INT Final Presentation

CS 5604 Information Retrieval

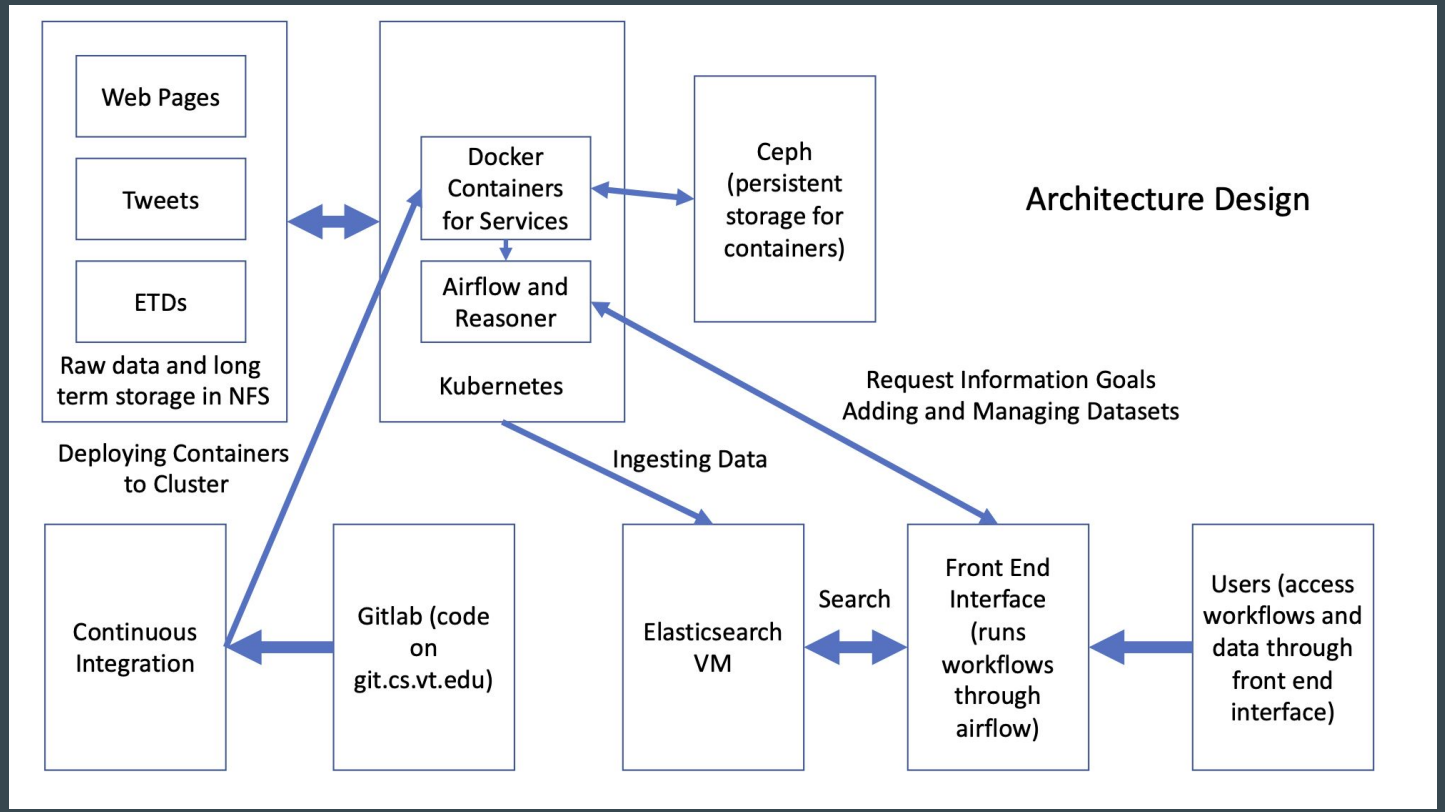
Integration and Implementation Team

Alex Hicks, Mohit Thazhath, Suraj Gupta, Cherie Poland, Xingyu Long, Hsinhan Hsieh, Yash Mahajan

Virginia Tech, Dept. of Computer Science, Blacksburg, VA 24061

12/7/20

Architecture Design



Managed Virtual Machines

- Elasticsearch
  - elasticsearch.cs.vt.edu
  - Elasticsearch in Docker
    - Kibana
- KGI
  - kgi.cs.vt.edu
  - Gitlab Runner

```
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS
           NAMES
615682167def        gitlab/gitlab-runner   "/usr/bin/dumb-init …"   23 hours ago        Up 23 hours
         gitlab_gitlab-runner_1
```

```
CONTAINER ID        IMAGE                                                  COMMAND                  CREATED
STATUS                 PORTS                                  NAMES
34c8596b6ead        docker.elastic.co/elasticsearch/elasticsearch:7.10.0   "/tini -- /usr/local…"   2 weeks ago
Up 2 weeks             9200/tcp, 9300/tcp                     es03
0cbacaa3dd3e        docker.elastic.co/elasticsearch/elasticsearch:7.10.0   "/tini -- /usr/local…"   2 weeks ago
Up 2 weeks (healthy)   0.0.0.0:9200->9200/tcp, 9300/tcp       es01
1436ccd31bde        docker.elastic.co/elasticsearch/elasticsearch:7.10.0   "/tini -- /usr/local…"   2 weeks ago
Up 2 weeks             9200/tcp, 9300/tcp                     es02
```

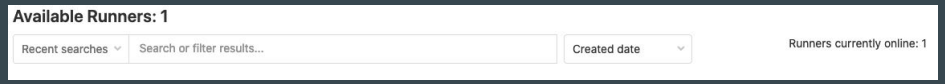
Elasticsearch

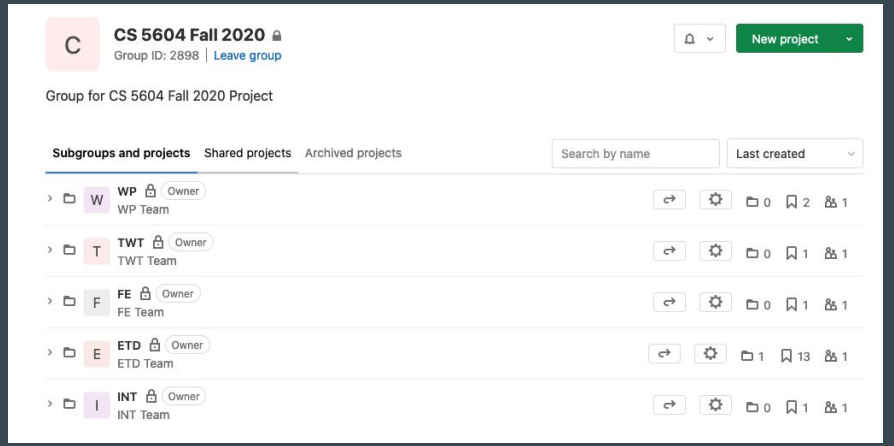
- Index data from content teams
- Store ingested documents and metadata
- Provide access to Front End for display and interaction

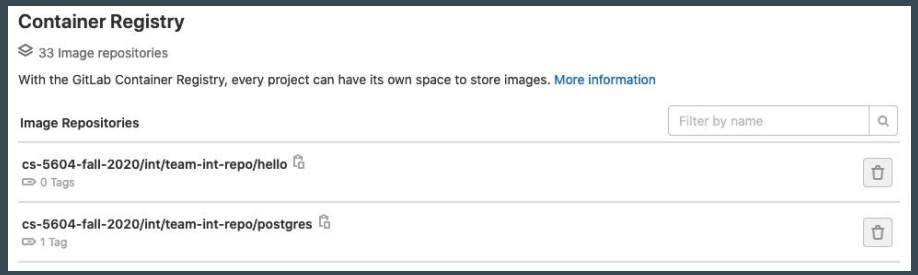
```
green open twt              bFB082MPS-aVHbz571dfCA 1 1 3123 0 775.9kb 387.9kb
green open .security-7      Hki0jBtHS-CVaePQBSLCFA 1 1    7 0  51.3kb  25.6kb
green open test-index       _lR0i2JjTrWLkubfQDLvTA 1 1    1 0   9.7kb   4.8kb
green open etd              _YY3NKFaTCeulBEnEmxpvA 1 1 1491 0   8.6mb   4.3mb
green open fe_etd_metadata  xW5lbwBRT86TRFMVcWheKA 1 1    9 0 122.8kb  61.4kb
```

Gitlab

- VT CS hosted Gitlab instance
  - git.cs.vt.edu
- Provides almost complete DevOps system
  - Version Control
  - Runner
  - Container Registry

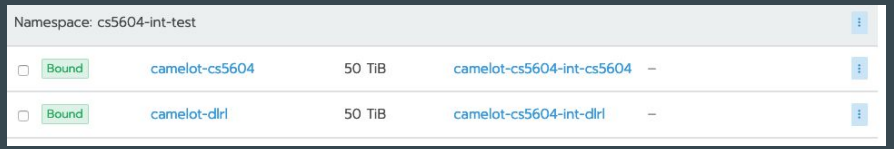


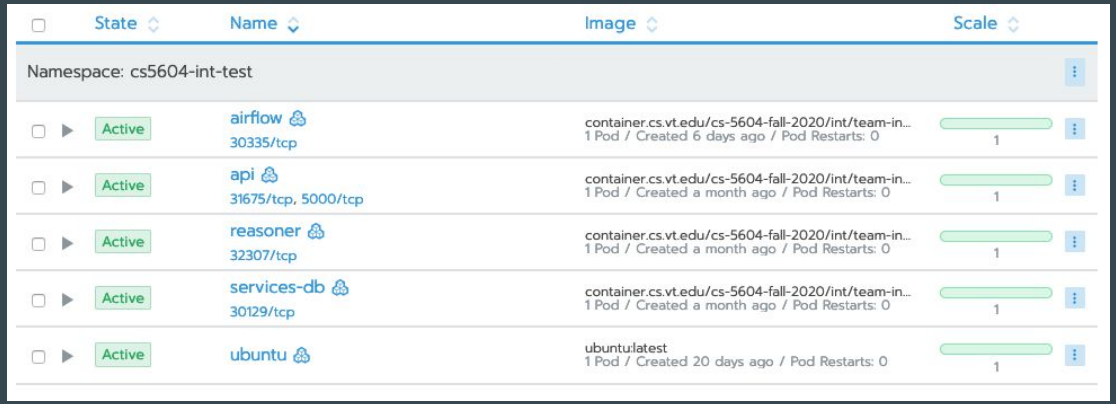




Kubernetes

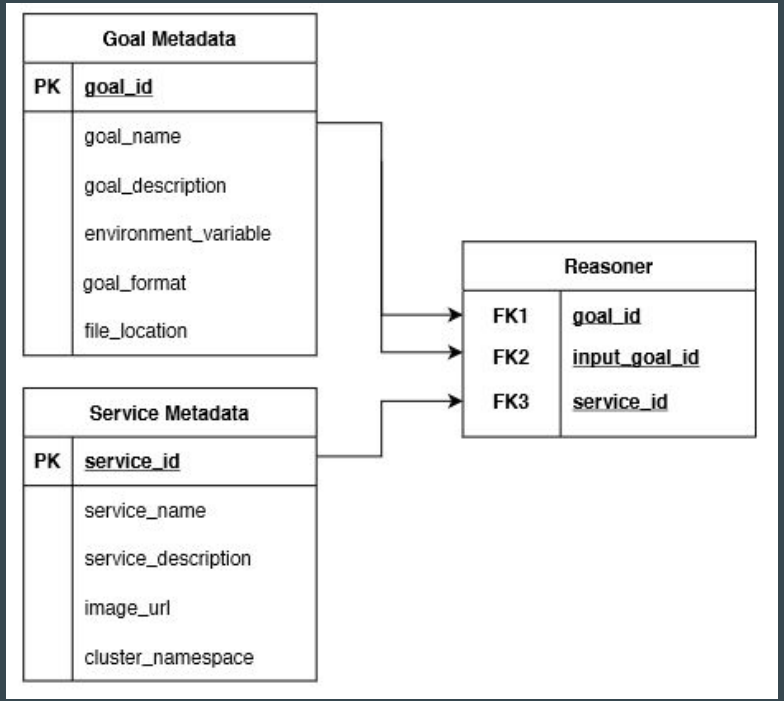
- Created projects for each team
- Volume mounts
- Airflow integration
- Load Balancing for Front End interface





Postgres

- Postgres instance running on Kubernetes
  - Service and Goals tables
  - Reasoner
  - Airflow
- Database was also used by other teams
  - Front End - users and permissions



Using the API

Service API

- CRUD API to pass to Front End for registering services

```python
@api.route('/services/', methods=['GET'])
def get_services():
    '''
    Returns the list of available services within
    the service_metadata table in the db
    '''
    services = services_model.select()
    return jsonify(services)
```

```python
@api.route('/services/', methods=['POST'])
def create_services():
    '''
    Create a service within the service_metadata table in the db

    Inputs
    ------
    service_name : str
        Name for the service
    service_description : str
        Description of the use case for the service
    image_url : str
        Docker url
    cluster_namespace : str
        Namespace in cluster where pod will be deployed
        Default cs5604-int-test
    owned_by : str
        Team name
    '''
    data = request.get_json()
    values = [{
        service_cols[1]: data['service_name'],
        service_cols[2]: data['service_description'],
        service_cols[3]: data['service_format'],
        service_cols[4]: data['service_location'],
        service_cols[5]: data['service_owned_by']
    }]
    services = services_model.insert(values)
```

Reasoner

- Mines workflows from the reasoner table
- API built to make it accessible from the frontend
- API call returns relevant information to the Front End

Mining and Running a Workflow

Mining a workflow

```
services=# select goal_id, goal_name, goal_description, owned_by from goal_metadata where owned_by='TWT';
 goal_id |   goal_name   |                                              goal_description                                               | owned_by
---------+---------------+-------------------------------------------------------------------------------------------------------------+----------
      16 | twirole       | json file with geolocation field & hashtags field & username field & mentions field & twirole field        | TWT
      17 | filtered      | filtered fields                                                                                             | TWT
      18 | elasticsearch |                                                                                                             | TWT
       7 | rawwarc       | Raw data                                                                                                    | TWT
       8 | rawjson       | JSON File                                                                                                   | TWT
       9 | id            | json file with geolocation field & hashtags field & username field & mentions field & twirole field & id field | TWT
      10 | username      | json tweets file with geolocation field & hashtags field & username field                                  | TWT
      11 | timestamp     | json file with geolocation field & hashtags field & username field & mentions field & twirole field & id field & timestamp field | TWT
      12 | hashtag       | json tweets file with hashtags field                                                                        | TWT
      13 | mentions      | json tweets file with geolocation field & hashtags field & username field & mentions fieldtweets           | TWT
      14 | geolocation   | json tweets file with geolocation field                                                                     | TWT
      15 | keywords      | json tweets file with geolocation field & hashtags field & username field & mentions field & keywords field | TWT
(12 rows)
```

```
services=# select service_id, service_name, service_description from service_metadata where owned_by='TWT';
 service_id |            service_name            |                                  service_description
------------+------------------------------------+------------------------------------------------------------------------------------------
          4 | warc-to-json                       | converts incoming WARC file to Json
          5 | add-id                             | Add ID field
          6 | add-username-field                 | Extracts Username field
          7 | add-timestamp-field                | Extracts Timestamp field
          8 | add-hashtags-field                 | Extracts Hashtag field
          9 | add-mentions-field                 | Extract Mentions field
         10 | add-geolocation-field              | Extracts Geolocation field
         11 | add-keywords-field                 | Extracts Keyword field
         12 | add-twirole-classification-field   | Adds field with Twirole classification (string value is either "male"/"female"/"brand")
         13 | merge-fields                       | Merge all extracted fields
         14 | generalized-indexing-using-els     | Index for elasticsearch
(11 rows)
```

Apache Airflow

- Used to run workflows on Kubernetes
- API created for the frontend to trigger workflows
- Workflow status and service logs made available to the frontend through the API

CI/CD

- Gitlab Runner
  - KGI VM
- Virginia Tech Computer Science Container Registry
  - container.cs.vt.edu
- Virginia Tech Computer Science Gitlab
  - git.cs.vt.edu

```
image: docker:latest

services:
- name: docker:18.09.7-dind

variables:
    #DOCKER_HOST: tcp://docker:2378
    DOCKER_DRIVER: overlay2
    DOCKER_TLS_CERTDIR: ""


stages:
    - build
    - test


build:api:
    stage: build
    tags:
        - docker
    script:
        - echo "Building API"
        - docker build -t api:latest images/api/
```

Unit Testing

- Created unit testing framework
- Worked with content teams through testing manager
- Goal: All services have unit tests and integration tests
  - Not quite met

Challenges

- Docker rate limits
- Elasticsearch security
- Ceph and NFS storage mounting
- Gitlab CI/CD interactions
- Communication
- Airflow currently only handles I/O through files

Future Work

- Service API
  - Fix update operation
  - Connect to Front End
- Add tests
  - Specifically for Service API and others

Questions?