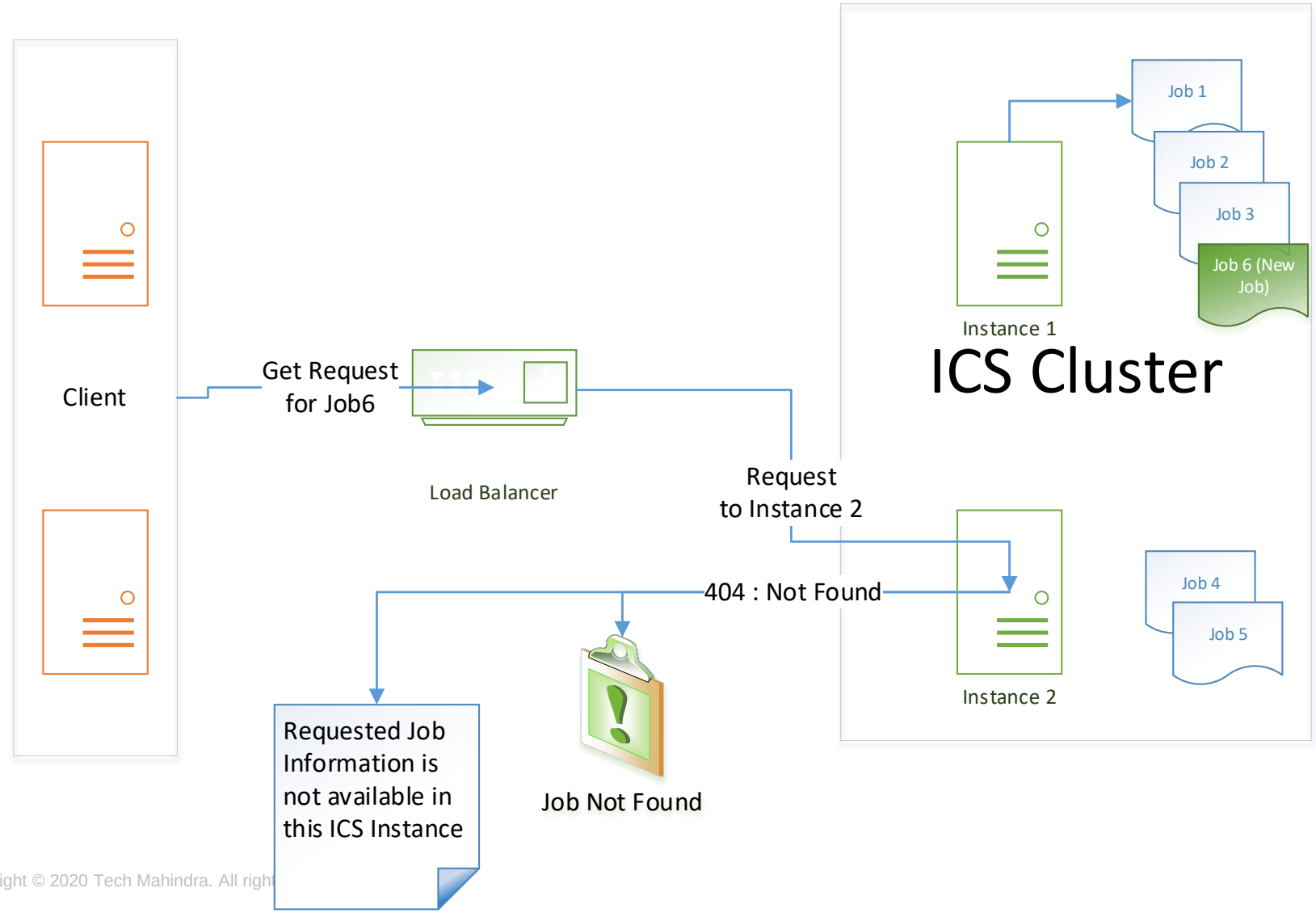
ICS Design

ICS Design – Job Creation (As Is)



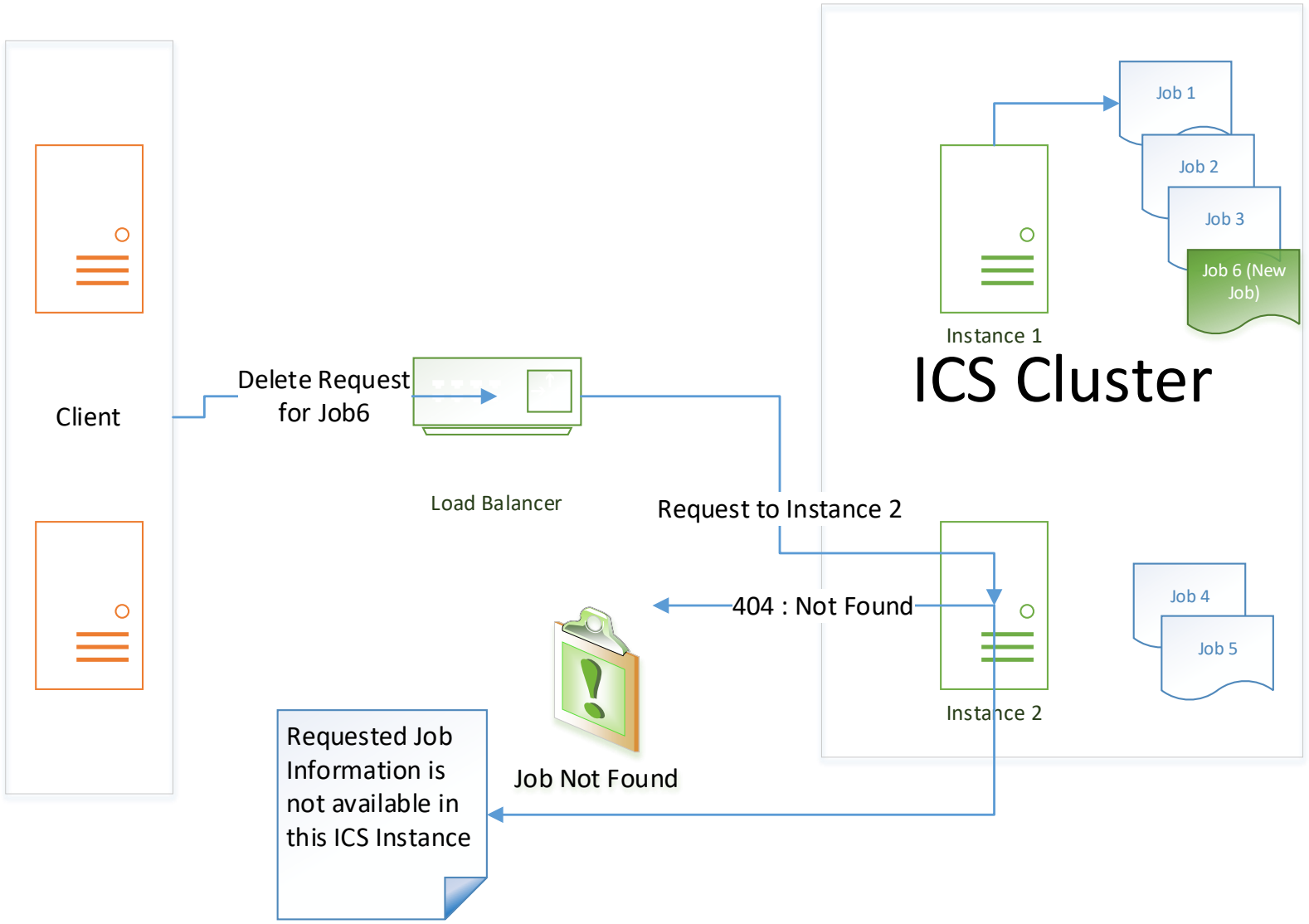
- Implementation of Job data is in File
- If the request from client comes to ICS Instance 1, then the Job information is stored only in the Instance 1 as the default implementation is based on FileSystem

ICS Design – Job Retrieval (As Is)



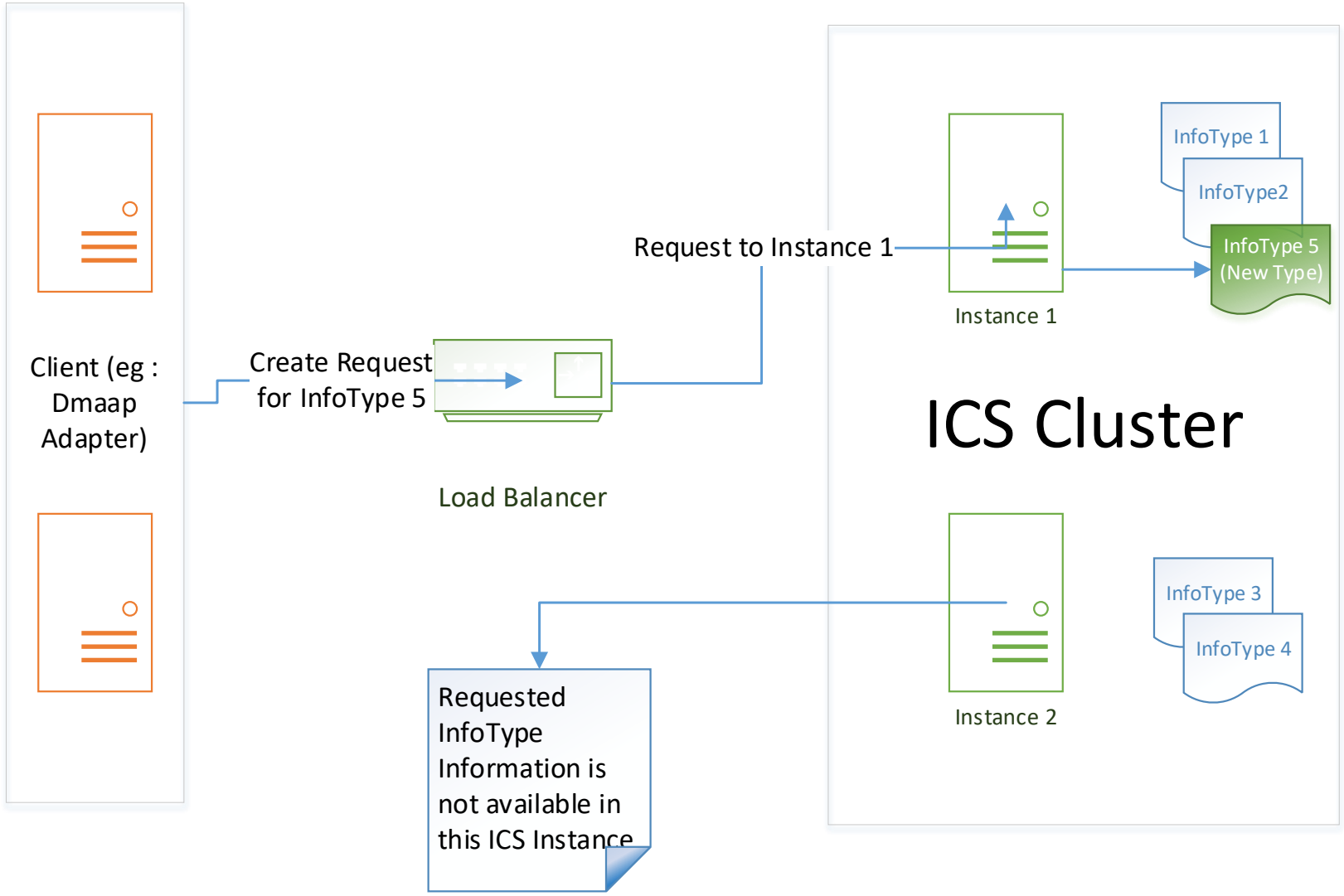
- Since the Job information is stored in Instance 1, and the request comes to Instance 2,, the Job information is not available
- ICS will respond with 404 : Job Not Found

ICS Design – Job Deletion (As Is)



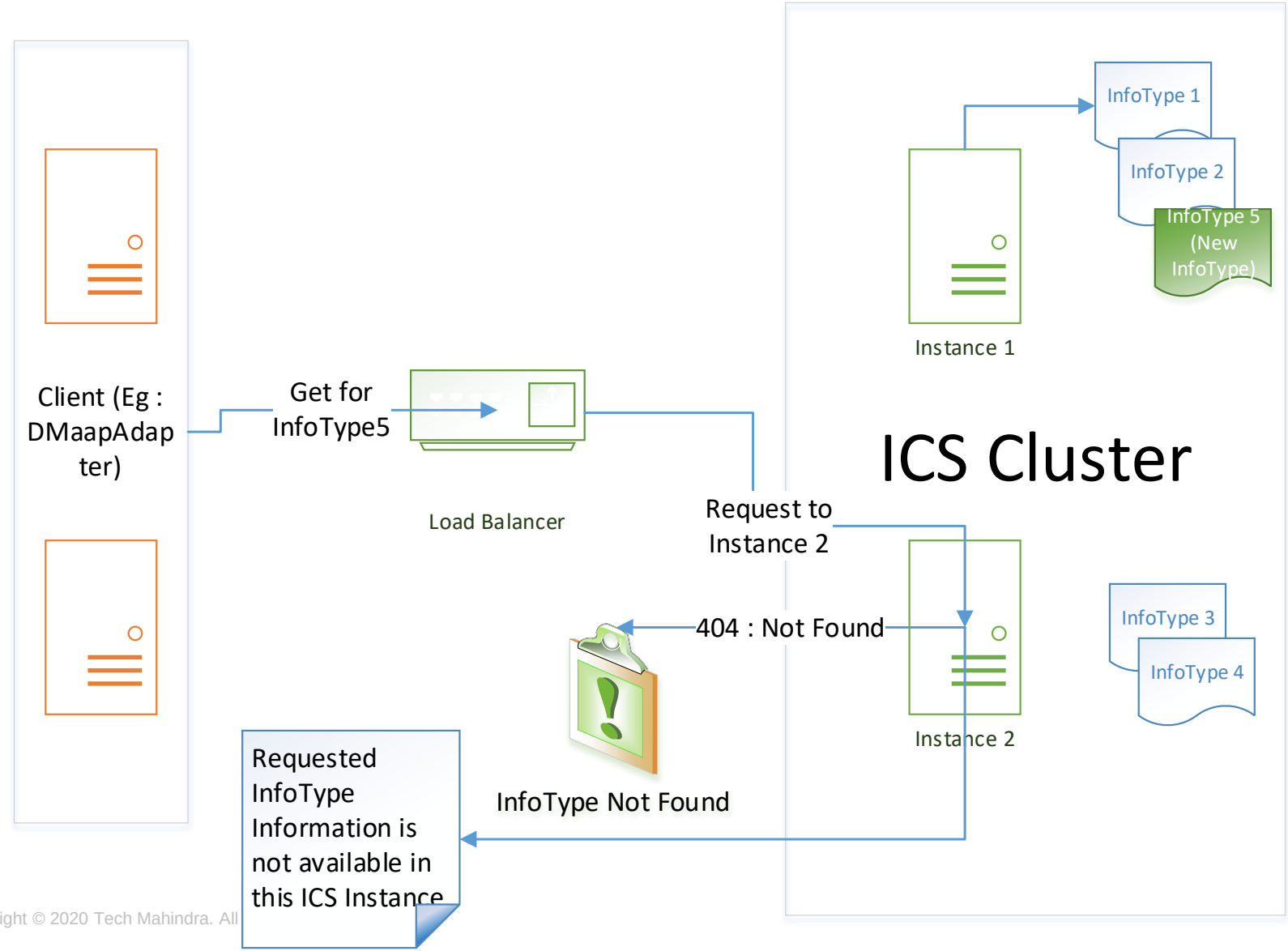
- Since the Job information is stored in Instance 1, and if the delete request comes to Instance 2, the Job information is not available
- ICS will respond with 404 : Job Not Found

ICS Design – InfoType Creation (As Is)



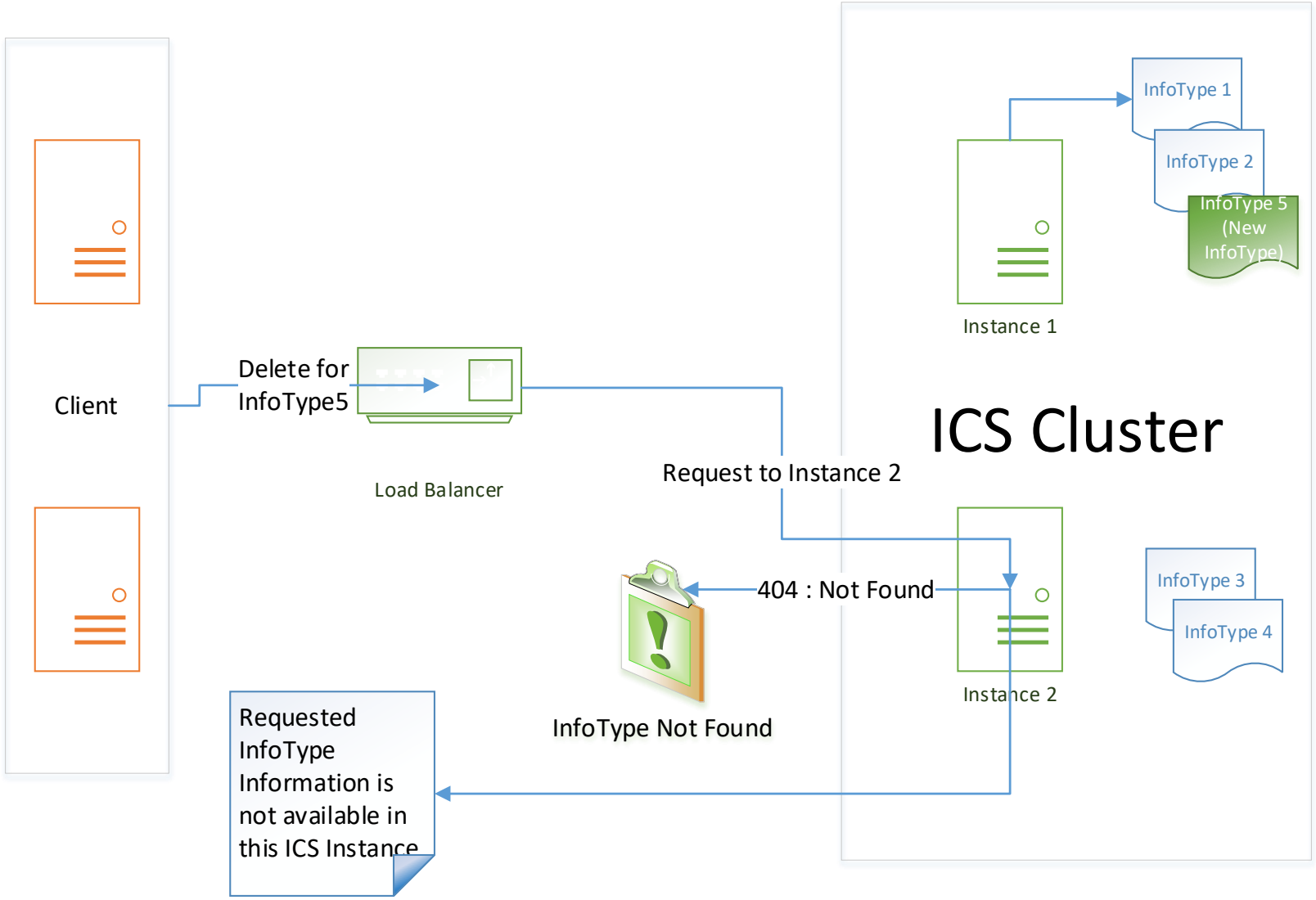
- Implementation of InfoType is in File
- If the request from client (like DMaapAdapter) comes to ICS Instance 1, then the InfoType information is stored only in the Instance 1 as the default implementation is based on FileSystem
- Here new InfoType5 is stored in Instance 1

ICS Design – InfoType Retrieval (As Is)



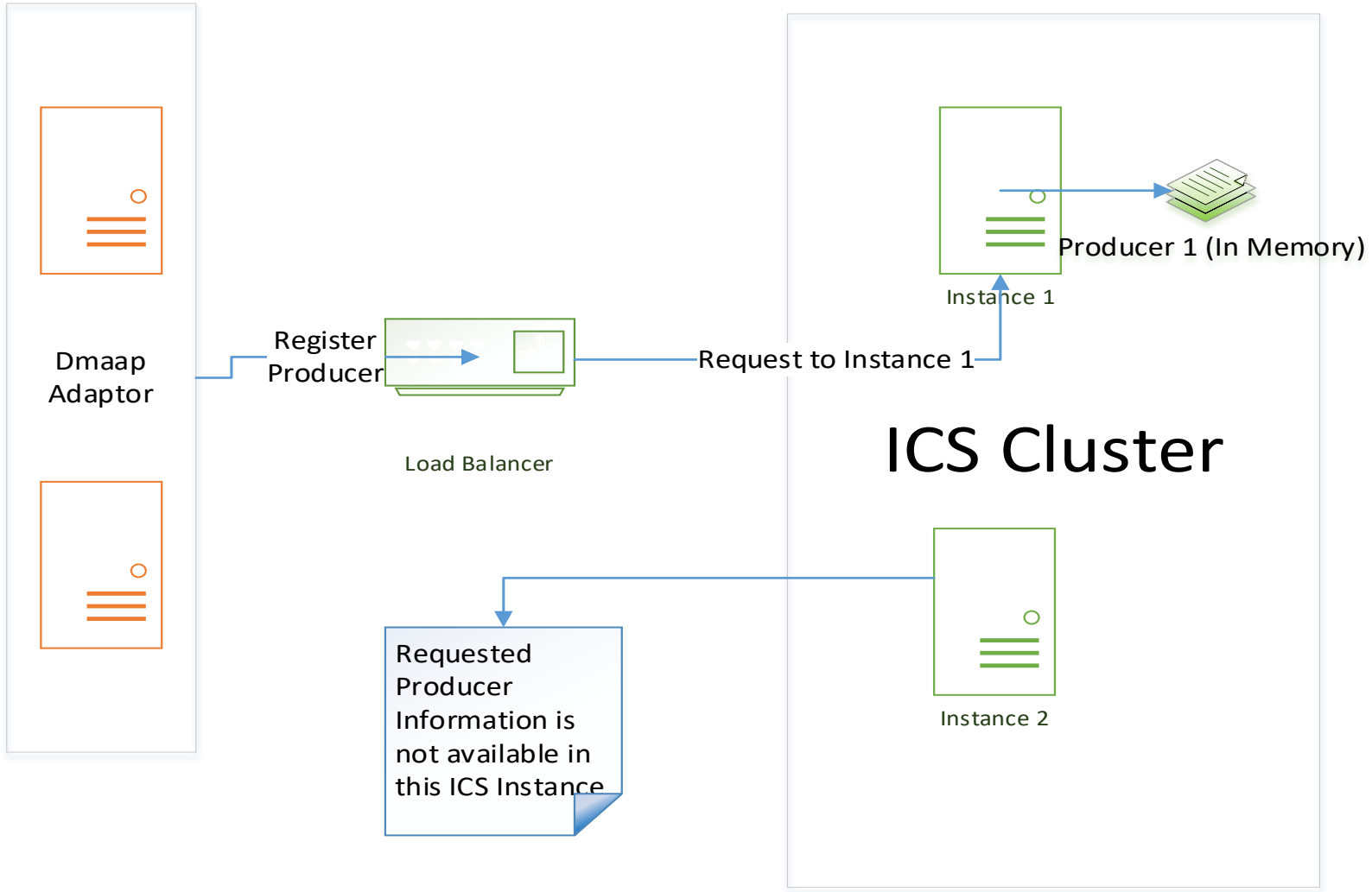
- Since the new InfoType information is stored in Instance 1, and the request comes to Instance 2,, the InfoType information is not available
- ICS will respond with 404 : InfoType Not Found

ICS Design – InfoType Delete (As Is)



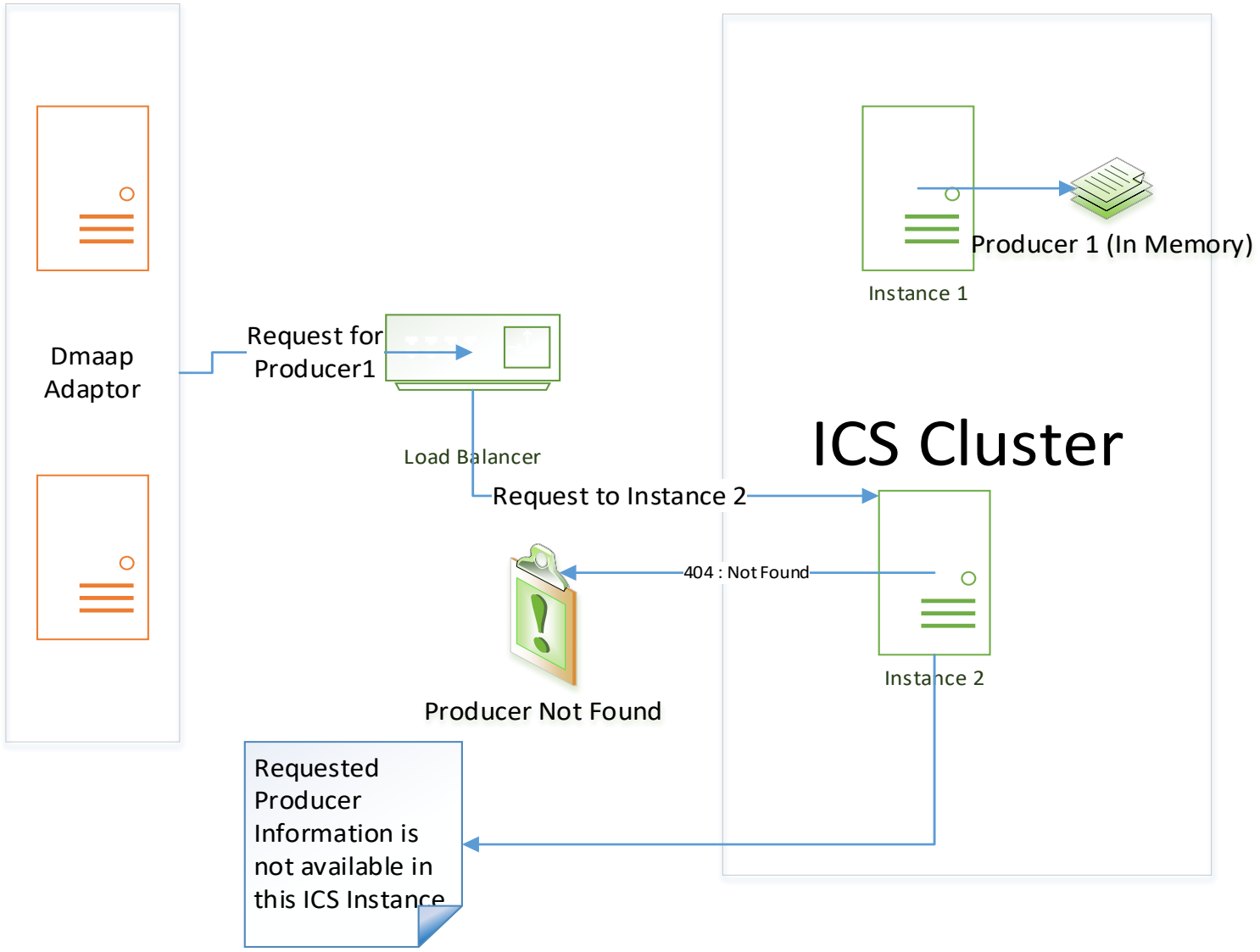
- Since the InfoType information is stored in Instance 1, and if the delete request comes to Instance 2, the InfoType information is not available
- ICS will respond with 404 : InfoType Not Found

ICS Design – Producer Registration (As Is)



- Implementation of Producer information is in maintained in memory
- If the request from client (say DmaapAdapter) comes to ICS Instance 1, then the Producer information is maintained only in the Instance 1 as it is maintained only in-memory

ICS Design – Producer Retrieval (As Is)

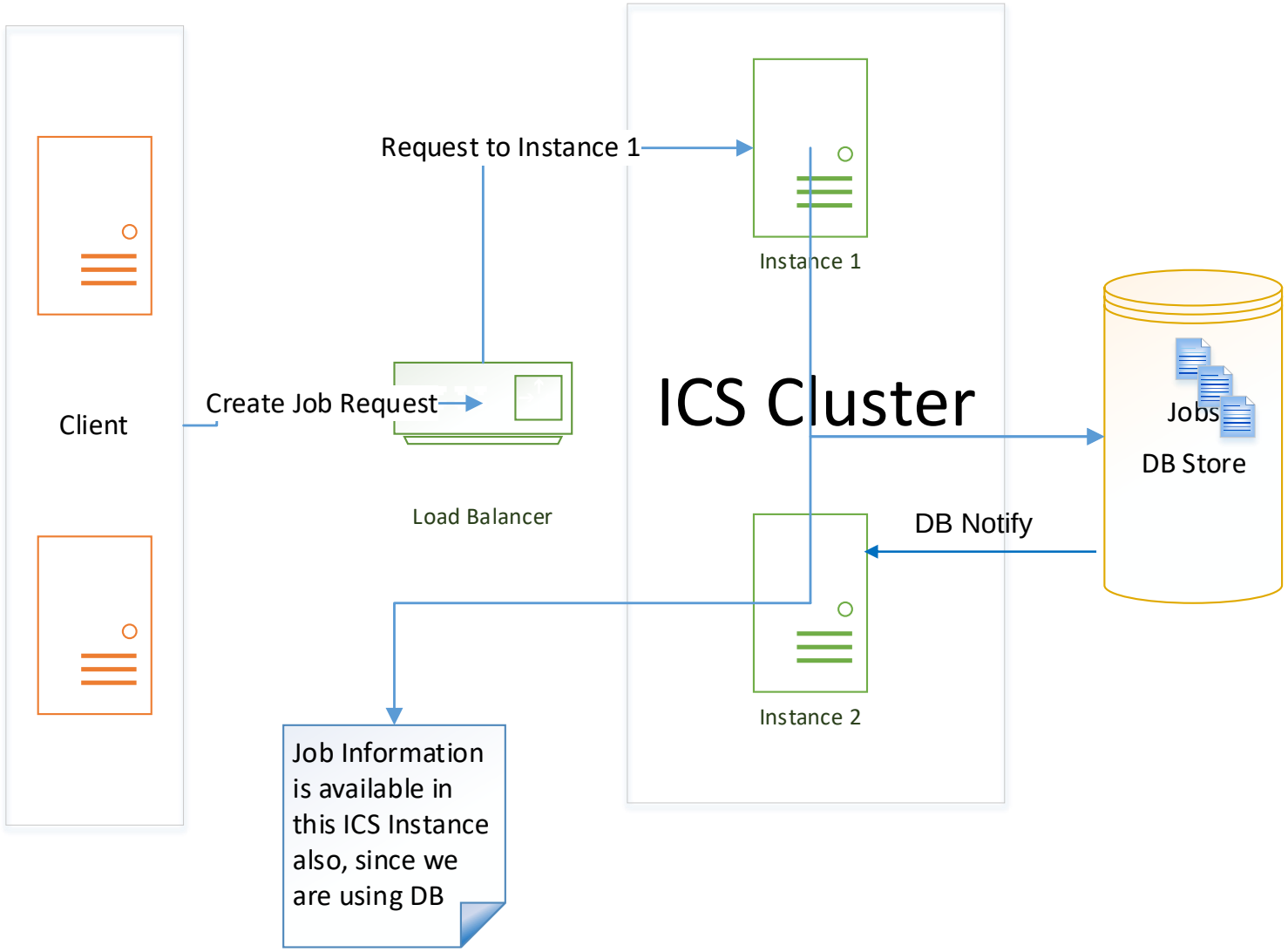


- If the request from client (say DmaapAdapter) for retrieval for Producer1 comes to ICS Instance 2, then it'll respond with 404 : Producer Not Found

Solution

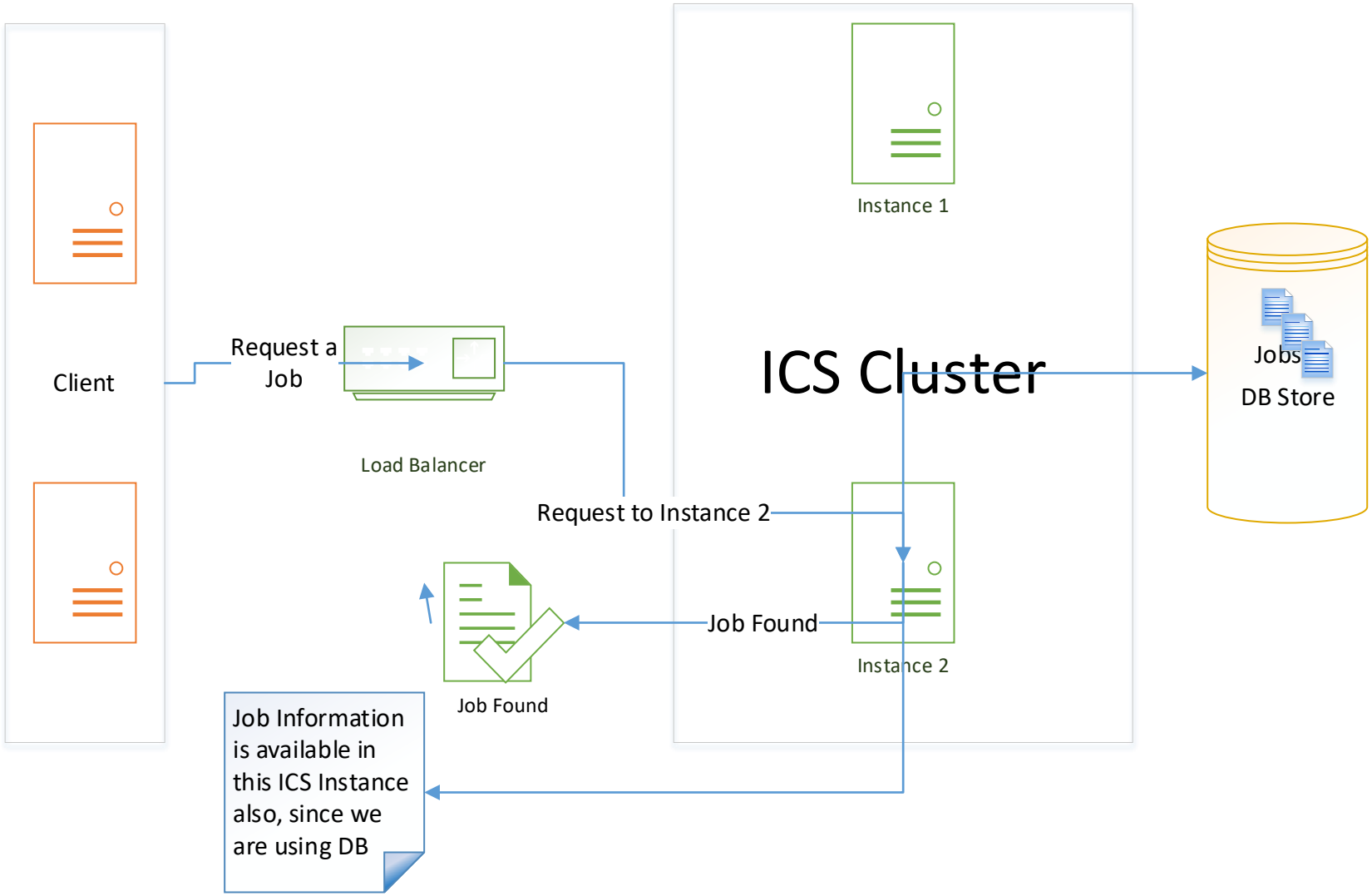
- Implement a shared DB to store Job information between ICS instances

ICS Design – Job Creation (To Be)



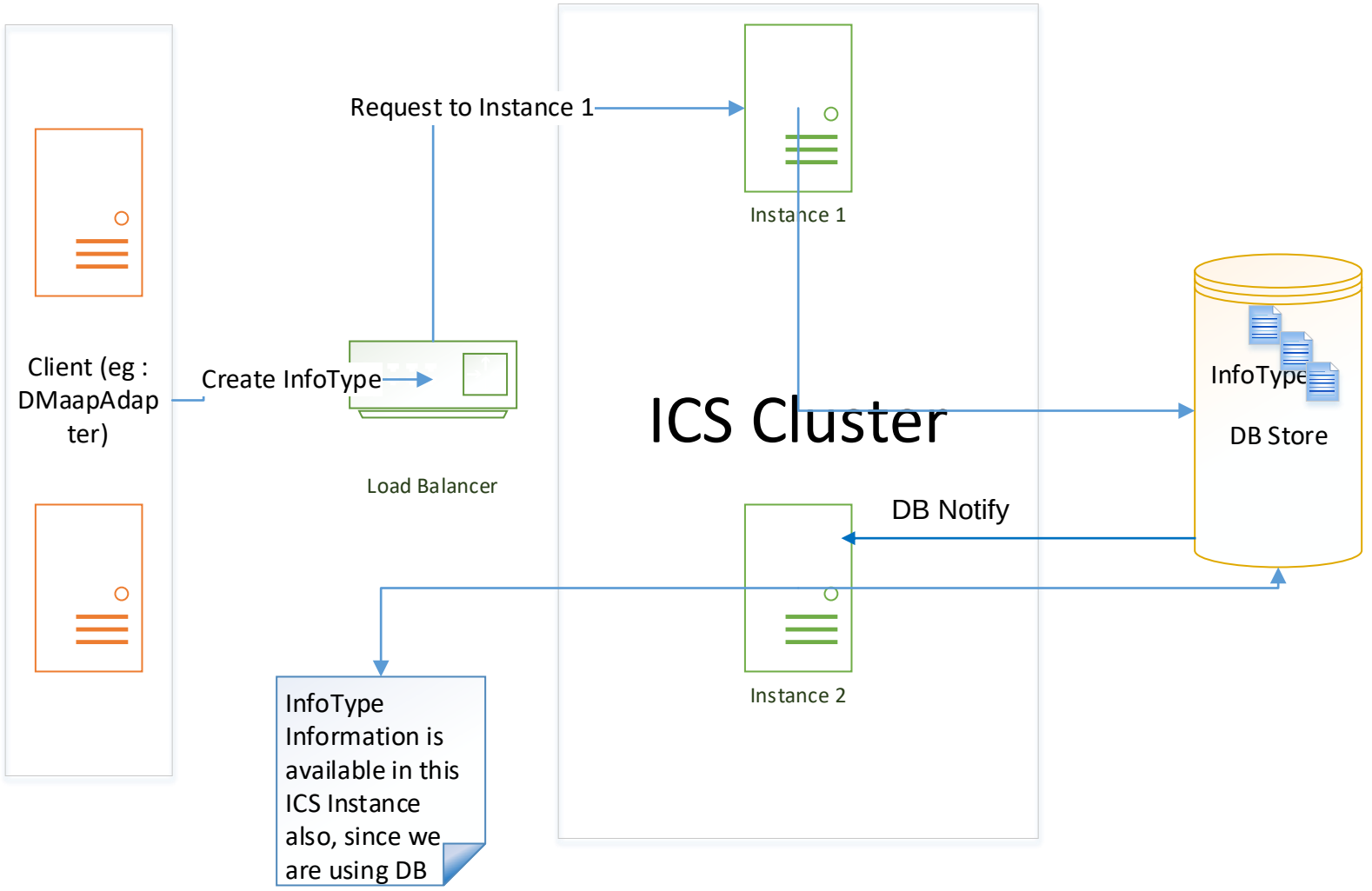
- Implementation of Job data is in Database Store
- If the creation request comes to Instance 1 the details are stored in Database.
- The job details are thus available to all the Instances in the cluster.
- Implement DB trigger to notify the other instances of ICS

ICS Design – Job Retrieval (To Be)



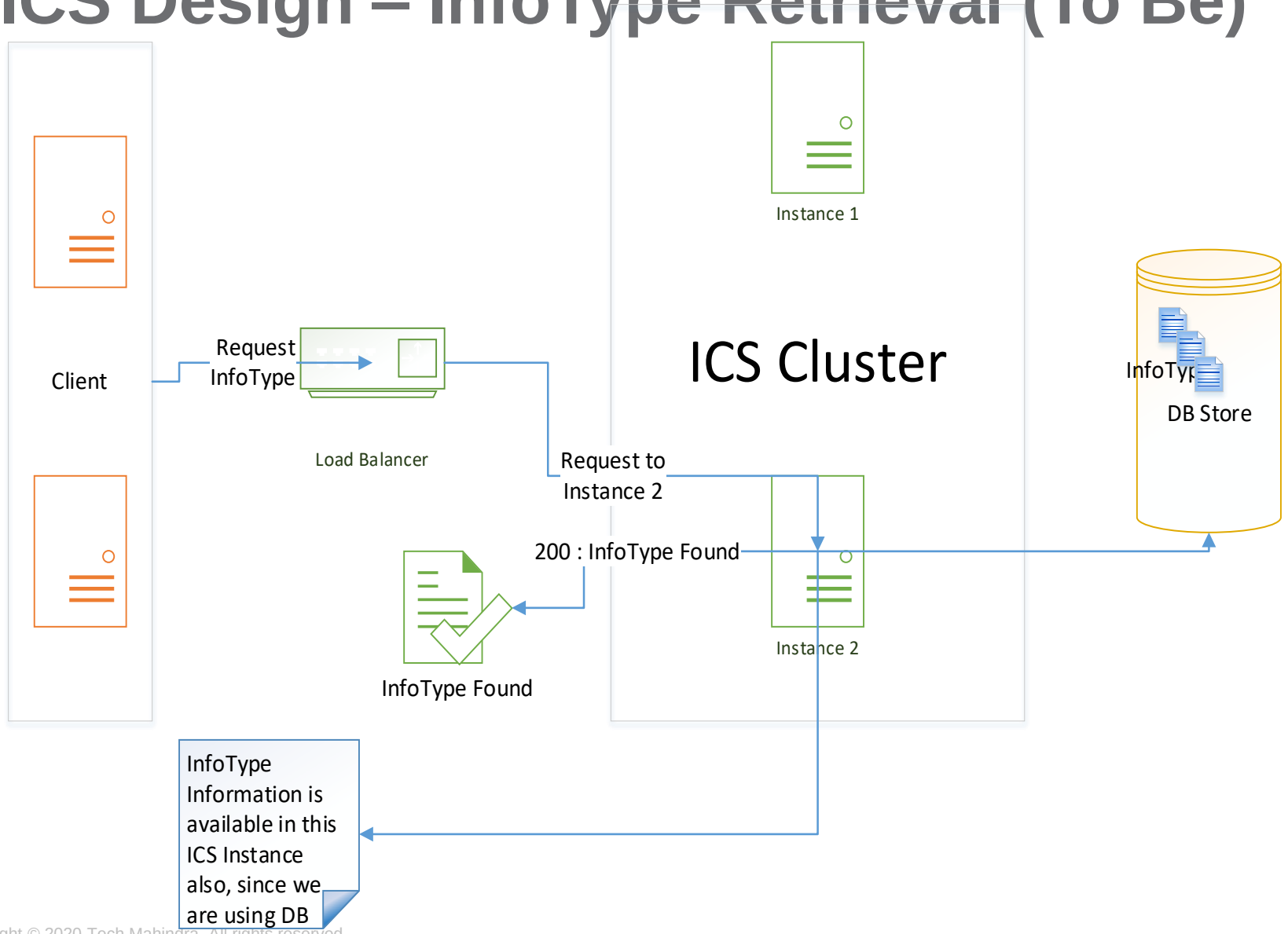
- If the get request for Job comes to Instance 2, since the details are available in Database, it can retrieve the information.
- Instance 2 responds with Job information

ICS Design – InfoType Creation (To Be)



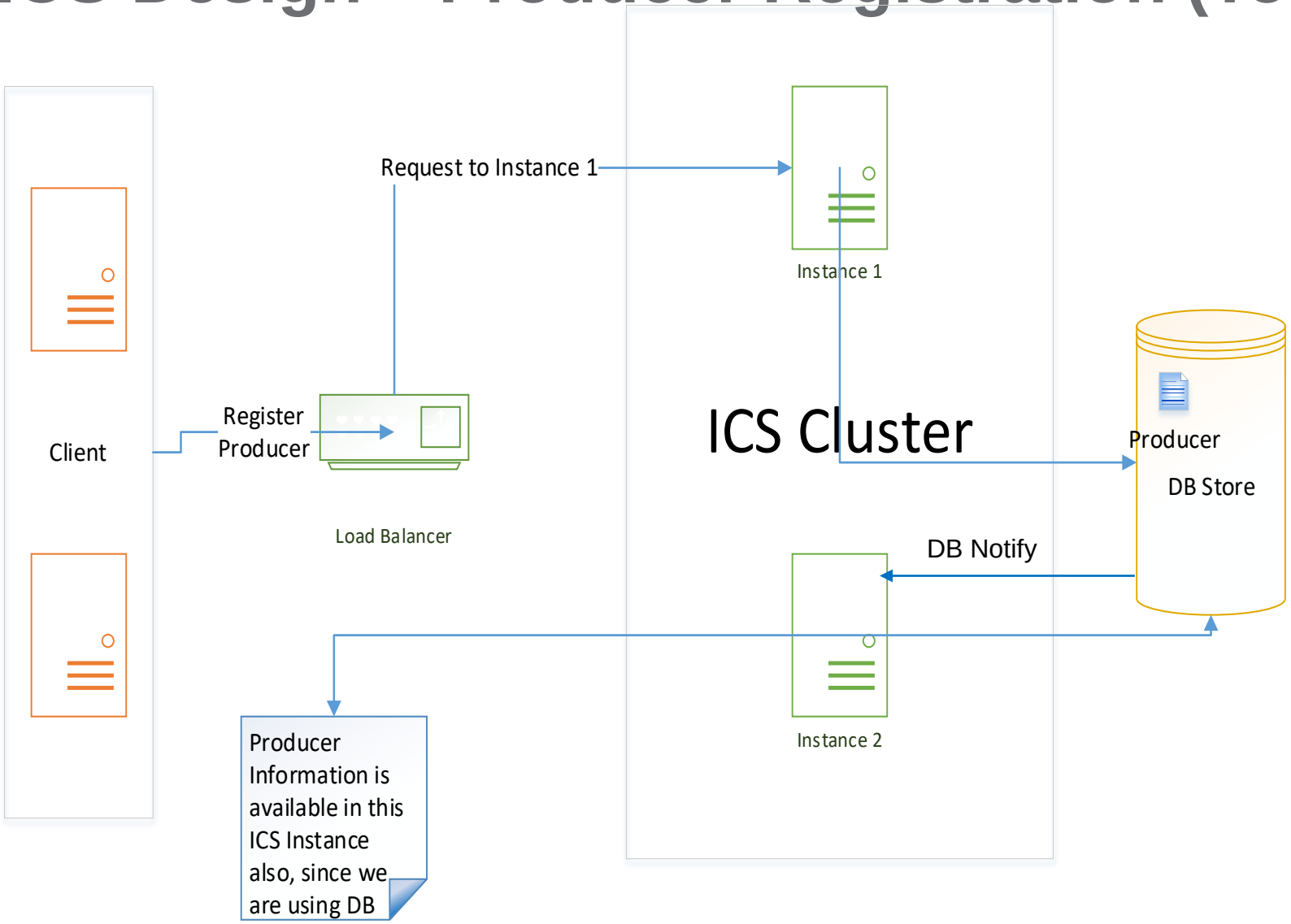
- Implementation of InfoType data is in Database Store
- If the InfoType creation request comes to Instance 1 the details are stored in Database.
- The InfoType details are thus available to all the Instances in the cluster.

ICS Design – InfoType Retrieval (To Be)



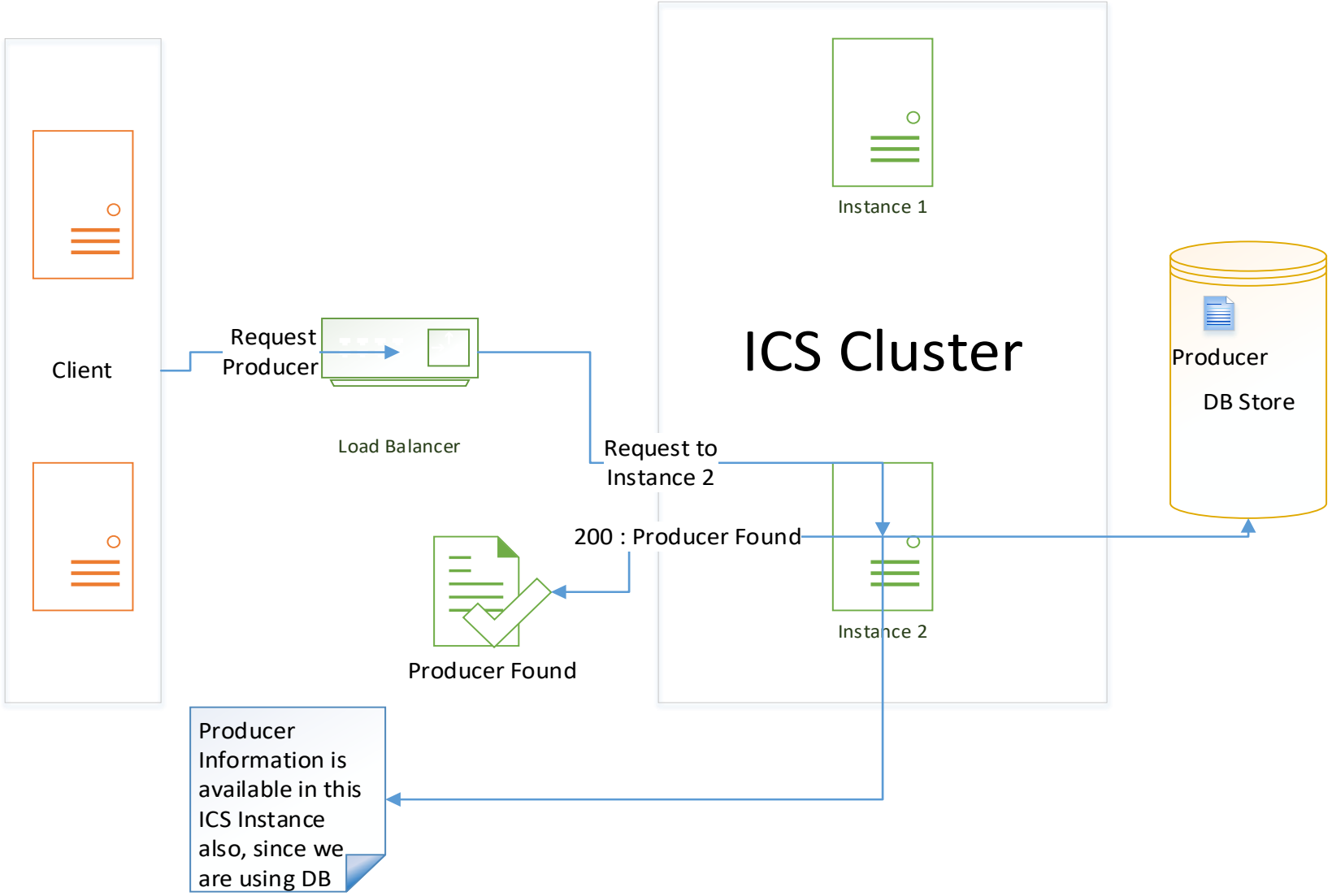
- If the get request for InfoType comes to Instance 2, since the details are available in Database, it can retrieve the information.
- Instance 2 responds with InfoType information

ICS Design – Producer Registration (To Be)



- Implementation of Producer data is in Database Store
- If the Producer registration request comes to Instance 1 the details are stored in Database.
- The Producer details are thus available to all the Instances in the cluster.

ICS Design – Producer Retrieval (To Be)



- If the get request for Producer comes to Instance 2, since the details are available in Database, it can retrieve the information.
- Instance 2 responds with Producer information

Thank You

Disclaimer

Tech Mahindra Limited, herein referred to as TechM provide a wide array of presentations and reports, with the contributions of various professionals. These presentations and reports are for information purposes and private circulation only and do not constitute an offer to buy or sell any services mentioned therein. They do not purport to be a complete description of the market conditions or developments referred to in the material. While utmost care has been taken in preparing the above, we claim no responsibility for their accuracy. We shall not be liable for any direct or indirect losses arising from the use thereof and the viewers are requested to use the information contained herein at their own risk. These presentations and reports should not be reproduced, re-circulated, published in any media, website or otherwise, in any form or manner, in part or as a whole, without the express consent in writing of TechM or its subsidiaries. Any unauthorized use, disclosure or public dissemination of information contained herein is prohibited. Individual situations and local practices and standards may vary, so viewers and others utilizing information contained within a presentation are free to adopt differing standards and approaches as they see fit. You may not repackage or sell the presentation. Products and names mentioned in materials or presentations are the property of their respective owners and the mention of them does not constitute an endorsement by TechM. Information contained in a presentation hosted or promoted by TechM is provided “as is” without warranty of any kind, either expressed or implied, including any warranty of merchantability or fitness for a particular purpose. TechM assumes no liability or responsibility for the contents of a presentation or the opinions expressed by the presenters. All expressions of opinion are subject to change without notice.