HENTSŪ
SYSTEMATIC AGILITY

CASE STUDY

Hedge Fund DevOps and Multi-Cloud Kubernetes

INITIAL COMPONENTS

- 4-person in-house development team
- Alternative data sets
- In-house VMware and AWS cloud
- Proprietary Python and Tensor-Flow code
- MongoDB for data and results

The Challenge

In this case study we shall examine the uses and advantages of Docker architecture and the benefits of a Kubernetes cluster.

One of our existing clients had been using their own machine learning strategies to develop an in-house platform in order to produce trading signals from a range of alternative datasets. The 4-person development team had been running for six months, working on building a suite of Python applications and Big Data processing pipelines, both on premise and in Amazon AWS cloud.

The client approached Hentsū to extend their own small development team, and to improve the overall software development. The pace of functionality releases was slow, the applications were suffering from complexity and the code quality was poor.

The in-house developers were experiencing significant struggles to work as a unified team. Code was being committed and deployed with broken library dependencies, resulting in manual fixes every release to ensure code was running correctly. The applications were disjointed and inconsistent, with very loosely coupled sets of scripts, software, and services. There was no robust deployment of the applications, and once deployed there was often the need to intervene manually.

Key Requests

1. Deliver more functionality, faster
2. Reduce code bugs and improve stability of the application
3. Deploy the application faster to any environment

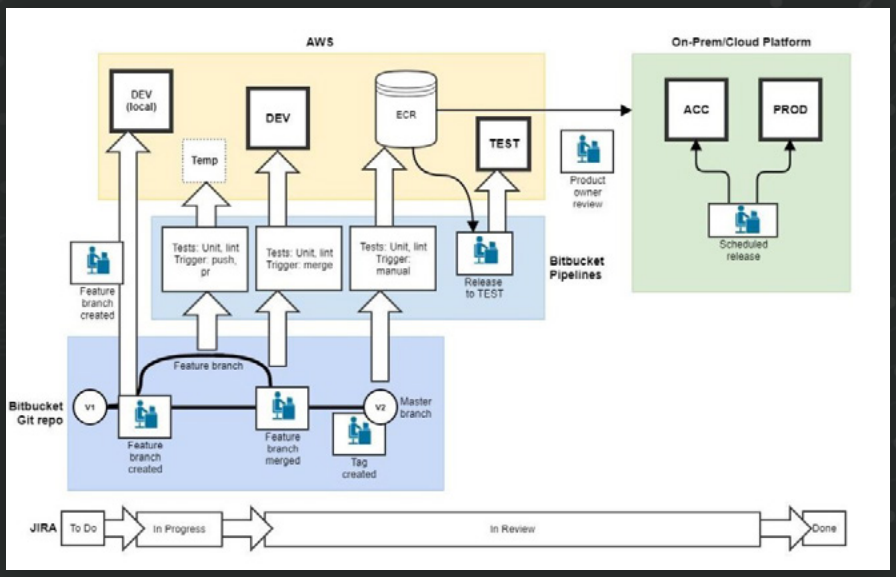
TECHNOLOGIES USED

- Atlassian JIRA – roadmap, sprints and issue tracking
- Atlassian Confluence – documentation
- Atlassian BitBucket – source code
- Atlassian Pipelines – build, testing and deployment
- Amazon AWS – Elastic Container Registry (ECR) and environments

The Solution - DevOps Workflow

Hentsū promptly identified the need to deploy the most efficient Continuous Integration/Continuous Deployment (CI/CD) pipeline, as fast as possible. The focus had to be on feature development rather than tooling, as well as on the removal of any difficulties in getting great code from the developers – quickly.

As a first phase, Hentsū deployed a development workflow, which was based around the Atlassian suite of products. The goal was to enable rapid iteration of the team's code, whilst ensuring overall software testing, quality control and integration. The workflow relied on properly defined environments – Development, Testing, Acceptance, Production (DTAP).



Deploying these steps and enforcing the code flow produced an instant improvement in both the collaboration across the team, and the quality of software. There was much improved visibility on what any one developer was pushing into the branches and its effect on the overall software.

Separately, Hentsū worked with the developers to restructure the Git software repositories logically into specific areas of concern (apps/services/dependencies). Each repository would contain its own tests, dependency tree and Bitbucket pipeline YAML config. This enabled more autonomy in development, whilst retaining efficient control over the cross-platform dependencies and testing.

Finally, the agile methodology was improved through the use of clearer structure and scheduling. Ensuring better code quality and higher feature throughput was key. So, there was a focus on activities such as sprint start, standups, development time, smoke tests, and backlog refinement. Product ownership and feedback were improved within the business. This was accomplished by clearly identifying each feature owner and involving them in the sprint process.

TECHNOLOGIES USED

- Docker – containerisation
- Angular – front end interfaces
- Kubernetes – cluster management
- Helm – package management
- Kubespray and Ansible – Kubernetes deployment
- AWS Elasticsearch, Prometheus and Grafana – application and system monitoring

Docker Architecture

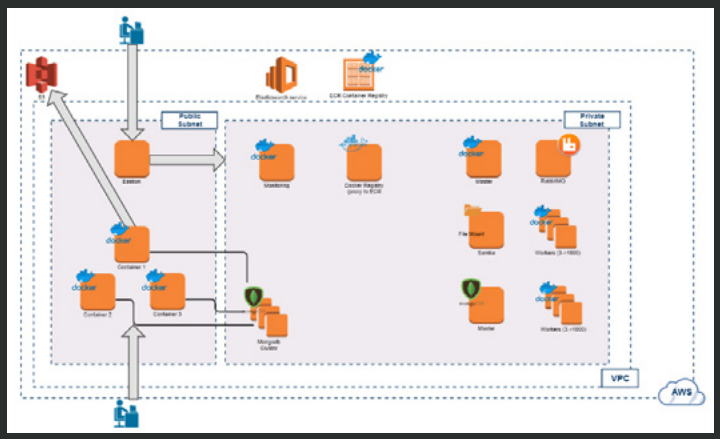
Hentsū deployed a 3-person team to augment the in-house developers. The team brought Python and containerisation expertise to re-architect the applications and make them more stable, self-contained, and easily distributed across environments. Whilst the Git repositories were restructured, the corresponding Docker images were rolled out for each specific service.

Docker registry and Elasticsearch services from AWS were used to help with the deployments and monitoring, without having to stand up infrastructure. To help with the deployment, scaling and management of the Docker containers, a Hentsū customised Kubernetes platform was rolled out. The customisation also allowed the client to overcome the limitations in the AWS EKS service and integrate VMware environments. This ensured consistency of deployment and tooling, but also allowed for the applications to be deployed to Azure and Google Cloud Platform (GCP).

Kubernetes Cluster

Using a Kubernetes cluster, Hentsū enabled the additional functionality of automatic scaling. Worker nodes were able to run as a static number, which could be useful on-premise to limit the impact on other resources. However, as the Python code had the capability to work in parallel, deploying autoscaling allowed the number of nodes to ramp up quickly based on the queue of work. If there was a bigger queue of incoming data to process, the entire cluster could autoscale to thousands of nodes if needed. Each individual worker node was a small enough unit of compute/memory that the autoscaling for different loads of work became very linear and cost efficient.

Combining the Hentsū Kubernetes cluster management and AWS meant that the client had many more options to manage the workloads. The cluster could rapidly adapt between specific GPU enabled worker instances, whilst simultaneously the client was able to use the AWS Spot market for cheaper resources when available and move the application between regions or even cloud providers. Another new possibility this opened up was deploying to bare metal, allowing for VMware to be discarded



Security Considerations

With the ability to run the Python code in various cloud platforms, and potentially also utilise Platform as a Service (PaaS) from the cloud providers, the security of the intellectual property was of concern. Hentsū deployed the entire solution with strict adherence to its own developed ISO 27001 cloud security checklist. Encryption was built into the application from the start, and all user access controls were tied back to the client's corporate Active Directory.

Impact

The improvements and options Hentsū enabled meant developers were happier and substantially more productive with their coding. We've employed both Docker architecture and Kubernetes cluster successfully. Additionally, the team collaboration and the engagement of the business stakeholders meant that more features than initially planned were released to the end users, and in a faster timeframe.

The number of bugs which were raised in production from each two-week release cycle was reduced from an average of over 30, to below 2. This code quality success was ensured by the improvements Hentsū implemented to the scheduling and structure, such as the Pipeline unit tests, the consistency of the development and acceptance environments, and the rigorous smoke tests.

The greatest impact was the overall delivery of the project. When Hentsū was first engaged the estimate for remaining time to deliver the project was 18-24 months; however with the changes delivered by Hentsū the project was completed in under 6 months.

HENTSŪ LONDON

30 Crown Place
London, EC2A 4EB
United Kingdom
+44 203 857 1630

HENTSŪ NEW YORK

600 Fifth Avenue
New York, NY 10020
United States
+1 315 257 4284

HENTSŪ BOSTON

125 High Street
Boston, MA 02110
United States
+1 315 257 4284