Supporting Collaborative Model and Data Service Development and Deployment with DevOps

Olaf David[1], Mazdak Arabi[1], Jack Carlson[1], Kyle Traff[1], Wes Lloyd[2], Ken Rojas[3]

H41B-1327

[1] Colorado State University, Fort Collins
[2] University of Washington, Tacoma
[3] USDA Natural Resources Conservation Service, Fort Collins

Abstract

Efficient deployment of modeling technology and data management using a Service-oriented Architectures SoA in a scientific research environment requires the efficient management of rapid development/deployment cycles. Changes of modeling solutions are frequent and must be quickly made available to the collaborating community.

DevOps provides best practices, tools, and organizational structures to optimize the transition from model service development to deployment by minimizing the operational burden and turnaround time for (model) developers. We have developed and implemented a methodology by integrating a suite of tools for application lifecycle management, version control, continuous integration, container management, and container scaling to enable model and data service developers in various institutions to collaboratively build, run, deploy, test, and scale services within minutes. Our methodology automates most of the workflow for service roll out and deployment.

Clouds provide an excellent platform for service deployment, however, they do not provide usually workflow and comprehensive resource management. Scientific computing in cloud environments typically suffer from heavy virtualization overhead due to OS replication on the VM. Leveraging experience managing scientific service deployments on Eucalyptus and Amazon we have developed a continuous workflow for service deployment and delivery using Kubernetes/Docker, Jenkins CI and Mercurial/Git.

We use the implemented workflow to develop, deploy and test micro services developed using the Cloud Services Integration Platform (CSIP). CSIP is a RESTful/JSON based service infrastructure based on the Object Modeling System, which was developed at Colorado State University providing for collaborative integration of environmental models into scalable model and data services as a micro-services platform.

CSIP/DevOps make model service improvements available in a short amount of time while engaging a wider community of model service users in testing and model evaluation while taking into account deployment scalability, redundancy and reliability, access security, and development efficiency.

Contact

Olaf David
odavid@colostate.edu
http://www.engr.colostate.edu/~odavid

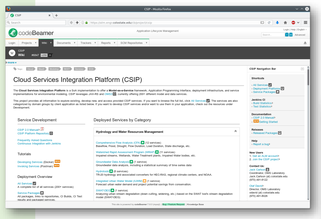
Colorado State University

Introduction

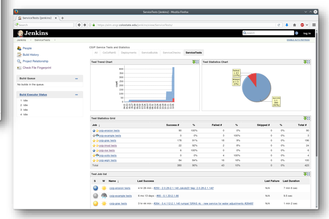
The Cloud Services Integration Platform (CSIP) developed over the last 5 years at Colorado State University provides for collaborative integration of environmental models into scalable model and data services as 'micro-services' platform with API and deployment infrastructure. CSIP, initially developed to support USDA natural resource applications, has proven to be suitable for a wide range of scientific applications spanning environmental modeling. To date, more than 160 model and data services are available for applications in hydrology (PRMS, Hydrotools, CFA, ESP), water and wind erosion prediction (WEPP, WEPS, RUSLE2), soil quality trends (SCI, STIR), water quality analysis (SWAT-CP, WQM, CFA, AgES-W), stream channel degradation assessment (SWAT-DEG), hydraulics (cross-section), and grazing management (GRAS). In addition, supporting data services include soil (SSURGO), ecological site (ESIS), climate (CLIGEN, WINDGEN), land management and crop rotations (LMOD), and pesticides (WQM), developed using this workflow automation and decentralized governance.

Adopting DevOps practices for model service development and deployment enables a community to engage in service-oriented modeling and data management. While extending its scope and visibility it became apparent community integration and adequate work flow support through the full model development and application cycle drove successful outcomes. DevOps provide best practices, tools, and organizational structures to optimize the transition from model service development to deployment by minimizing the (i) operational burden and (ii) turnaround time for modelers. We have developed and implemented a methodology to fully automate a suite of applications for application lifecycle management, version control, continuous integration, container management, and container scaling to enable model and data service developers in various institutions to collaboratively build, run, deploy, test, and scale services within minutes.

https://alm.engr.colostate.edu/cb/project/csip



The CSIP Project Application Lifecycle Management site to manage resources such as Repositories, Tracker, Service endpoint descriptions, Documentation, etc.



Jenkins setup for service build , deployment and testing. It is enabled to allow for remotely triggering builds and deployments

Objectives

Our operational workflow enables decentralized builds and deployment of services:

- Support developers to easily deploy a new service to a platform for internal testing and results evaluation.
- Track deployments in version history to capture provenance.
- Enable developers to remotely trigger service builds, deployments, and tests using a common web user interface.
- Manage service configuration changes without interrupting service availability while addressing failure through enabling roll backs to previous working versions.
- Provide multiple event based methods to trigger services deployment in response to version control actions, remote URL triggers, or direct UI triggers.
- Encapsulate management of software dependencies throughout micro services deployment to ensure all components are available for proper service operation.

Methods and Approaches

We integrated the following tools to implement this workflow:

CSIP

- The Cloud Services Integration Platform is a cross-platform Model-as-A-Service platform tailored for implementing and deploying environmental model and data services. It is a Java-based framework.

Jenkins

- Jenkins is a cross-platform, continuous integration and delivery application helps to automate the non-human part of the whole software development process, with now common things like continuous integration, but by further empowering teams to implement the technical part of a Continuous Delivery.

Codebeamer ALM

- Codebeamer is an Application Lifecycle Management Platform that integrates tools for managing resources and tracking progress of the entire software lifecycle.

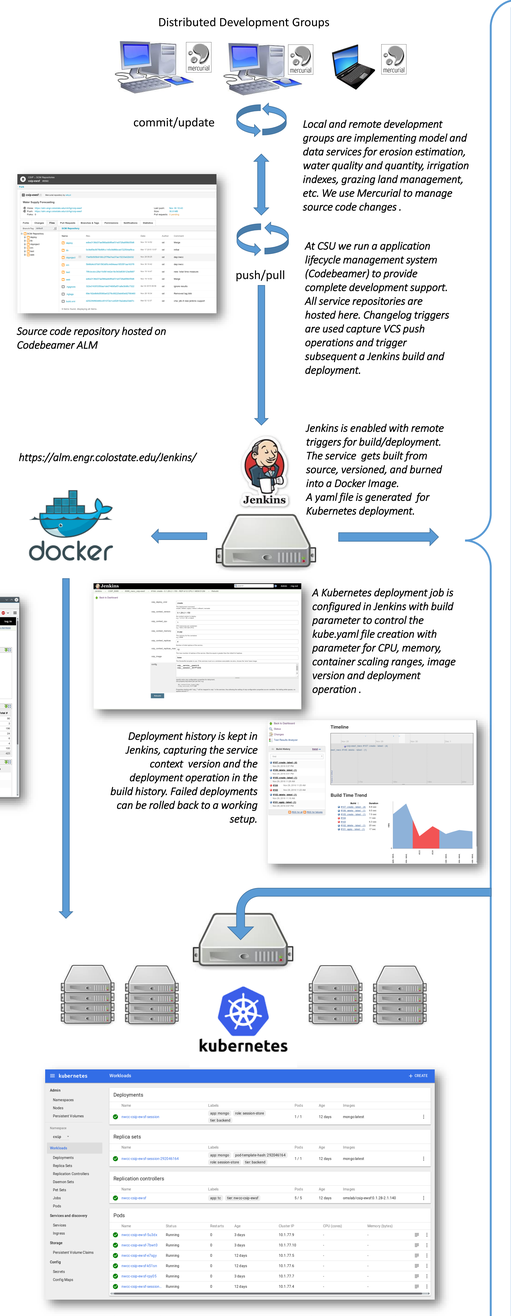
Mercurial

- Mercurial is a distributed source control management tool. It efficiently handles projects of any size and offers an easy and intuitive interface.

Kubernetes

- Kubernetes is an open source container cluster manager providing a platform for automating deployment, scaling, and operations of application containers across clusters of hosts. It integrates with Docker.

Workflow Implementation



We setup a Kubernetes cluster on 16 core DELL servers with 128 GB RAM each. We used the docker-multimode setup for master and worker nodes from 'https://github.com/kubernetes/kube-deploy'. Worker nodes can be easily added and removed. The Kubernetes dashboard allows monitoring of the overall health as well as deployment, pods, replica set, and service configuration.

The kube.yaml file (Listing 1) is auto-generated by Jenkins using a python template and environment variable injection. It populated the replica-sets, services, and horizontal auto-scaling. This file can be obtained from the Jenkins workspace.

```
apiVersion: v1
kind: Service
metadata:
  name: pf8088-nwcc-csip-ewsf-session
  namespace: csip
  labels:
    app: pf8088-nwcc-csip-ewsf-mongo
    tier: backend
spec:
  ports:
    # the port that this service should serve on
  - port: 27017
    targetPort: mongo-port
  selector:
    app: pf8088-nwcc-csip-ewsf-mongo
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: pf8088-nwcc-csip-ewsf-session
  namespace: csip
spec:
  # this replicas value is default
  # modify it according to your case
  #replicas: 1
  template:
    metadata:
      labels:
        app: pf8088-nwcc-csip-ewsf-mongo
    spec:
      containers:
      - name: session-store
        image: mongo:3.2  # or just image: mongo
        resources:
          requests:
            cpu: 250m
            memory: 256M
        ports:
        - name: mongo-port
          containerPort: 27017
---
apiVersion: v1
kind: Service
metadata:
  name: pf8088-nwcc-csip-ewsf
  namespace: csip
  labels:
    app: pf8088-nwcc-csip-ewsf
    tier: frontend
    context: csip-ewsf
    context_version: '0.1.28-2.1.150'
    platform: nwcc
    platform_port: '8088'
spec:
  # the port that this service should serve on
  - port: 48088
    targetPort: http-port
  selector:
    app: pf8088-nwcc-csip-ewsf
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: pf8088-nwcc-csip-ewsf
  namespace: csip
spec:
template:
    metadata:
      labels:
        app: pf8088-nwcc-csip-ewsf
    spec:
      containers:
      - name: pf8088-nwcc-csip-ewsf-tomcat
        image: omslab/csip-ewsf:0.1.28-2.1.150
        livenessProbe:
          httpGet:
            path: /csip-ewsf
            port: http-port
          initialDelaySeconds: 20
          timeoutSeconds: 5
          periodSeconds: 30
        resources:
          requests:
            cpu: '1'
            memory: '512M'
        env:
        - name: GET_HOSTS_FROM
          value: dns
        - name: session_ip
          value: pf8088-nwcc-csip-ewsf-session
        - name: csip__session__mongodb__uri
          value: 'mongodb://${session_ip}:27017/csip'
        ports:
        - name: http-port
          containerPort: 8080
---
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: pf8088-nwcc-csip-ewsf-autoscaler
  namespace: csip
spec:
  scaleTargetRef:
    apiVersion: extensions/v1beta1
    kind: Deployment
    name: pf8088-nwcc-csip-ewsf
  minReplicas: 4
  maxReplicas: 12
  targetCPUUtilizationPercentage: 70
```

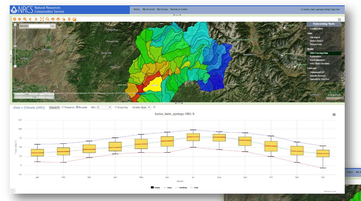

Listing 1: kube.yaml

Application Example – Detrended Kriging Service

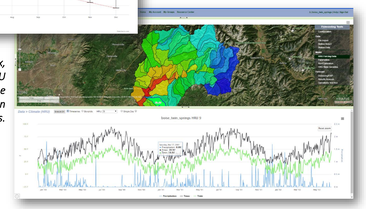
CSU is developing in cooperation with the NRCS National Water and Climate Center the next generation water supply forecasting system (eWSF). This system is being used to perform seasonal water availability forecasts to support water management decision in agriculture. This project aims to increase the functionality and efficiency of the water supply forecasting workflow deployed and operated by the NWCC. The system retrieves, processes, and de-biases climate data input for the PRMS model for twice-monthly water supply forecasts in the 600 forecast basins of the western United States. It analyzes and displays output to forecast hydrologists and provides data feeds to public facing portals and applications. It is implemented using the eRAMS geospatial platform and CSIP service infrastructure.

eWSF utilizes a *Detrended Kriging* service, which is based on the DK program (Garen et. al, 1994). The purpose of this program is to estimate spatial fields of precipitation, temperature, and snow water equivalent by interpolating among point measurements from standard surface stations. The program was written with daily time series data in mind, but it can also be used to interpolate data at other temporal resolutions, and it can be used for long-term averages.

The DK service uses climate station data/metadata and a DEM and HRU map as input and produces a climate data set with daily values for each HRU. The eWSF service package containing the DK service was automatically deployed on the Kubernetes container cluster using 64 cores using the yaml script shown in Listing 1.



eWSF GIS user interface to manage geospatial data such as HRU (Hydrological Response Unit) maps. The HRU map can be used to visualize the result of DK for computing tmin, tmax, and precip for a given HRU over a period of time using boxplots.



The generated time series for tmin, tmax, and precip can be visualized at the HRU level. The geospatial map and the time series plots are connected to a allow an interactive exploration of DK results.

Results

We implemented the presented workflow at CSU and using it for multiple projects.

- The time from committing a source code change to have a service available in multiple container replica set is less than **10 seconds** on average
- The version control system and CI are tracking a deployment history, either triggered by VCS commit hooks or CI builds.
- Service orchestrations can be adjusted by customizing the Kubernetes template.
- Creating a topology of services and taking them down without manually managing Kubernetes 'yaml' files is efficient.
- Model service developers can individually build test and deploy at scale without central governance.

Conclusions

We have implemented more than 160 model and data services for applications in hydrology (PRMS, Hydrotools, CFA, ESP), water and wind erosion prediction (WEPP, WEPS, RUSLE2), soil quality trends (SCI, STIR), water quality analysis (SWAT-CP, WQM, CFA, AgES-W), stream degradation assessment (SWAT-DEG), hydraulics (cross-section), and grazing management (GRAS), and supporting data services (SSURGO, ESIS, CLIGEN, WINDGEN), land management and crop rotations (LMOD), and pesticides (WQM), developed using this workflow automation and decentralized governance.

Container-based deployment using Kubernetes, Mercurial, and Jenkins provides all tools to allow for a continues delivery of scientific applications. Using the implemented workflow, we were able to allow a developer driven deployment for service testing in short amount of time while ensuring scalability. In addition we can track deployment history, revisit deployment parameter by leveraging Jenkins and Kubernetes features.

The implemented development and deployment solution is actively being used in several research projects.

Resources

- CSIP project: http://alm.engr.colostate.edu/cb/project/csip
- David, O., Lloyd, W., Rojas, K., Arabi, M., Geter, F., Ascough II, J., Green, T., Leavesley, G. and J. Carlson, 2014, Model-as-a-Service (MaaS) using the Cloud Services Innovation Platform (CSIP), In: Ames, D.P., Quinn, N.W.T., Rizzoli, A.E. (Eds.), Proceedings of the 7th International Congress on Environmental Modelling and Software, June 15-19, San Diego, California, USA. ISBN: 978-88-9035-744-2
- Garen, D. C., G. L. Johnson, and C. L. Hanson (1994). Mean areal precipitation for daily hydrologic modeling in mountainous regions. Water Resources Bulletin, 30(3), 481-491.
- Kubernetes project: http://kubernetes.io
- eRAMS platform: http://erams.com