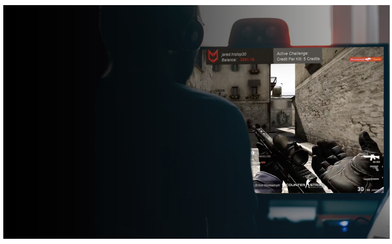
IntrospectData

Case Study: Single Instances to Massively Scalable: EC2 to Kubernetes and EKS

Awesome delivery velocity improves system stability and consistency while driving better customer engagement with better visibility into how it all works.



Challenger Interactive

Industry: Gaming and Entertainment
Location: San Diego, California

Company Bio
Today players of online games compete and trade digital assets ("skins") as an alternative to direct money transactions. Because there's a fluctuating market to convert each skin back into cash, this method of gaming for profit is a risky endeavor. In addition to value uncertainty, the process is also difficult and time consuming for players to arrange matches and skin exchanges. Despite this is long, drawn out, complex process, skins "trading" is utilized by millions of people and is expected to grow to $12+ billion per year by 2020.

Challenger turns 5 difficult steps into one transparent, real-time, streamlined process inclusive of automated score keeping and payouts across multiple games.

Their on-demand client application automates the process, enabling challengers to play when and how they want. Challenger introduces a new approach to gaming which enables casual and professional players to easily compete on a level field.

Overview

Rapid testing, promotion, validation, and deployment of components in a container-based solution which allows for increased deployment velocity from environment to environment without loss of service or up-time. Implementing a scheduling and management framework for container deployments through open-source solutions.

> "Deployment velocity can seem like a small thing but we were incredibly surprised with how it changed our perspective on how we deliver to customers. Ideas in the morning can turn into features in production that day - IntrospectData helped us align to make that a possibility" - **Chris Slovak**, CEO Challenger Interactive

> "We weren't even thinking about stuff that was 'cutting edge' - we thought that just getting it to work had to come first. IntrospectData helped us achieve both faster than we imagined."
>
> - ***Chris Andres,***
> *COO*
> *Challenger Interactive*

The Challenge

As Challenger continues to grow, the company need to be able to deploy, test, validate and promote components rapidly in their micro-service and component-based architecture. Features such as rolling updates, cleanly managed service endpoints and names as well as the ability to scale on a more granular level than at the instance level were key drivers in deciding to adopt a container-based solution.

That containers could be promoted from environment to environment and provided an opportunity to allow developers to debug with the solution that was in production in a repeatable and reliable manner were critical as well. This provided a means to introduce containers into the development workflow while working out how to leverage them in production.

The Approach

IntrospectData has a track record of building Kubernetes-based deployments on everything from bare metal to virtualized private data-centers and the cloud. Given the timing of this engagement in mid May of 2018, it was decided that we would first deploy to a standard cluster running on AWS' EC2 instances deployed via the kops command line deployment tool.

Our goal was to prove the viability of running all of Challenger's infrastructure on K8s before moving it to a managed Kubernetes service (Amazon EKS, released June 8, 2018). We also chose intentionally to make the move to EKS after we'd gained positive results in working on top of Kubernetes.

This created a 2-step migration. The initial migration from an Elastic Beanstalk and EC2-based deployment was migrated to Kubernetes which was then moved to EKS (late September, 2018).

The Solution

Part 1 - EC2 and Elastic Beanstalk migration to Kubernetes

Getting Challenger's applications ready for use on Kubernetes required two major updates to existing applications:

- Updates to how configuration data is provided to the application - we chose to standardize on environment variables
- Adding Dockerfiles, a build and a publish step to the existing CI scripts in order to build container images and push them to Amazon's Elastic Container Registry

In addition to this, we created a centralized repository to handle a GitOps-style process of managing Challenger's Kubernetes clusters from one place. This included Github authentication and workflow rule enforcement so that all changes require review before merging to master. From there, containers would be automatically deployed to the 'staging' cluster with an approval step in CircleCI controlling whether or not that release would be pushed to the production cluster.

Deployments themselves were organized by a custom Helm chart which encapsulated the deployment of each service and managed the application of configuration variables based on existing Kubernetes ConfigMaps and Secrets while also handling component-level decisions such as whether or not to create a Service for HTTP access.

Over the course of 3 weeks, IntrospectData deployed, configured and validated the Kubernetes cluster while partnering with application developers to coach and even update and validate the CI build process for the new staging environment. One week later, mostly due to scheduling concerns, the production Kubernetes cluster was deployed using kops and production was migrated to the new environment.

Part 2 - Kubernetes to Elastic Kubernetes Service (Amazon EKS)

Given that Amazon had released its own managed Kubernetes offering, we agreed that using this service additionally lower management overhead and would significantly improve stability over the long-term. While this was a migration to another Kubernetes cluster, we were incredibly pleased that it was performed with less than a few seconds of downtime. From a process perspective IntrospectData followed a straightforward process to achieve this:

1. Bring up EKS K8s cluster, configure basics such as log shipping, monitoring, etc.
2. Deploy back-end services to EKS cluster and remove from legacy cluster
3. Deploy front-end API services to EKS cluster and update public DNS to point to new cluster
    1. Once all traffic had migrated to new cluster, remove API services from legacy cluster
4. Deploy CMS to EKS cluster, update DNS to point to new end point and wait for traffic to migrate, delete from legacy cluster when complete
5. Validate all components running on new cluster and deleted from old
6. Tear down legacy cluster

The Challenger team successfully performed a major content update while the site was 'in-flight' which caused the actual final migration time to take just over 10 minutes of active management time to successfully migrate:

- 3 Namespaces
- 29 Deployments
- 77 Pods
- 34 Services
- 13 Ingress Rules

The Results

The migration events themselves were, largely, non-events. With no measurable downtime during the transitions themselves, both migrations were seen as very successful. Furthermore, moving to a managed Kubernetes service is allowing Challenger to leverage best-of-breed technologies and continue to build their microservices architecture with less overhead than with their previous EC2 and Elastic Beanstalk-based solution.

Measuring the results of this effort show some interesting detail related to developer use of tooling. The number, size and pace of deployments for Challenger skyrocketed - Continuous Integration and Deployment system utilization time jumped at every step

Project Stage	Active CI/CD System Utilization Time
Instance-Based Deployments	811.44 min/mo
Kubernetes on EC2 (kops)	2321 min/mo (311% increase over previous)
Amazon EKS	3160 min/mo (25% increase over previous)

More deployments with smaller changes has a tendency to reduce the risk of any given deployment and with the process automated via CircleCI, Amazon's Elastic Container Registry and Helm, the process was both highly consistent for the same project but also among the entire set of projects as well. Put simply, building in this level of automation allowed the team to build within the newly set standard which reduced the number of different deployment processes and even application architectural patterns to a handful of known best practices.

From a service availability perspective, average externally measured available of key services continues to improve as the Challenger team learns to leverage the platform:

Date	% Up-time
November 2018 (Through 19-Nov)	100.00%
October 2018	100.00%
September 2018	99.99%
August 2018	99.88%
July 2018	99.80%

> "Moving to K8 has given us incredible visibility into our applications in real time which means we're able to provide feedback to developers and set expectations with customers very quickly. Improving our time to resolution was critical for our team and our customers - simply changing the deployment platform really helped us 'level up."
>
> - ***Jared Hislop,***
> *Senior Product Manager*
> *Challenger Interactive*