Integration and Implementation Team
Final Project Presentation
Dec. 10, 2019

Rahul Agarwal, Hadeel Albahar, Eric Roth, Malabika Sen, Lixing Yu
Outline

1. Introduction
2. Objective
3. Background
4. Timeline and Milestones
5. Approach
6. Implementation
7. Testing and Evaluation
8. Challenges
9. Future Work
10. Summary
Introduction

● Let’s build a state-of-the-art IR system using latest technology
  ○ Docker container cluster; Rancher; Kubernetes / Kubectl
  ○ ElasticSearch - info retrieval; Kibana - visualization

● Use this system to support two collections:
  ○ ETDs
  ○ Tobacco Settlement Documents

● Project Teams: CME (ETDs); CMT (Tobacco); ELS (ElasticSearch); FEK (Front-end & Kibana); TML (Text Analysis & ML); INT (Integration & Implementation)
Objectives

1. Design and deploy Docker containers (containerize an IR system)
2. Manage the Kubernetes cluster on the CS Cloud via Docker, kubectl, and Rancher
3. Facilitate and manage data sharing between containers and Tobacco & ETDs VMs by using Ceph storage
4. Facilitate the evaluation and testing (CI/CD) of cluster components
5. Support a pipeline for ingestion of new collection documents (Kafka)
6. Deploy all cluster components seamlessly
Background: Containers

- A **lightweight** alternative to VMs
- Linux **cgroups** (resource limitation) and **namespaces** (container visibility, isolation)
- Enable a **quick, reliable, & consistent** application deployment regardless of deployment environment.

**Docker** is the leading container management framework.

![Application Containers: Total Market Revenue ($M)](chart)

Source: 451 Research

Source: 451 Research’s Market Monitor: Cloud-Enabling Technologies - Application Containers, November 2018
Infrastructure

Source: Docker Blog

Kubernetes & Docker work together to build & run containerized applications

/mnt/ceph

Tobacco VM

ETDs VM
**Timeline**

**Sept**
- **Namespaces and Projects**
  Assigned teams to respective workspace on Rancher’s testing cluster.

**Oct**
- **Expose Service as LoadBalancer**
  Exposed services deployed through external IP and documented the approach.
- **CentOS Container and Kubectl**
  Deployed the basic CentOS container and configured kubectl.
- **Custom Docker Deployment**
  Deployed various custom dockers via Dockerfile, Rancher Catalog and Docker Hub.
Timeline continued

Changes to ElasticSearch Configuration
Researched and implemented the changes requested onto ES container by FEK team

CI/CD and Stress Testing
Researched and presented a demo on how to leverage GitLab's CI/CD pipeline
Used Locust for stress testing frontend application

Dec
Kafka
Deployed Kafka as a service and built a framework to be used by other teams

Nov
MySQL and Frontend Application
Deployed the first version of the frontend application

Nov
Changes to ElasticSearch Configuration
Researched and implemented the changes requested onto ES container by FEK team

Nov
CI/CD and Stress Testing
Researched and presented a demo on how to leverage GitLab's CI/CD pipeline
Used Locust for stress testing frontend application

Nov
MySQL and Frontend Application
Deployed the first version of the frontend application
Approach

1. Containers
   a. Pull existing containers (e.g., Docker Hub, Rancher Catalogs, Helm charts, etc.)
   b. Create, push and pull custom containers (**Dockerfile**)

2. Shared Storage (mounting a **Ceph** volume)

3. Unit tests → as part of a CI/CD framework via **GitLab**

4. Stress testing → **Locust**

5. Automated Data Pipeline → **Kafka**

6. Monitoring changes to directory → **inotify**
Implementation

Total Number of containers deployed: 19
Testing and Evaluation - CI/CD

- Small features released quickly
- Test cases ensure system does not break
- Follows the Build → Test → Deploy model
- Examples of CI/CD pipelines:
  - Travis CI
  - Jenkins
  - GitLab
<table>
<thead>
<tr>
<th>CI/CD Options</th>
<th>GitLab</th>
<th>Travis</th>
<th>Jenkins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of Setup</td>
<td>.yml file helps set it up</td>
<td>Setting up is as easy as creating a config file</td>
<td>Needs elaborate setup</td>
</tr>
<tr>
<td>Hosted Service</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Performance</td>
<td>Supports public and private repositories along with a lot of other options like supporting container registry.</td>
<td>Best choice for open-source project because of ease of use and setup.</td>
<td>Has unlimited customization options.</td>
</tr>
<tr>
<td>Usage</td>
<td>Free for Virginia Tech-owned services</td>
<td>Free for Open Source Project and paid for Enterprise</td>
<td>Free</td>
</tr>
<tr>
<td>Server Machine</td>
<td>Cloud-based</td>
<td>Cloud-based</td>
<td>Server-based</td>
</tr>
</tbody>
</table>
GitLab Pipeline
Testing and Evaluation - Stress Test

- Performance testing or “Negative Testing”
- Focuses on robustness, availability and error-handling under a heavy load
- Different types of stress testing:
  - CPU stress testing
  - Load testing
# Stress Testing Tools

<table>
<thead>
<tr>
<th></th>
<th>JMeter</th>
<th>Locust</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scripting</td>
<td>Supports GUI and scripting</td>
<td>Supports Python coding</td>
</tr>
<tr>
<td>Best For</td>
<td>Performance testing of web applications.</td>
<td>It provides a functionality to check the simultaneous number the system can handle.</td>
</tr>
<tr>
<td>Capability</td>
<td>It works for web applications, servers,</td>
<td>It can perform load testing on multiple</td>
</tr>
<tr>
<td></td>
<td>group of servers, and network.</td>
<td>distributed machines.</td>
</tr>
<tr>
<td>Pricing</td>
<td>Free</td>
<td>Free</td>
</tr>
</tbody>
</table>
Results - Load Testing

### Load Testing Results

**LOCUST**

**Statistics**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th># requests</th>
<th># fails</th>
<th>Median (ms)</th>
<th>Average (ms)</th>
<th>Min (ms)</th>
<th>Max (ms)</th>
<th>Content Size (bytes)</th>
<th># reqs/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>//</td>
<td>676</td>
<td>0</td>
<td>730</td>
<td>809</td>
<td>290.15135765075684</td>
<td>1483.47806930542</td>
<td>13233</td>
<td>51.7</td>
</tr>
<tr>
<td>POST</td>
<td>//login</td>
<td>100</td>
<td>0</td>
<td>850</td>
<td>793</td>
<td>126.93214416503906</td>
<td>1309.4682693481445</td>
<td>13233</td>
<td>6.4</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>776</strong></td>
<td><strong>0</strong></td>
<td><strong>730</strong></td>
<td><strong>807</strong></td>
<td><strong>126.93214416503906</strong></td>
<td><strong>1483.47806930542</strong></td>
<td><strong>13233</strong></td>
<td><strong>58.1</strong></td>
</tr>
</tbody>
</table>

**LOCUST**

**Statistics**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th># requests</th>
<th># fails</th>
<th>Median (ms)</th>
<th>Average (ms)</th>
<th>Min (ms)</th>
<th>Max (ms)</th>
<th>Content Size (bytes)</th>
<th># reqs/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>//</td>
<td>203</td>
<td>158</td>
<td>2400</td>
<td>2584</td>
<td>694.36812400081787</td>
<td>6731.888055801392</td>
<td>13233</td>
<td>16.88</td>
</tr>
<tr>
<td>POST</td>
<td>//login</td>
<td>282</td>
<td>100</td>
<td>2200</td>
<td>2274</td>
<td>243.99256706237793</td>
<td>7160.060167312622</td>
<td>13233</td>
<td>27.5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>485</strong></td>
<td><strong>258</strong></td>
<td><strong>2300</strong></td>
<td><strong>2404</strong></td>
<td><strong>243.99256706237793</strong></td>
<td><strong>7160.060167312622</strong></td>
<td><strong>13233</strong></td>
<td><strong>44.38</strong></td>
</tr>
</tbody>
</table>
### Results - Load Testing (contd)

**LOCUST**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th># requests</th>
<th># fails</th>
<th>Median (ms)</th>
<th>Average (ms)</th>
<th>Min (ms)</th>
<th>Max (ms)</th>
<th>Content Size (bytes)</th>
<th># reqs/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>//</td>
<td>1751</td>
<td>0</td>
<td>610</td>
<td>1297</td>
<td>289.03685814208984</td>
<td>21299.317121505737</td>
<td>13233</td>
<td>87.8</td>
</tr>
<tr>
<td>POST</td>
<td>//login</td>
<td>324</td>
<td>0</td>
<td>640</td>
<td>3662</td>
<td>265.34247398376465</td>
<td>20514.59527015686</td>
<td>13233</td>
<td>5.5</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>2075</td>
<td>0</td>
<td>610</td>
<td>1666</td>
<td>265.34247398376465</td>
<td>21299.317121505737</td>
<td>13233</td>
<td>93.3</td>
</tr>
</tbody>
</table>

**LOCUST**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th># requests</th>
<th># fails</th>
<th>Median (ms)</th>
<th>Average (ms)</th>
<th>Min (ms)</th>
<th>Max (ms)</th>
<th>Content Size (bytes)</th>
<th># reqs/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>//</td>
<td>986</td>
<td>0</td>
<td>450</td>
<td>743</td>
<td>283.10680389404297</td>
<td>11792.44613647461</td>
<td>13233</td>
<td>64.3</td>
</tr>
<tr>
<td>POST</td>
<td>//login</td>
<td>343</td>
<td>0</td>
<td>550</td>
<td>1550</td>
<td>278.5482406616211</td>
<td>11010.176658630371</td>
<td>13233</td>
<td>17.6</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>1329</td>
<td>0</td>
<td>460</td>
<td>951</td>
<td>278.5482406616211</td>
<td>11792.44613647461</td>
<td>13233</td>
<td>81.9</td>
</tr>
</tbody>
</table>
Migration to Production Cluster

Dockerfiles → Docker Build → Docker images → Push to DockerHub

docker-compose.yml

```
version: "2"
services:
  foo:
    build: "./build"
    image: docker.io/foo/bar
```

kompose up → Kubernetes
Implementation Challenges / Limitations

- CS cloud and hosted container cluster is always in flux
- Unstable cloud/cluster led to many issues across all teams, which we had to address
- Containers: Some teams based their development in VMs; therefore, parameters for container creation were not available
- Unit tests not available in time to implement full system CI/CD
- Ingestion was not a design priority early enough for it to be connected with Kafka; earlier collection efforts had to be prioritized for doc processing
- Production environment (teaching cluster) was not sufficient
- We lost a team member early in the semester
Conclusion

- Designed and deployed Docker containers
- Facilitated and manage data sharing between containers and Tobacco & ETDs VMs by using Ceph File System
- Facilitated the evaluation and testing (CI/CD) of cluster components
- System stress test proved Front End’s robustness
- Deployed Kafka infrastructure and designed data ingestion pipeline
- Successfully handled support requests from other teams
Future Work

- Complete the containerization of cluster components
- Integrate Kafka (Kafkacat producer and consumers) into all the producers and consumers scripts to enable an automated data pipeline
- Motivate teams to adopt GitLab
- Deploy CI/CD pipeline for each team
- Base new system tests on sample user logs
- Deploy all IR system components in Teaching cluster (Production)
Thank You

Questions?