CS5604 - Information Storage and Retrieval
Virginia Tech, Blacksburg, VA 24061

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)**

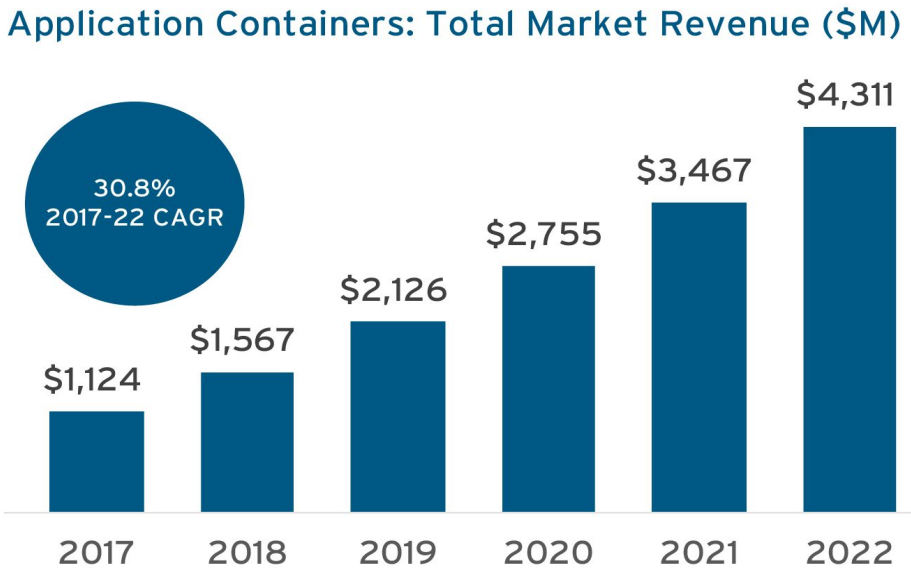
Objectives

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.

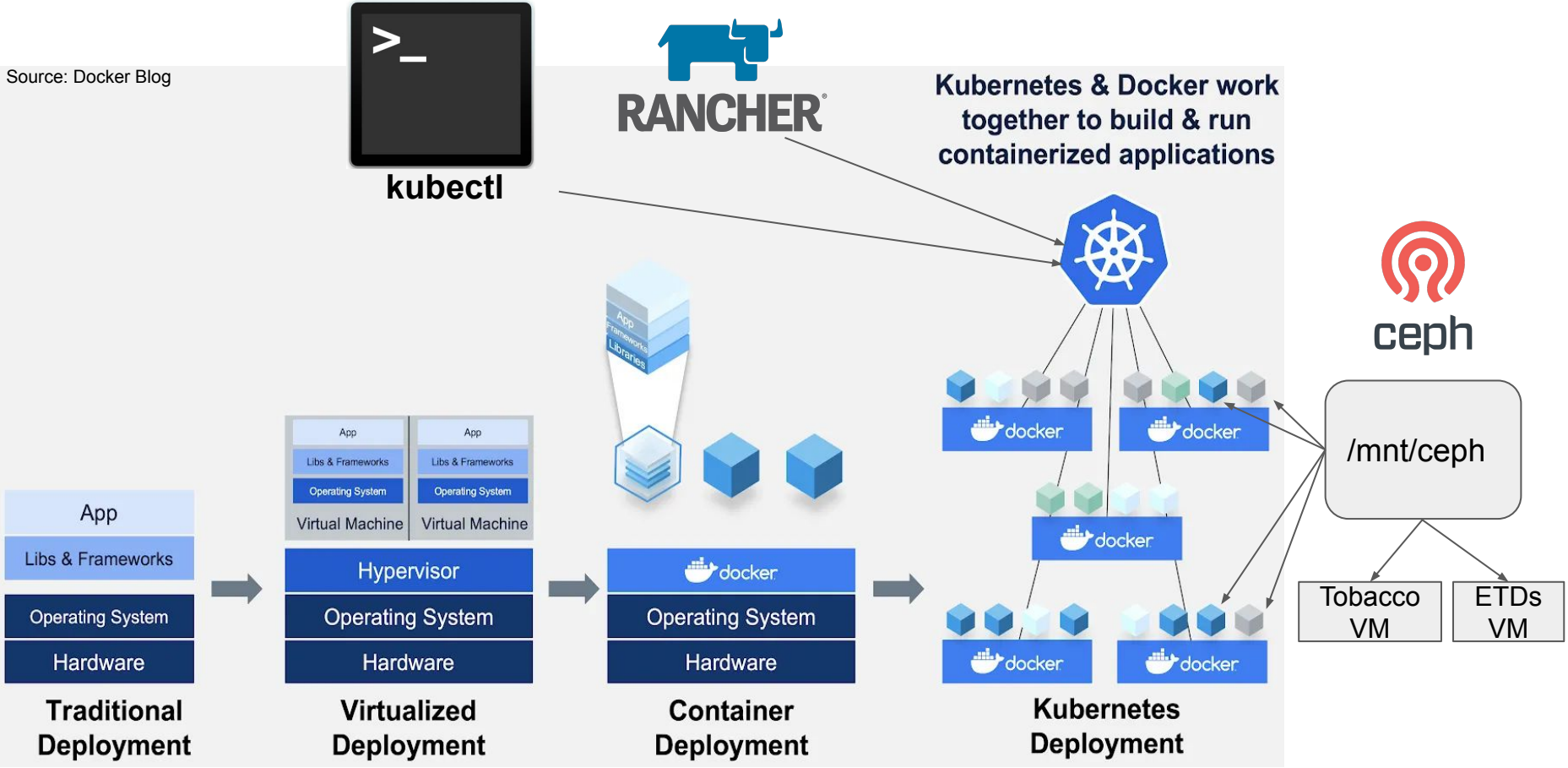
Docker is the leading container management framework.



Source: 451 Research

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

Infrastructure



Timeline

Namespaces and Projects

Assigned teams to respective workspace on Rancher's testing cluster.

Sept

Sept

CentOS Container and Kubectl

Deployed the basic CentOS container and configured kubectl

Expose Service as LoadBalancer

Exposed services deployed through external IP and documented the approach

Oct

Oct

Custom Docker Deployment

Deployed various custom dockers via Dockerfile, Rancher Catalog and Docker Hub

Timeline continued

Nov

Changes to ElasticSearch Configuration

Researched and implemented the changes requested onto ES container by FEK team

Nov

MySQL and Frontend Application

Deployed the first version of the frontend application

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

Dec

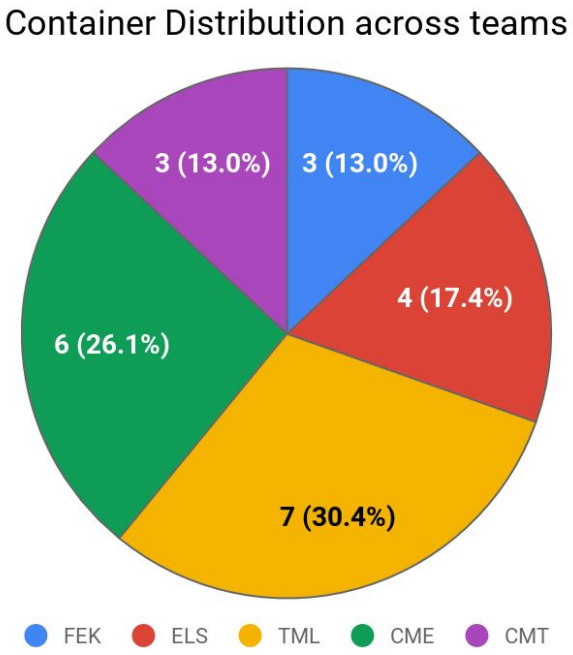
Kafka

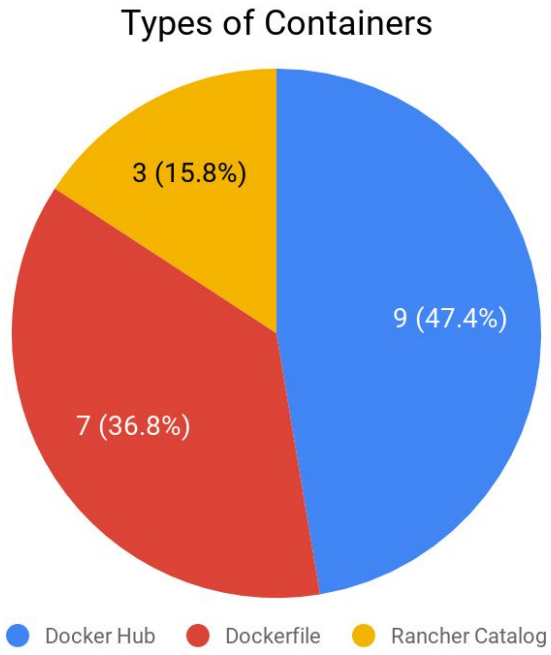
Deployed Kafka as a service and built a framework to be used by other teams

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

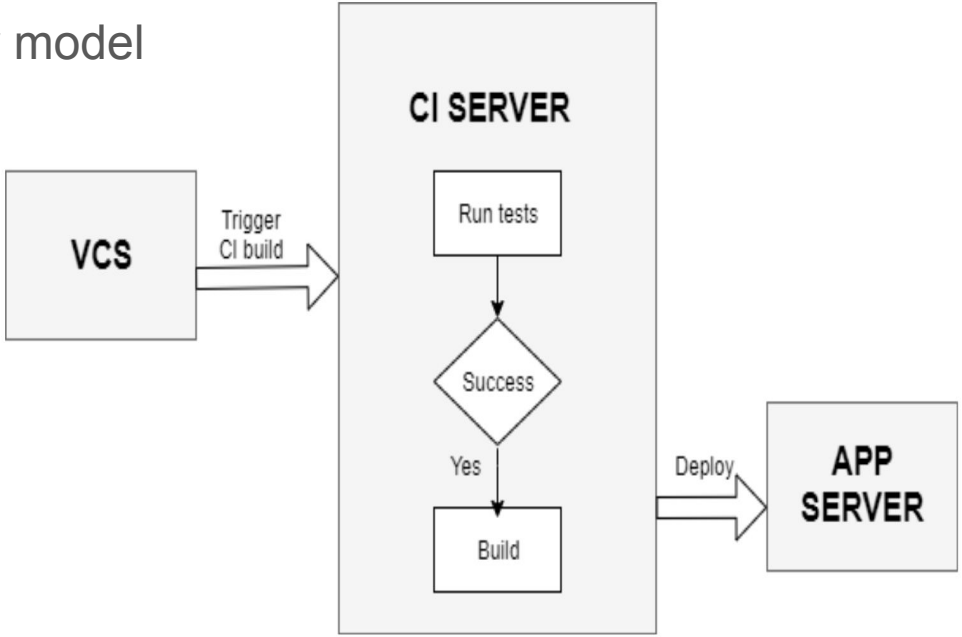




Total Number of containers deployed : 19

Testing and Evaluation - CI/CD

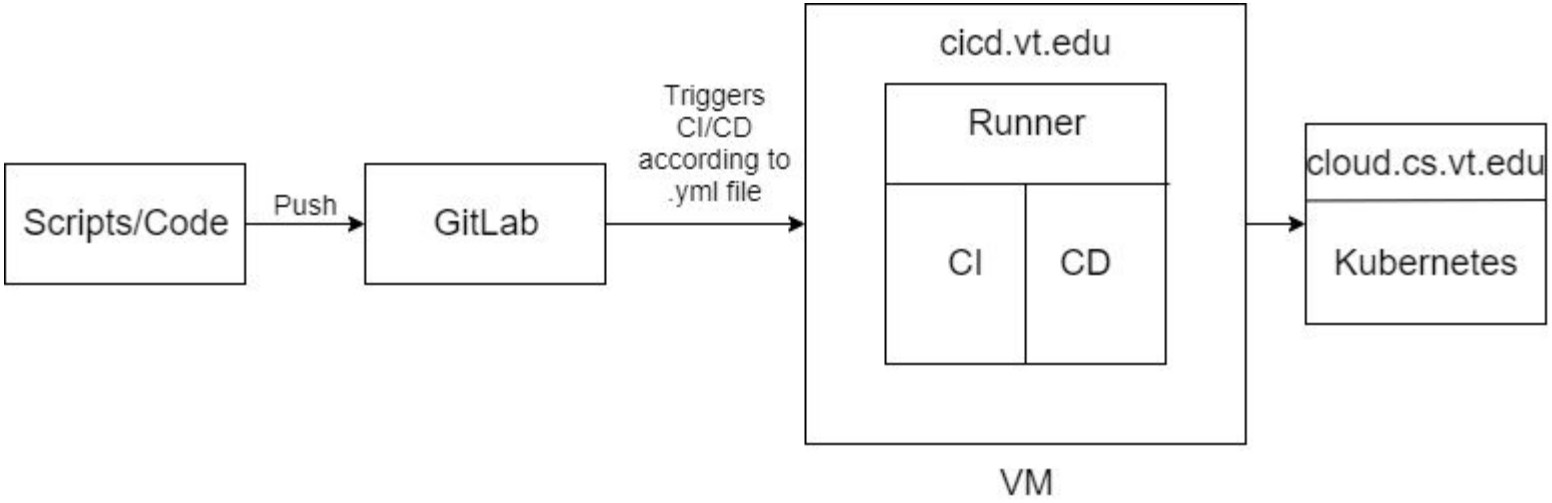
- Small features released quickly
- Test cases ensure system does not break
- Follows the Build → Test → Deploy model
- Examples of CI/CD pipelines:
  - Travis CI
  - Jenkins
  - GitLab



CI/CD Options

	GitLab	Travis	Jenkins
Ease of Setup	.yml file helps set it up	Setting up is as easy as creating a config file	Needs elaborate setup
Hosted Service	Yes	No	Yes
Performance	Supports public and private repositories along with a lot of other options like supporting container registry.	Best choice for open-source project because of ease of use and setup.	Has unlimited customization options.
Usage	Free for Virginia Tech-owned services	Free for Open Source Project and paid for Enterprise	Free
Server Machine	Cloud-based	Cloud-based	Server-based

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

	JMeter	Locust
Scripting	Supports GUI and scripting	Supports Python coding
Best For	Performance testing of web applications.	It provides a functionality to check the simultaneous number the system can handle.
Capability	It works for web applications, servers, group of servers, and network.	It can perform load testing on multiple distributed machines.
Pricing	Free	Free

Results - Load Testing

LOCUST

HOST
http://2001.0468.0c80.6102.0001.7015.a60f.cf44.ip6.name:3000/

STATUS
RUNNING
100 users
Edit

RPS
58.1

FAILURES
0%

STOP

Reset Stats

Statistics | Charts | Failures | Exceptions | Download Data

Type	Name	# requests	# fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Content Size (bytes)	# reqs/sec
GET	//	676	0	730	809	290.15135765075684	1483.47806930542	13233	51.7
POST	//login	100	0	850	793	126.93214416503906	1309.4682693481445	13233	6.4
	Total	**776**	**0**	**730**	**807**	**126.93214416503906**	**1483.47806930542**	**13233**	**58.1**

LOCUST

HOST
http://2001.0468.0c80.6102.0001.7015.a60f.cf44.ip6.name:3000/

STATUS
RUNNING
500 users
Edit

RPS
44.38

FAILURES
35%

STOP

Reset Stats

Statistics | Charts | Failures | Exceptions | Download Data

Type	Name	# requests	# fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Content Size (bytes)	# reqs/sec
GET	//	203	158	2400	2584	694.3681240081787	6731.888055801392	13233	16.88
POST	//login	282	100	2200	2274	243.99256706237793	7160.060167312622	13233	27.5
	Total	**485**	**258**	**2300**	**2404**	**243.99256706237793**	**7160.060167312622**	**13233**	**44.38**

Results - Load Testing (contd)

LOCUST

HOST
http://2001.0468.0c80.6102.0001.7015.a60f.cf44.ip6.name:3000/

STATUS
RUNNING
500 users
Edit

RPS
93.3

FAILURES
0%

STOP

Reset Stats

Statistics | Charts | Failures | Exceptions | Download Data

Type	Name	# requests	# fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Content Size (bytes)	# reqs/sec
GET	//	1751	0	610	1297	289.03865814208984	21299.317121505737	13233	87.8
POST	//login	324	0	640	3662	265.34247398376465	20514.59527015686	13233	5.5
	Total	**2075**	**0**	**610**	**1666**	**265.34247398376465**	**21299.317121505737**	**13233**	**93.3**

LOCUST

HOST
http://2001.0468.0c80.6102.0001.7015.a60f.cf44.ip6.name:3000/

STATUS
RUNNING
1000 users
Edit

RPS
81.9

FAILURES
0%

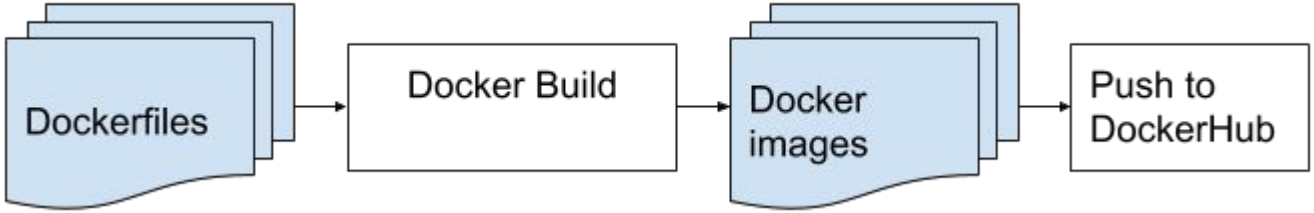
STOP

Reset Stats

Statistics | Charts | Failures | Exceptions | Download Data

Type	Name	# requests	# fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Content Size (bytes)	# reqs/sec
GET	//	986	0	450	743	283.10680389404297	11792.44613647461	13233	64.3
POST	//login	343	0	550	1550	278.5482406616211	11010.176658630371	13233	17.6
	Total	**1329**	**0**	**460**	**951**	**278.5482406616211**	**11792.44613647461**	**13233**	**81.9**

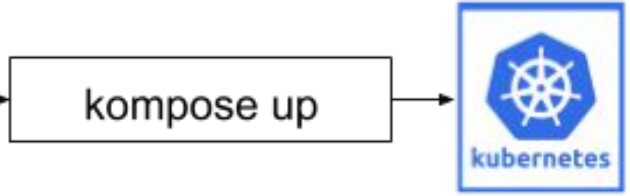
Migration to Production Cluster



docker-compose.yml

```
version: "2"

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



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?