



Case Study



Dynamic and cost optimized deployments of multiple products from a centralized platform

A product company provides quick access to multiple products from their marquee platform using an operator skeleton framework

Executive Summary

Our customer has developed multiple purpose built products. The end users need many of these products for different kinds of analysis. Working with GS Lab, the customer made all of their products accessible from their marquee platform reducing efforts and costs for the end customer.

Overview

Our customer is a market leader in enterprise big data management and exploratory analytics. They help build large enterprise-wide data lakes which allow customers to consolidate and build data storage & retrieval facilities. Companies analyze this data using different types of processes to generate meaningful insights. Our customer has built multiple special purpose products that leverage such data sets. Every product built by our customer needed its own deployment.

With deployments comes the hassles of infrastructure planning, costs, optimization, security and monitoring. In some cases, the end customer would even choose a competitive product. Our customer had a vision of making it all very simple and convenient for the end customer. This would empower customers to unearth powerful insights.

Challenge

Our customer decided to make all their products available from their marquee platform UI. Their idea was to deploy the product on-the-fly, run the product on a large database and then close the instance. Although this solution would bring cost savings and convenience to the end customer, it posed its own challenges:

1. Users deploying a product needed to get complete visibility of the status of the deployment. Most commonly used deployment methods did not provide this much-needed visibility.
2. Companies needed complete control over their deployment configuration.
3. The deployment process has to be modular and flexible. Companies needed the ability to couple it with GUI or run it independently.

4. Capital expenditure needed to be minimized.
5. After using the product instance, it had to be torn down completely to avoid costly surprises.
6. The solution needed to be as cloud agnostic as possible.

Solution

Exploration and decisions

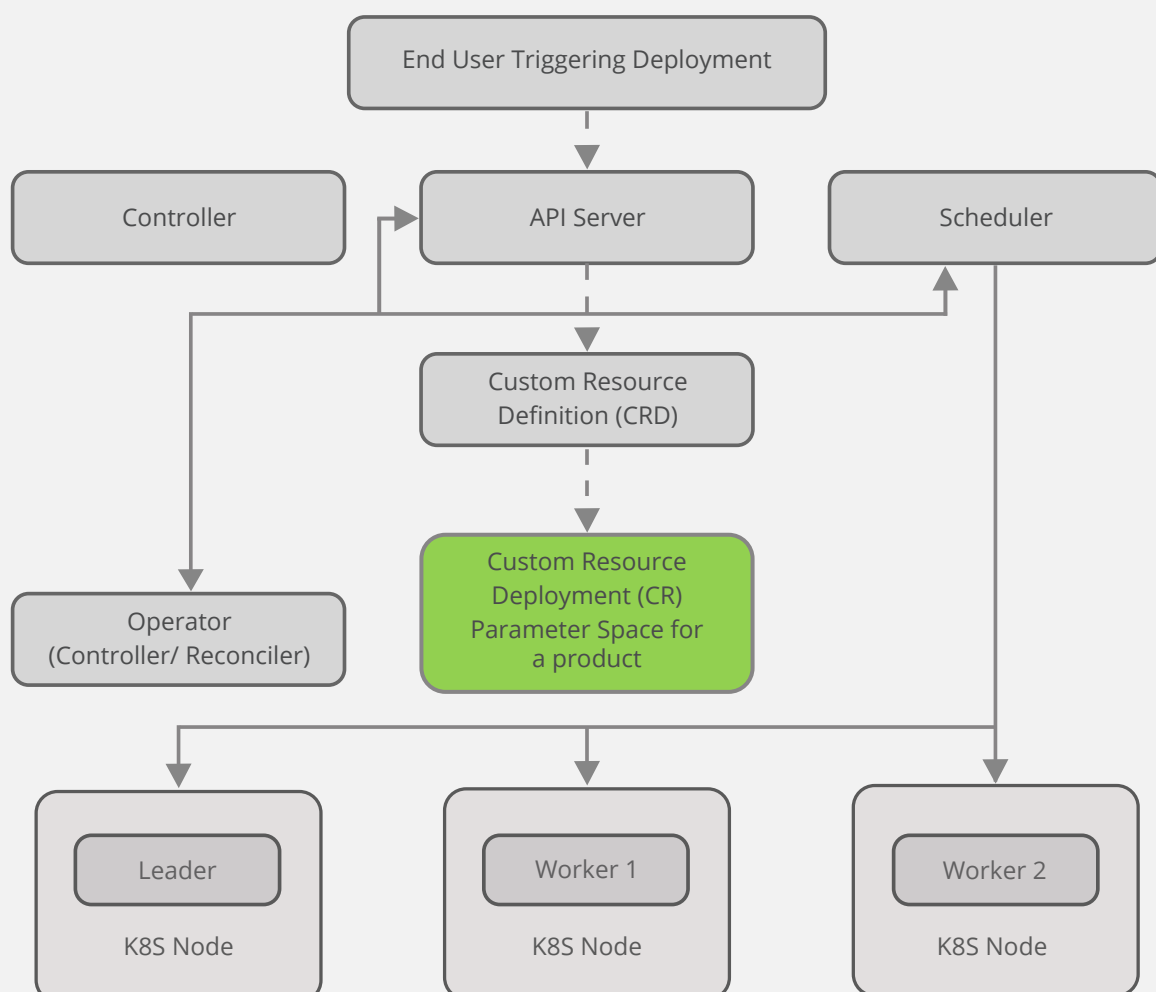
1. For each product, engineers decided to develop Custom Resource Definitions (CRDs) which would consist of customized feature sets supported by specification 'parameters' for end customer use cases.
2. GS Lab decided to use Kubernetes operators to integrate CRDs. These operators allowed customization of the product built-up inside it.
3. We explored multiple operator SDKs that could generate skeletons. We decided to use an open source operator-sdk which could generate not only the basic Golang, but could also auto generate the CRDs. This SDK provided the much needed event triggers for the deployment status.

Solution Details

GS Lab worked closely with the customer on the details of the architecture and the deployment functionality. We developed and tested solutions for each product.

1. The solution, although integrated with UI, was decoupled so that it could be deployed independently in any Kubernetes cluster.
2. We developed operators which are tiny pods that run in a namespace and control the deployment of a single product.

3. We finalized parameter sets while keeping in mind the product requirements and implemented them inside Golang APIs.
4. We also Implemented customized eventing structures and heartbeats. When an end customer deployed the product, these provided real-time updates in the form of a progress bar inside the UI.
5. The solution used standard k8s services provided by public clouds, AWS, GCP and Azure. The solution is deployable on k8s clusters hosted on all these 3 cloud service providers, as well as k8s clusters deployed in 'on-premise' environments
6. The operator sdk we used was still in early stages of maturity. We had to constantly monitor and tweak it to keep the system working smoothly.



Impact



Reduced costs



Improved scalability



Architecture extended to platform

1. This solution brings down costs for end customers. They can license products which are required when they actually need them, instead of buying licenses for prolonged periods .
2. End users can run multiple products as and when needed. The infrastructure gets torn down when done. There is no wastage of resources.
3. This solution gives freedom to use public clouds or on-premise infrastructure. The solution is designed to work with AWS, Azure and GCP.
4. The end user gets the flexibility to run the products from UI or from the kubernetes cluster
5. The end user needs to simply invoke a product from the UI and define the configuration. Everything from provisioning to tear-down is automated.
6. The dynamic deployments have made our customer's products more accessible while improving customer experience and stickiness.

Great Software Laboratory (GS Lab) has been the technology partner of choice to 100+ organizations across North America, Europe and Asia-Pacific for over 17 years. Leveraging our expertise in 130+ tools & technologies, we have created 300+ 'first-of-its-kind' solutions to real-world problems. Our 'Beyond code' philosophy ensures that we not only push boundaries of existing technologies but also try out newer problem solving approaches to keep our customers one step ahead of their competitors. Our global team of 1200+ employees is adept at creating 'real value' at each stage of the customer growth journey, right from proof-of-concepts to completely scaled up products. For more information about our solutions & offerings, please visit www.gslab.com

Copyright©2020 Great Software Laboratory. All rights reserved. No part of this document may be reproduced, stored in a retrieval system, transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the express written permission from Great Software Laboratory. The information contained herein is subject to change without notice. All other trademarks mentioned herein are the property of their respective owners.

