



Dream Big Data

2021

# Dream Big Data's migration to the cloud

OPENSIFT CASE STUDY

FABRE, OLDEN

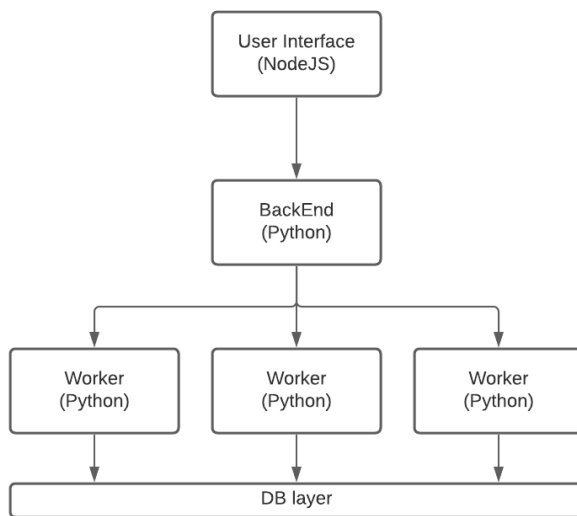
## Introduction

Dream Big Data, a fictive, but well-known figure in the world of data engineering, is looking at modernizing its framework and offer flexibility and power to its developer. They are currently developing mainly on premise, using standard size workstations and a large compute node that they bought to run heavy calculation.

However, their development cycle is long and costly, as they must manually deploy their final package to their test machine. Additionally, they know they will need some more computation resource soon.

With those requirements and background in shape, they are wondering how they can achieve a successful transformation to OpenShift to both reduce costs and increase performances.

For reference, here is a diagram of their current application:



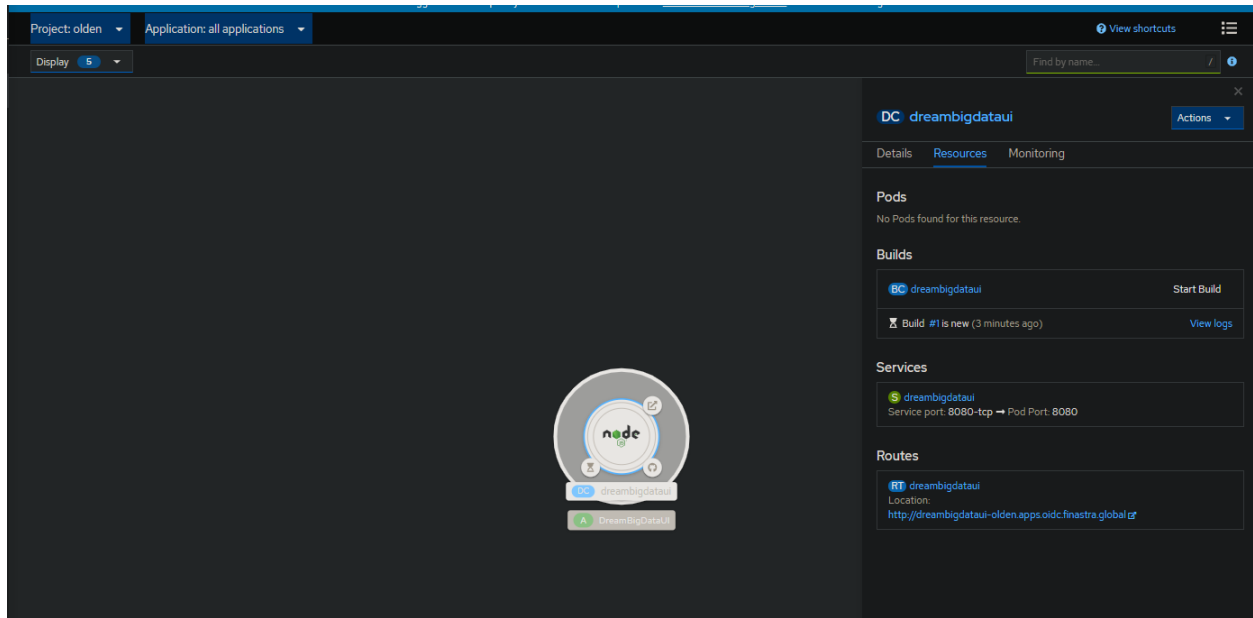
The number of workers is scalable so that it can handle more data. This is a good point regarding the path they want to take. However, the application is stateful, as it is relying on the database. This is also an aspect that they would like to improve. Moreover, all communication between the BackEnd and the workers happens through a data bus that also needs modernization. Finally, the concurrent access to the database will be a bottleneck in the future.

After discussion with a consultant at Red Hat, they come to the following plan:

## Switch the build to OpenShift

### Improving CI with OpenShift

A first step to increase the productivity of the team will be to create an automated process to build and deploy their application. For this, they will start with the UI part of their application:



With this image build in place, they ensure that their UI is always up to date with the updates they are issuing.

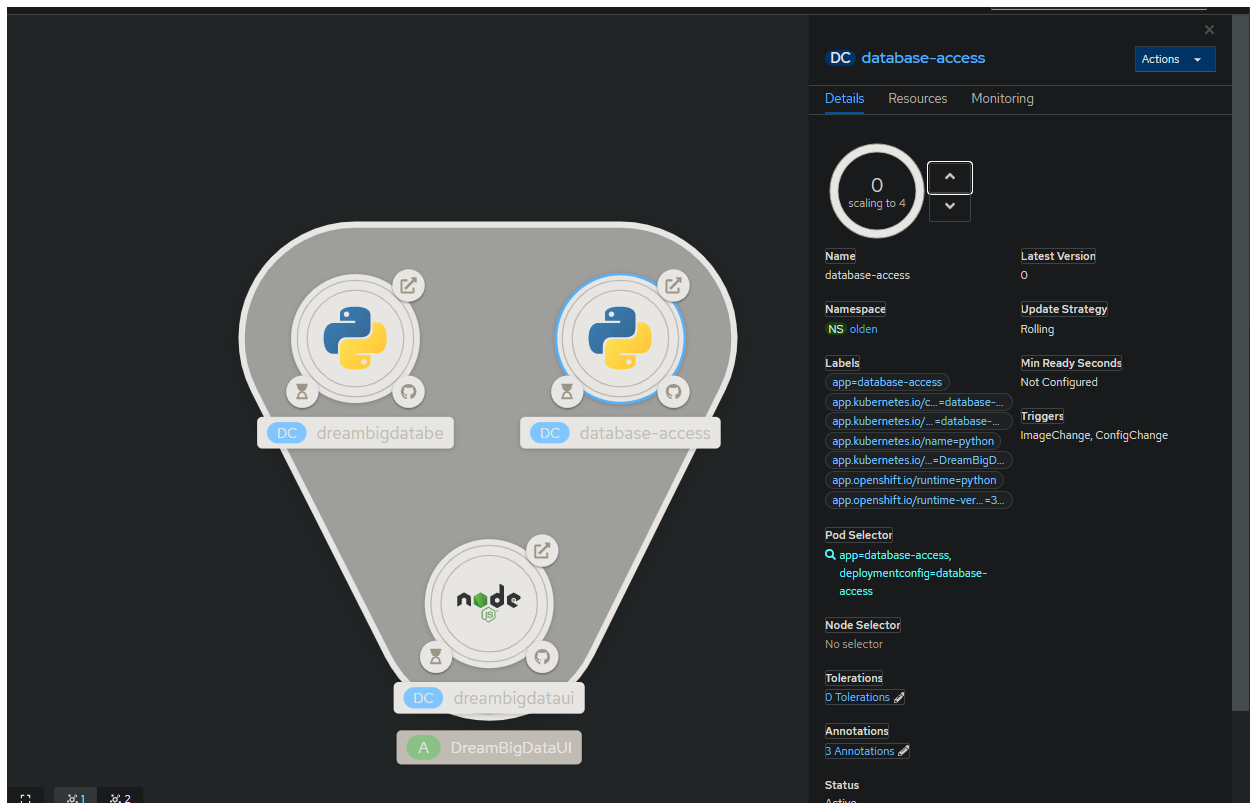
They plan on proceeding the same for their Backend. However, as the builds are not co-dependent, there is no need for a dependency between the two builds. They will simply update their image as soon as a difference appears on their source code.

As they are not planning on a full DEV / UAT / PROD model with multiple environments deployed, they are not interested in the promotion mechanism. The dev will be responsible in case his application is not smoothly integrated. In the future, an integration with ArgoCD could be in the picture if their dev team grows and the need for integration test appears.

### Using the UI for an easy embedding

As stated before, the dev team will use the full range of RedHat tool to create BuildStreams for their three components. This will only be a first step. Indeed, the company wants to reach a better maturity level where the developer would be able to deploy directly from its workstation rather than waiting for the build to occur on the OpenShift cluster. This will be addressed in a later chapter.

So far, they were able to generate a snapshot of their application, by modifying their internal configuration and linking the services together:



They were also able to scale their workers to achieve the same parametrization as before, on their big workstation.

However, this approach is not enough, as it does not address the biggest issue they have when running the application, which is the concurrent access to the database. Red Hat will go in more details on this in the next Chapter

## Continuous deployment

Thanks to this continuous build and test of their application, Dream Big Data was able to answer one of their need: Deliver faster and better. Indeed, as soon as an image has been validated, it can be exchanged with the client using a public registry such as quay.io or their private registry, that they use for client delivery.

## Split the monolith

### Separate the application further with REST API

Until now, the team is happy with the result that they were able to achieve by splitting their application in 3 distinct entities. However, the communication between the back end and the workers is still a bit rusty, with an old data bus model. To smooth the transition to a REST API model, Dream Big Data is going to use a standard build pack, delivered by Red Hat. By using this, they ensure that their process will be automatically compatible with OpenShift. They chose to build their application on top of Django, which can use the standard python build pack, with the import of the library. Even though Django still heavily relies on a database, the fact that it comes as a preconfigured pack will help the team leverage on OpenShift to monitor their builds.

You are logged in as a temporary administrative user. Update the [cluster OAuth config](#)

Project: olden


Application: all applications

## Create Source-to-Image Application

### Builder

Builder Image Version \*

IST 3.6

 **Python 3.6**  
BUILDER PYTHON

Build and run Python 3.6 applications on RHEL 7. For more information about using this builder image, including OpenShift considerations, see <https://github.com/sclorg/s2i-python-container/blob/master/3.6/README.md>.

Sample repository: <https://github.com/sclorg/django-ex.git>

### Git

Git Repo URL \*

[Try Sample ↗](#)

› Show Advanced Git Options

### General

Application

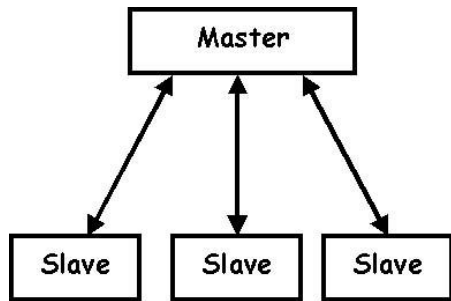
DreamBigDataUI

Select an application for your grouping or Unassigned to not use an application grouping.

Name \*

## Stateless application

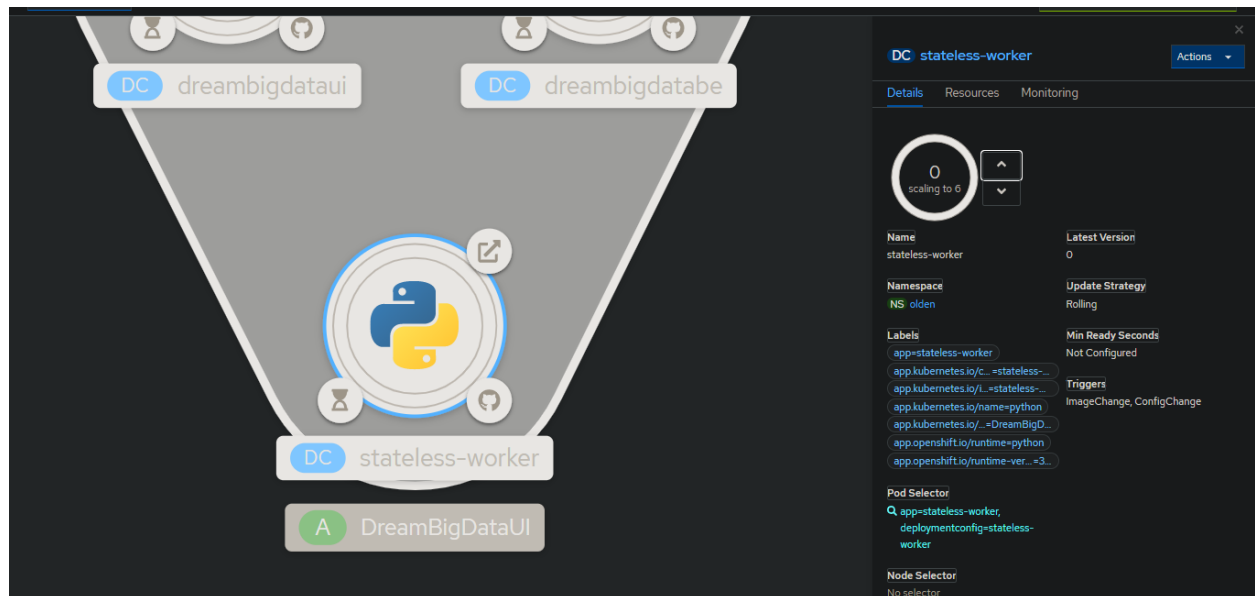
To achieve a full scalability, Dream Big Data also decided, in the future, to get rid of their database dependency, and enter a new Era of purely Stateless application. By leveraging on the previously stated Rest API, Dream Big Data would like to embed all results of the treatment directly in the body of the response and let the Back End application responsible for the split of the workload. It will also reconcile the output of the different workers. This step will occur in a second phases, after the switch to 3 services has been done. As a matter of fact, the second step contains a dependency to Django, which is a stateful database. For the second phase, we will get rid of this Database for workers and only the managing server will keep the link to the database to write and read inputs for the workers.



This will lead us to the next step that Dream Big Data would like to achieve.

## Concurrency

After the split has been done with the new stateless application, Dream Big Data will be able to scale their workers even more without fearing a bottleneck on the database access.

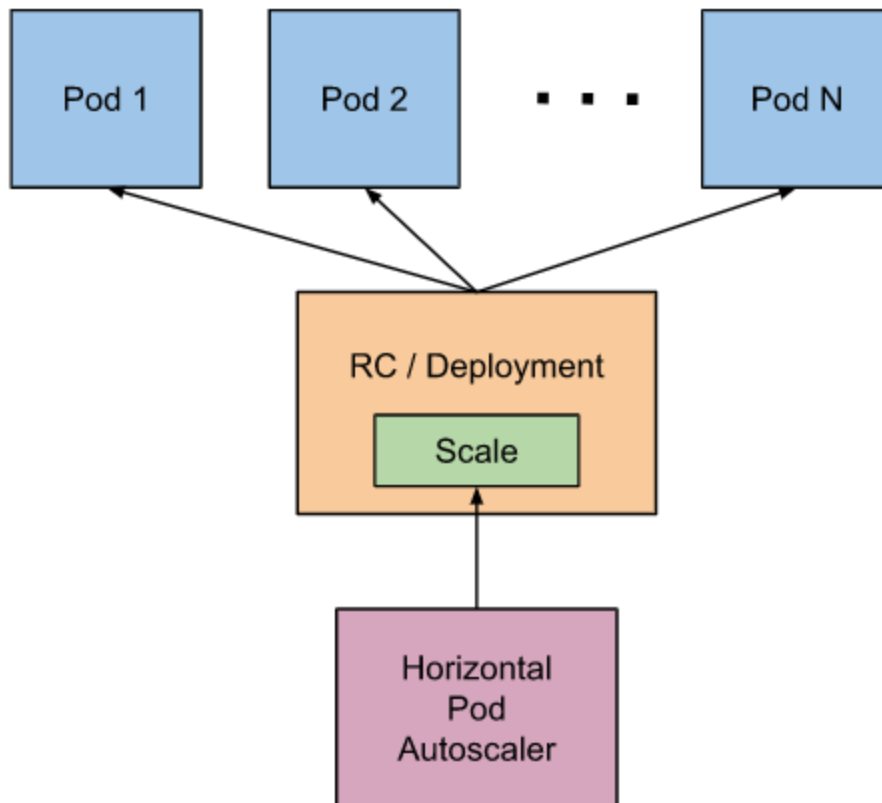


From now on, when a heavy calculation will occur, the team will be able to start multiple workers to help handle this workload more efficiently. However, this is not fully automated, and this is where Red Hat will push the design a little further.

## Scale

### Horizontal Pod Autoscaler

Thanks to all the work done on the modernization of their application, Dream Big Data can scale up and down their deployment according to the load they will receive. However, they would like to automate that scaling. That's where Red Hat will introduce the Horizontal Pod Autoscaler:



Now, by specifying simple rules, based on the workload, Dream Big Data will be able to automatically handle more workload without any external intervention. They can let their application run at night with a worker limit of 100 and the amount of resources they will use will adapt automatically to their needs. It means that now, their machine can be used for other tasks than the heavy computation during the day, and load more during the night, all thanks to the Horizontal Pod Autoscaler (HPA).

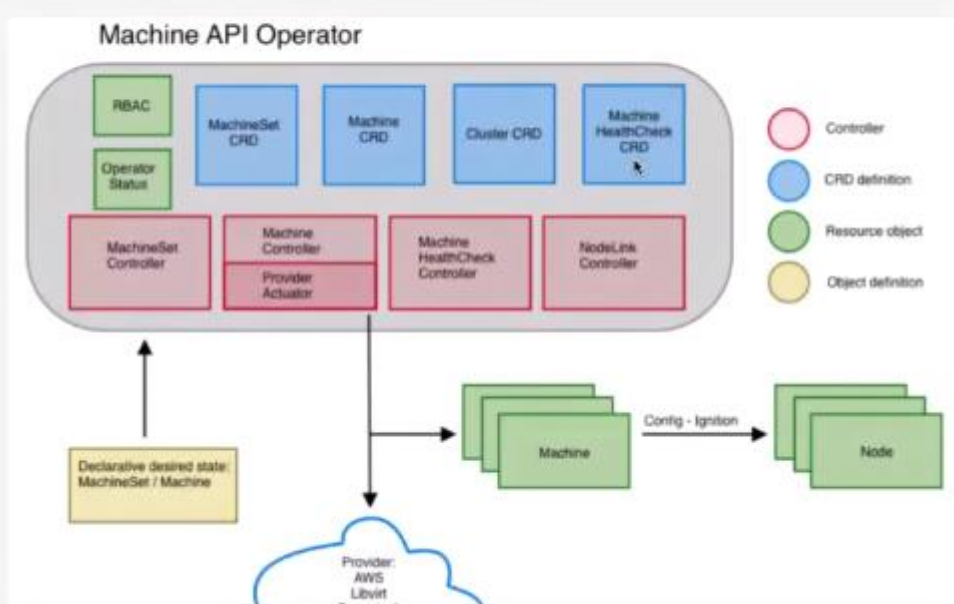
However, this leads us to a new question, what if Dream Big Data could get rid of their big, costly infrastructure altogether and switch to the cloud directly? By using one of the numerous cloud providers that OpenShift supports (AWS, AKS, GKE, etc.), the company could reduce its cost and even leverage on new functionality, directly embedded in OpenShift.

Thanks to the work at the beginning of this exercise, we can feel the real advantage of switching to a serverless paradigm. What if, for instance, we could startup new worker nodes (or machines) automatically?

### MachineSet auto scaling

Until now, the workload was running on a big machine where OpenShift was installed. Of course, we don't gain much as the infra is still there and the cost has been paid upfront. However, this changes when switching to an OpenShift installed in the cloud. By leveraging on MachineSet and MachineConfig, all embedded in OpenShift, Dream Big Data will be able to exponentially grow their computation power and not be bothered by heavy calculations.

# Machine Management Overview

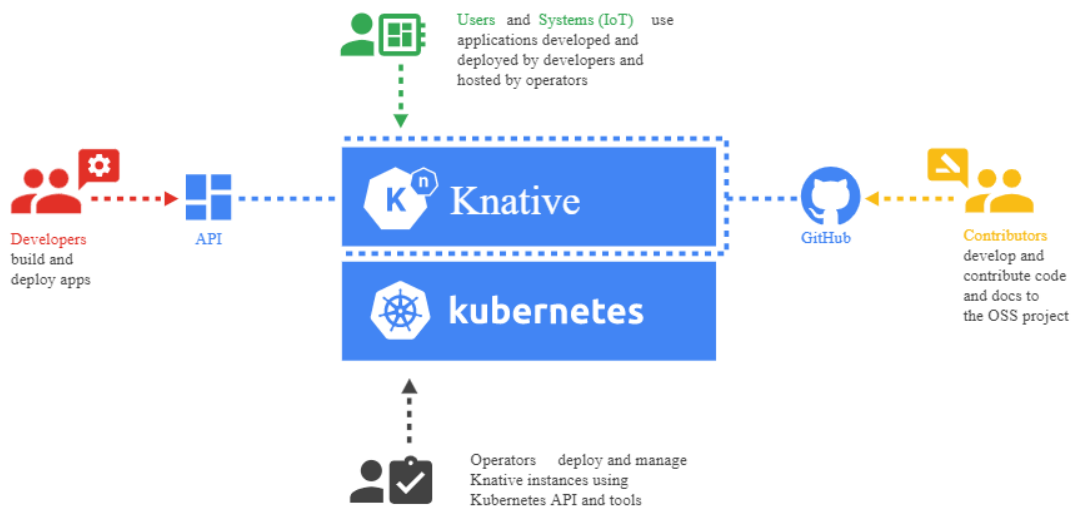


When the application is at rest, only one worker node can answer the basic test requests of the developer. However, when the real load test starts, the cluster will scale the pods accordingly. If it exceeds the amount of computation power of the only worker node, others will start and add computation power to the cluster.

## [Use Knative to speed up the development loop](#)

This implementation has been nice so far, but one missing part of the puzzle is the developer inner development loop. As we said at the beginning, the developer would like to leverage on the power of the cluster to develop his application. Therefore, Red Hat would advise Dream Big Data to look into knative as a final step. Knative will allow the developer to have a developer experience, while using the power of the infrastructure that is given to him. The deployment of new services is easy and will allow for an extension of the application in the future using new, future-proof, micro-services.





The whole development experience will be joyful and more effective. And we also reach the ultimate step of automation where a service, after a time of inactivity, can be reduced to 0, allowing all costs to be reduced to 0 as far as infrastructure goes. The machine can be entirely dedicated to other tasks. Or in the case of the cloud, the usage of resources is down to 0.

## Notes

Other technologies could enter the picture a bit later. For instance, if more services would be required to enrich the functionalities of Dream Big Data software, then introducing a new pipeline architecture with Tekton, for instance, could help ease the process of building the application and maintaining up to date images. The use of ArgoCD to handle the full deployment of the application could also be encouraged for the future, but the product is at an early stage on the automation path, making the use of such tool not relevant for the moment.

## Sum Up

By following all the described steps in this document, Dream Big Data has been embark on a journey to put in place a full CI/CD process, based on Kubernetes, an open source technology, while ensuring the stability of their system by relying on Red Hat to manage their infrastructure.

By leveraging on powerful mechanism, such as automated build and deployment, Dream Big Data was able to automatize their process completely and make full use of continuous integration.

With all the advice from Red Hat, Dream Big Data has already made a leap towards their cloud native journey, and the tools presented to them should allow them to go beyond their expectations, all thanks to the power of OpenShift.

*"Dream Big (Data) or Go Home"*