Final Demo

Building Cyber Infrastructure for Researchers

Mentors:

Abraham Matta and Ali Raza

Team Members:

Tian Chen, Donovan Jones, Komal Kango, Jing Song and Kristi Perreault

Project Overview

Create Infrastructure for Researchers from the Earth Science Department at BU that allows them to run submitted code on large data sets and display the results.

Targets end-users in the Earth and Science Department

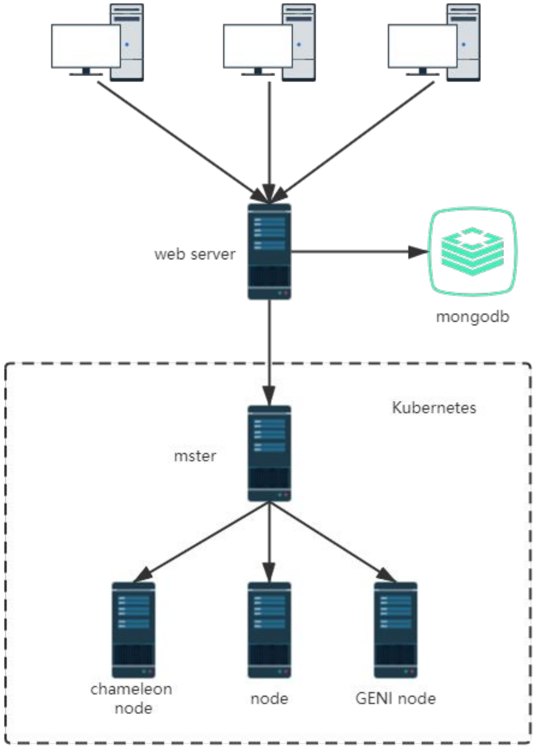
Users do not include:

- Non- ecological Researchers
- Advanced users with complex requirements beyond the scope of the project.

Vision & Goals

- Provide a web service with a simple user experience such that researchers can submit code.
- Develop reliable infrastructure on unreliable nodes using a Kubernetes Cluster
- Focus on Function as a Service with OpenWhisk as a proof of concept
- Provide a user interface that allows for comparisons between multiple models on the same data set along with comparisons of models using periodic data sets in order to determine model accuracy.

System Diagram



Compute with OpenWhisk

- Using Helm to enable Kubernetes cluster with OpenWhisk
- OpenWhisk API exposed externally with nginx
- Working with Chameleon/GENI to add/remove worker nodes on a Kubernetes cluster
- Complex Request/Response with OpenWhisk on a cluster
- Working with Plotly & MongoDB for displaying results

Orchestration with Kubernetes

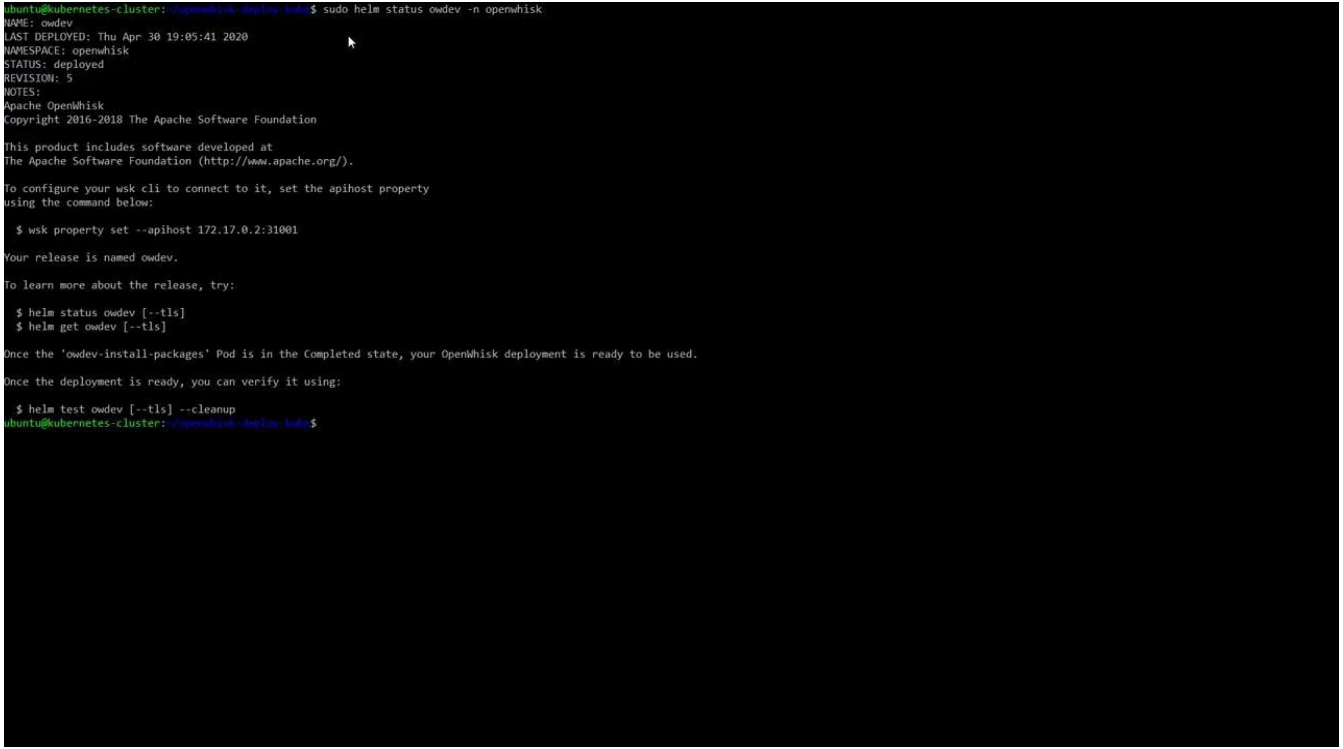
- Allows for a consistent layer over which anything can be deployed
- Focus is Function as a Service (FaaS) with OpenWhisk
- Install OpenWhisk on the cluster using Helm
- Deploy Kubernetes Cluster on the Mass Open Cloud

Unreliable Nodes

- Utilize virtual machine in the MOC along with Chameleon and GENI as the unreliable nodes to build reliable infrastructure
- Capability for infrastructure to "loan out" these nodes to services or applications as needed*
- Monitor the availability of these nodes, including up/down time and proximity to data stores*

(*) Indicates stretch goal

OpenWhisk & Kubernetes Demo



```
ubuntu@kubernetes-cluster:~/openwhisk-deploy-kube$ sudo helm status owdev -n openwhisk
NAME: owdev
LAST DEPLOYED: Thu Apr 30 19:05:41 2020
NAMESPACE: openwhisk
STATUS: deployed
REVISION: 5
NOTES:
Apache OpenWhisk
Copyright 2016-2018 The Apache Software Foundation

This product includes software developed at
The Apache Software Foundation (http://www.apache.org/).

To configure your wsk cli to connect to it, set the apihost property
using the command below:

  $ wsk property set --apihost 172.17.0.2:31001

Your release is named owdev.

To learn more about the release, try:

  $ helm status owdev [--tls]
  $ helm get owdev [--tls]

Once the 'owdev-install-packages' Pod is in the Completed state, your OpenWhisk deployment is ready to be used.

Once the deployment is ready, you can verify it using:

  $ helm test owdev [--tls] --cleanup
ubuntu@kubernetes-cluster:~/openwhisk-deploy-kube$
```

User Interface

- User registration & login
- User hierarchy with system admin, project leads, project members
    - System Admin: manage all projects and users
    - Project Lead: Add/remove team members
- Request to join projects
- Code submission with codemirror editor
- Results visualization with plotly
- Ability to view previous computations

Database Management

- User information stored and managed in MongoDB
- Store output of computation in MongoDB as a JSON object

ecoforecast.bu.edu

- Currently under deployment

User Interface Demo

New feature compare to the previous project

Old project:

Web server written by **python script**

Disadvantages: difficult to test and debug; different to add new functions

Our project:

Web server written by **flask**

Advantages: Easy to test and debug; easy to extent new features

New feature compare to the previous project

Old project:

Creating a thread to start OpenWhisk Command-line Interface to connect to the openwhisk

Disadvantages: Can’t get the data directly from openwhisk; Do not know whether the request was failed

---

Our project:

Using OpenWhisk API

Advantages: Easy to test and debug; get information directly from openwhisk including fail information

New feature compare to the previous project

Old project:

Deploying OpenWhisk on GENI or chameleon

Disadvantages: resource will expire and need to apply again, user can’t access to openwhisk during this time

Our project:

Deploy OpenWhisk on GENI or chameleon, GENI and chameleon node managed by kubernetes

Advantages: we can add new node to the kubernetes and not affect the current system in the meantime

What we learned this semester

- How to install and work with OpenWhisk
- Testing API calls with Postman
- Working with VMs and security on the MOC
- Standing up a Kubernetes node and cluster
  - Using Helm to install OpenWhisk
- Flask, HTML, and Python for the user interface
- Using Plotly & MongoDB for displaying results

Semester Problems & Limitations

- Access and working with the MOC was difficult
  - Security issue
- Getting up to speed with how to use OpenWhisk took a while
- Working with Helm to create Kubernetes cluster took some time
- Ran out of space on the Kubernetes cluster when trying to run OpenWhisk commands
- COVID-19 repercussions led to many issues
  - Difficult to find time to meet & work together with time zones
  - Had to adjust project deliverables
  - Constant communication
  - Mental & physical health

Semester Summary

- Created a Kubernetes cluster on MOC to facilitate Chameleon & GENI worker nodes
- Installed OpenWhisk on cluster to run functions for researchers
- Created MongoDB databases for storing user data and computation results
- Created a new, cleaner UI featuring:
  - User login & registration
  - Dashboard
  - User hierarchy (admin, project leads, project members)
  - Code submission and results

Future Work

- Results comparison with previous computations
- More plotting options to view data in different ways
- More Kubernetes clusters, worker nodes spread geographically
- System Admin UI
  - View which nodes are up or down
  - See geographic distribution of data & nodes
- Monitoring Application
  - Monitor worker nodes
    - If expiring, get new resource from Geni or Chameleon and add worker
  - Interface that resources from the cloud and adds it to system as spot instances

Thank you!