CldAws230 Final Project

DSpace Pull Request Deployment Service

Terry Brady

https://github.com/terrywbrady/info

Goal

DSpace is an open-source repository platform used by academic libraries and other institutions.

Goal

Many institutions have very limited developer support. End users within these institutions have expertise to offer to the project.

Goal

This project will build an automated system to build and deploy a test instance of DSpace using code from a specific pull request.

Goal

The system will carefully manage deployed instances in order to control costs.

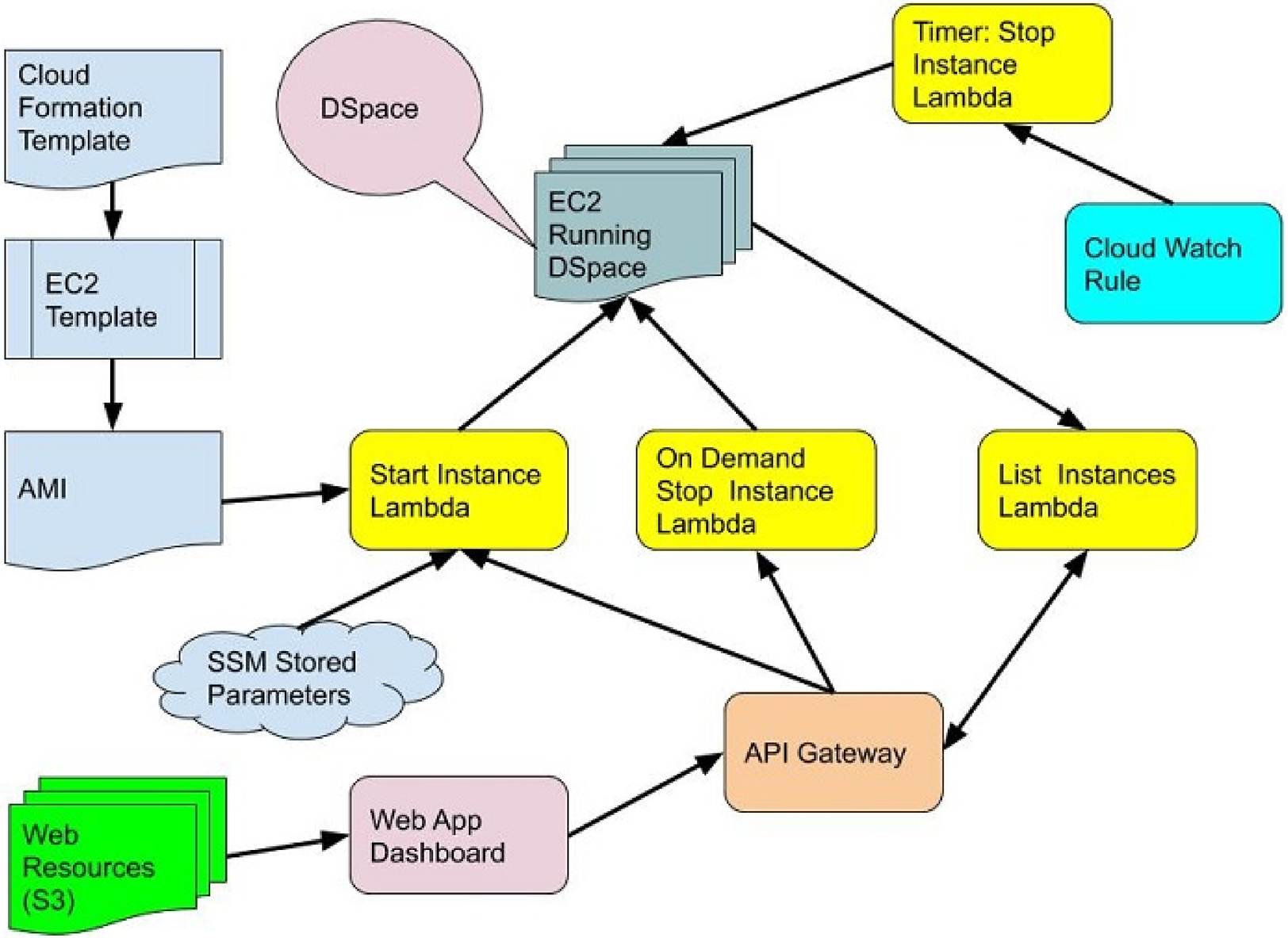
Video Playback Disabled

Services Used

- AWS Cloud Formation
- S3
- EC2
- Lambda
- API Gateway
- Cloud Watch Rule
- System Manager (SSM) Parameter Store

Non AWS Services

- GitHub API
- Docker
- Docker Compose (launching DSpace)


Cloud Formation Template
DSpace
Timer: Stop Instance Lambda
EC2 Template
EC2 Running DSpace
Cloud Watch Rule
AMI
Start Instance Lambda
On Demand Stop Instance Lambda
List Instances Lambda
SSM Stored Parameters
API Gateway
Web Resources (S3)
Web App Dashboard

Security and Cost Considerations

- Cost
- Security

Cost Management

- DSpace requires an m2.large or larger EC2 instance (SSM Param)
- The system will cap the number of instances that can be started at once (SSM Param)
- The system will set an expiration time on each instance (SSM Param)
- A Cloud Watch Rule + Lambda will terminate resources that have exceeded uptime

Security - Potential Enhancements

- Web resources
- Lambdas
- CORS Headers
- EC2 Ports

Security - Public Web Resources

- EC2 resources will be publicly accessible (for end users)
  - The dashboard will make test resources accessible
- Lambdas will be publicly accessible
  - The StartInstance lambda can trigger costs
  - Consider limiting access to this resource

Security - CORS

- CORS resources are wide open

Security - EC2 Ports

- Ports in use
  - DSpace 6: 8080, 3030
  - DSpace 7: 8080, 3030, 3000, 8983
- The ports on the running DSpace instances could be more carefully restricted

Demonstration

- Dashboard (Web App): Start, List, Stop
- DSpace: View default
- Create a DSpace PR
- Deploy PR
- DSpace: View modified version

A Quick Preview of DSpace

- Navigate to an item
- Items can contain digital media

DSPACE

Login

DSpace Home

DSpace Repository

DSpace is a digital service that collects, preserves, and distributes digital material. Repositories are important tools for preserving an organization's legacy; they facilitate digital preservation and scholarly communication.

Communities in DSpace

Select a community to browse its collections.

Dog Photos Community

Report Tools Demos

Recently Added

Regular item 5 (Added after embargo items)

Unknown author (2018)

Regular item 4 (Added after embargo items)

Search

BROWSE

- All of DSpace
- Communities & Collections
- By Issue Date
- Authors
- Titles
- Subjects

MY ACCOUNT

- Login
- Register

DISCOVER

- Author

DSpace Instance Manager Dashboard

- Select a DSpace PR
  - Launch instance
- Select a DSpace branch
  - Launch instance

DSpace Launcher Dashboard

DSpace is an open-source repository platform used by universities and other institutions. This tool is designed to help end users test pull requests for the platform. The tool will launch an instance of DSpace within Docker on an AWS instance.

Running Instances

Refresh

Instance Id	PR	Branch	Title	State	Action	DNS	End Time
i-03f187f28bb1dbce7		dspace-6_x	Branch	running	Stop	ec2-54-218-84-207 xmlui jspui	2019-05-13 20:39:20-07:00

Start Instance

; master; Branch

Start Instance

Code Repo

https://github.com/terrywbrady/Cldaws230

DSpace - Standard Home Page

DSpace Repository

DSpace is a digital service that collects, preserves, and distributes digital material. Repositories are important tools for preserving an organization's legacy; they facilitate digital preservation and scholarly communication.

Communities in DSpace

Select a community to browse its collections.

Dog Photos Community

Report Tools Demos

Create a DSpace PR

Demonstration Pull Request for Tutorial #2424

Edit

Open terrywbrady wants to merge 7 commits into `DSpace:dspace-6_x` from `terrywbrady:cldaws230`

Conversation 0 | Commits 7 | Checks 0 | Files changed 3

Changes from all commits | File filter... | Jump to... | +10 −3

Diff settings | Review changes

4 dspace-xmlui-mirage2/src/main/webapp/styles/_style.scss

```
@@ -10,3 +10,7 @@
 10  10   * You can put your own style in this file.
 11  11   * It will be included in all color schemes
 12  12   */
     13 + 
     14 + div.main-content {
     15 +     background-color: goldenrod;
     16 + }
```

Change Background Color

6 dspace-xmlui-mirage2/src/main/webapp/styles/classic_mirage_color_scheme/_utils.scss

```
@@ -35,4 +35,8 @@
 35  35 
 36  36 .justify {
 37  37     text-align: justify;
 38     - }
     38 + }
     39 + 
     40 + div.main-content {
     41 +     background-color: goldenrod;
     42 + }
```

Change Header for Home Page

3 dspace/config/news-xmlui.xml

```
@@ -2,8 +2,7 @@
 2  2   <document xmlns="http://di.tamu.edu/DRI/1.0/" xmlns:i18n="http://apache.org/cocoon/i18n/2.1" version="1.1">
 3  3   <body>
 4  4   <div id="file.news.div.news" n="news" rend="primary">
 5     - <head>DSpace Repository</head>
 6     - <p>DSpace is a digital service that collects, preserves, and distributes digital material. Repositories are important tools for p
    5  + <head>Hi CLDAWS 230!!!</head>
 7  6   </div>
 8  7   </body>
 9  8   <options/>
```

DSpace: Demonstrate PR Changes

DSPACE

Login

DSpace Home

Hi CLDAWS 230!!!

Communities in DSpace

Select a community to browse its collections.

Dog Photos Community

Report Tools Demos

Search

BROWSE

- All of DSpace
- Communities & Collections
- By Issue Date
- Authors
- Titles
- Subjects

MY ACCOUNT

- Login
- Register

RSS FEEDS

- RSS 1.0
- RSS 2.0
- Atom

Theme by ATMIRE

Choose Instance from dashboard

- Choose instance without PR changes
- Choose instance with PR changes

DSpace Launcher Dashboard

DSpace is an open-source repository platform used by universities and other institutions. This tool is designed to help end users test pull requests for the platform. The tool will launch an instance of DSpace within Docker on an AWS instance.

Running Instances Refresh

Instance Id	PR	Branch	Title	State	Action	DNS	End Time
i-0beb9363ab4cd3b09	2424	dspace-6_x	zzz	running	Stop	ec2-52-40-138-200 xmlui jspui	2019-05-13 21:22:47-07:00
i-02034e15b49b0002c		dspace-6_x	Branch	running	Stop	ec2-34-221-28-83 xmlui jspui	2019-05-13 21:32:08-07:00

Start Instance

; master; Branch

Start Instance

Code Repo

https://github.com/terrywbrady/Cldaws230

Further Enhancements

- Functionality
- AWS Operations

Additional Functionality

- Make EC2/Docker startup logs accessible
  - Flask on server OR use Kinesis
- Post incremental status updates during startup process
  - Use tag api
- Explore ECS or EKS instead of EC2 for deployment
- Explore code pipeline to build and publish PR images

AWS Operations - Functionality

- Make logs accessible
- Security enhancments (see earlier slide)
- Regional deployment

AWS Operations - Portability to another account

- Generate AMI with CloudFormation
- Create Lambda security group with Cloud Formation
- Deploy Lambda and API Gateway with Cloud Formation
- Deploy S3 website with Cloud Formation

References

Time will not likely permit this level of detail.

- Cloud Formation
- Bootstrap Script
- Lambda Permissions
- Deployment Process
- Lambda Instances
- Lambda GitHub API
- Webapp JavaSciprt

Cloud Formation to Create AMI

```
{
  "Parameters" : {
    "UserDataUrlParameter" : {
      "Type" : "String",
      "Default" : "https://raw.githubusercontent.com/terrywbrady/CldAws23(
      "Description" : "Startup script URL."
    },
    "InstanceNameParameter" : {
      "Type" : "String",
      "Default" : "ec2-dspace",
      "Description" : "Generated Instance Name."
    },
    "InstanceTypeParameter" : {
      "Type" : "String",
      "Default" : "t2.large",
      "Description" : "Instance Type."
    },
    "KeyName": {
```

```
{
  "Parameters" : {
    "UserDataUrlParameter" : {
      "Type" : "String",
      "Default" : "https://raw.githubusercontent.com/terrywbrady/CldAws23(
      "Description" : "Startup script URL."
    },
    "InstanceNameParameter" : {
      "Type" : "String",
      "Default" : "ec2-dspace",
      "Description" : "Generated Instance Name."
    },
    "InstanceTypeParameter" : {
      "Type" : "String",
      "Default" : "t2.large",
      "Description" : "Instance Type."
    },
    "KeyName": {
```

URL Parameter for startup script

```
    "Description" : "Startup script URL."
  },
  "InstanceNameParameter" : {
    "Type" : "String",
    "Default" : "ec2-dspace",
    "Description" : "Generated Instance Name."
  },
  "InstanceTypeParameter" : {
    "Type" : "String",
    "Default" : "t2.large",
    "Description" : "Instance Type."
  },
  "KeyName": {
    "Description": "Name of an existing EC2 KeyPair to enable SSH access
    "Type": "AWS::EC2::KeyPair::KeyName",
    "ConstraintDescription" : "must be the name of an existing EC2 KeyPa
  }
},
```

Other Parameters

```
  }
},
"Resources": {

  "Instance": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
      "ImageId": "ami-01e24be29428c15b2",
      "InstanceType": { "Ref" : "InstanceTypeParameter" },
      "KeyName": {
        "Ref": "KeyName"
      },
      "UserData": {
        "Fn::Base64": {
          "Fn::Join": [
            "\n",
            [
              "#!/bin/bash",
```

Image Type and Key Name Parameter Ref

```
"UserData": {
  "Fn::Base64": {
    "Fn::Join": [
      "\n",
      [
        "#!/bin/bash",
        {
          "Fn::Join": [
            " ",
            [
              "curl -o /tmp/startup.sh",
              { "Ref" : "UserDataUrlParameter" }
            ]
          ]
        },
        "chmod 744 /tmp/startup.sh",
        "/tmp/startup.sh"
      ]
    ]
```

Inject startup script URL into UserData

```
        },
        "chmod 744 /tmp/startup.sh",
        "/tmp/startup.sh"
      ]
    ]
  }
},
"Tags": [
  {
    "Key": "Name",
    "Value": { "Ref" : "InstanceNameParameter" }
  }
],
"NetworkInterfaces": [
  {
    "AssociatePublicIpAddress": true,
    "DeviceIndex": 0
  }
```

Add Name Paremeter to Tags

Bootstrap Script for EC2 that will become an AMI

```
# Update OS software
sudo -n yum -y update

# Install Java 8
sudo -n yum -y install java-1.8.0-openjdk-devel
sudo -n yum -y remove java-1.7.0-openjdk

# Install Git
sudo -n yum -y install git

# install docker
sudo yum install docker -y

# Start the Docker Service
sudo service docker start

# Add the ec2-user to the docker group so you can execute Docker commands
sudo usermod -a -G docker ec2-user
```

```
# Update OS software
sudo -n yum -y update

# Install Java 8
sudo -n yum -y install java-1.8.0-openjdk-devel
sudo -n yum -y remove java-1.7.0-openjdk

# Install Git
sudo -n yum -y install git

# install docker
sudo yum install docker -y

# Start the Docker Service
sudo service docker start

# Add the ec2-user to the docker group so you can execute Docker commands
sudo usermod -a -G docker ec2-user
```

Install Java

```
# Update OS software
sudo -n yum -y update

# Install Java 8
sudo -n yum -y install java-1.8.0-openjdk-devel
sudo -n yum -y remove java-1.7.0-openjdk

# Install Git
sudo -n yum -y install git

# install docker
sudo yum install docker -y

# Start the Docker Service
sudo service docker start

# Add the ec2-user to the docker group so you can execute Docker commands
sudo usermod -a -G docker ec2-user
```

Install Git

```
sudo -n yum -y remove java-1.7.0-openjdk

# Install Git
sudo -n yum -y install git

# install docker
sudo yum install docker -y

# Start the Docker Service
sudo service docker start

# Add the ec2-user to the docker group so you can execute Docker commands
sudo usermod -a -G docker ec2-user

# Install Docker compose
# https://docs.docker.com/compose/install/

sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/
```

Install Docker

```
# install docker
sudo yum install docker -y

# Start the Docker Service
sudo service docker start

# Add the ec2-user to the docker group so you can execute Docker commands
sudo usermod -a -G docker ec2-user

# Install Docker compose
# https://docs.docker.com/compose/install/

sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/(
sudo chmod 755 /usr/local/bin/docker-compose


# Clone DSpace-Docker-Images
```

Install Docker Compose

```
# install docker
sudo yum install docker -y

# Start the Docker Service
sudo service docker start

# Add the ec2-user to the docker group so you can execute Docker commands
sudo usermod -a -G docker ec2-user

# Install Docker compose
# https://docs.docker.com/compose/install/

sudo curl -L "https://github.com/docker/compose/releases/download/1.23.2/(
sudo chmod 755 /usr/local/bin/docker-compose


# Clone DSpace-Docker-Images
```

Pre-load Docker Images

Lambda Permissions

```
[
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
```

```
        "Resource": "arn:aws:logs:*:*:*"
    },
    {
        "Sid": "VisualEditor1",
        "Effect": "Allow",
        "Action": [
            "ec2:Describe*",
            "ec2:TerminateInstances",
            "ec2:Start*",
            "ec2:RunInstances",
            "ec2:CreateTags",
            "ec2:Stop*"

        ],
        "Resource": "*"
    },
    {
        "Sid": "VisualEditor2",
```

EC2 Operations Performed by Lambda

Simple Deployment Process

```
#!/bin/sh

# This script is intended to be run from AWS Cloud 9 to deploy DSpace Laur
#
# Set the following variables in Code9
# - CLDAWS_API - API Gateway Address
# - CLDAWS_BUCKET - AWS S3 Bucket for deployment

# Create a build directory to contain a zip file of lambda code modules
rm build/*
cp lambda/*.py build
cd build
zip -r cldawsLambda.zip *.py

# Make the API Gateway address accessible to the web application
cd ..
echo "var API_BASE='${CLDAWS_API}';" > build/dspaceLauncher.init.js
```

```
# Set the following variables in Code9
# - CLDAWS_API - API Gateway Address
# - CLDAWS_BUCKET - AWS S3 Bucket for deployment

# Create a build directory to contain a zip file of lambda code modules
rm build/*
cp lambda/*.py build
cd build
zip -r cldawsLambda.zip *.py

# Make the API Gateway address accessible to the web application
cd ..
echo "var API_BASE='${CLDAWS_API}';" > build/dspaceLauncher.init.js

aws s3 cp --acl public-read build/dspaceLauncher.init.js s3://${CLDAWS_BU

aws s3 cp --acl public-read web/index.html         s3://${CLDAWS_BUCKET}
aws s3 cp --acl public-read web/dspaceLauncher.js  s3://${CLDAWS_BUCKET}
```

Prep Resources

```
cd build
zip -r cldawsLambda.zip *.py

# Make the API Gateway address accessible to the web application
cd ..
echo "var API_BASE='${CLDAWS_API}';" > build/dspaceLauncher.init.js

aws s3 cp --acl public-read build/dspaceLauncher.init.js s3://${CLDAWS_BU

aws s3 cp --acl public-read web/index.html         s3://${CLDAWS_BUCKET}
aws s3 cp --acl public-read web/dspaceLauncher.js  s3://${CLDAWS_BUCKET}
aws s3 cp --acl public-read web/dspaceLauncher.css s3://${CLDAWS_BUCKET}


# Install the code for each Lambda
aws lambda update-function-code --function-name projTestZip       --zip-
aws lambda update-function-code --function-name projGetPRs        --zip-
aws lambda update-function-code --function-name projListInstances  --zip-
aws lambda update-function-code --function-name projStopInstance   --zip-
```

Deploy to S3

```
# Make the API Gateway address accessible to the web application
cd ..
echo "var API_BASE='${CLDAWS_API}';" > build/dspaceLauncher.init.js

aws s3 cp --acl public-read build/dspaceLauncher.init.js s3://${CLDAWS_BU

aws s3 cp --acl public-read web/index.html         s3://${CLDAWS_BUCKET}
aws s3 cp --acl public-read web/dspaceLauncher.js  s3://${CLDAWS_BUCKET}
aws s3 cp --acl public-read web/dspaceLauncher.css s3://${CLDAWS_BUCKET}


# Install the code for each Lambda
aws lambda update-function-code --function-name projTestZip        --zip-
aws lambda update-function-code --function-name projGetPRs         --zip-
aws lambda update-function-code --function-name projListInstances  --zip-
aws lambda update-function-code --function-name projStopInstance   --zip-
aws lambda update-function-code --function-name projCreateInstance --zip-
```

Deploy to Lambda

Lambda Instances

```
# Source: https://github.com/terrywbrady/CldAws230/blob/project/lambda/get
#
# This code is used by the DSpace Launcher Dashboard Application.  This co
# - projListInstances  - list EC2 instances started by the DSpace Launcher
# - projStopInstance   - manually stop an EC2 instance started by the DSpa
# - projCreateInstance - start a new EC2 instance that will run DSpace ust
# - projTimer          - kill a DSpace EC2 instance that has exceeded its

import boto3
import json
import sys
import base64
import dateutil.tz
import datetime

# ==================================================
# AWS System Manager Parameter Store
#
```

```python
import dateutil.tz
import datetime

# ==========================================================
# AWS System Manager Parameter Store
#
# The parameter store is designed to throttle the costs associated with th
# resources launched from the dashboard without needing to redeploy code.
# ==========================================================
ssm = boto3.client('ssm', region_name="us-west-2")
def getSSMParam(key, value):

    p = ssm.get_parameter(Name=key)
    return p['Parameter']['Value'] if p else value

def getSSMIntParam(key, value):
    return int(getSSMParam(key, value))

# ==========================================================
```

Read SSM Stored Parameters

```
# ==========================================================
# Constants
# ==========================================================

DSPACE_TAG_NAME  = "DSpace"
DSPACE_TAG_VALUE = "DSpace"
MAX_INSTANCE     = getSSMIntParam("DSPACE_DASHBOARD.MAX_INSTANCES", 2)
REGION           = 'us-west-2'
TZONE            = dateutil.tz.gettz('US/Pacific')
UPTIME           = getSSMParam("DSPACE_DASHBOARD.UPTIME", "60")
INSTTYPE         = getSSMParam("DSPACE_DASHBOARD.INSTANCE_TYPE", "t2.xlarg
AMI              = getSSMParam("DSPACE_DASHBOARD.AMI", "ami-01861f3408641
KEYNAME          = getSSMParam("DSPACE_DASHBOARD.KEYNAME", "week8key")

# ==========================================================
# Get Instances started from the DSpace Launcher dashboard
# ==========================================================
```

Read SSM Stored Parameters

```python
# ==================================================
# Get Instances started from the DSpace Launcher dashboard
# ==================================================

# Lambda invoked from web form via API gateway
# ----------------------------------------------
def lambda_getInstances(event, context):
    return {
        'statusCode': 200,
        'headers': { 'Access-Control-Allow-Origin': '*'},
        'body': json.dumps(getInstanceJsonObjects())
    }

# Get a list of EC2 instances launched from the dashboard.
# Return the list of instances as dashboard objects
def getInstanceObjects():
    fres = []
```

Get Instances Lambda -> ec2.describe_instances

```
return [
    {
        'Key': DSPACE_TAG_NAME,
        'Value': DSPACE_TAG_VALUE
    },
    {
        'Key':'Name',
        'Value': title
    },
    {
        'Key':'UPTIME',
        'Value': UPTIME
    },
    {
        'Key':'Branch',
        'Value': branch
    },
    {
```

Create EC2 Tags

```
# launch Docker Compose.
# See https://github.com/DSpace-Labs/DSpace-Docker-images for an explanat
# DSpace Docker Compose Options.
# If DSpace were to pre-build docker images for every active pull request,
# could be simplified.
def getUserData(pr, branch):
    ver = " -f d7.override.yml"
    if branch == "master":
      ver = " -f d7.override.yml -f load.entities.yml"
    elif branch == "preview":
      ver = " -f d7.override.yml -f d7.preview.yml -f load.entities.yml"
    elif branch == "dspace-6_x":
      ver = " -f d6.override.yml"
    elif branch == "dspace-5_x":
      ver = " -f d5.override.yml"
    elif branch == "dspace-4_x":
      ver = " -f d4.override.yml"
```

Construct EC2 UserData - docker compose

```
elif branch == "dspace-6_x":
  ver = " -f d6.override.yml"
elif branch == "dspace-5_x":
  ver = " -f d5.override.yml"
elif branch == "dspace-4_x":
  ver = " -f d4.override.yml"

commands = [
    "cd /home/ec2-user/DSpace-Docker-Images",
    "DNS=`curl -s http://169.254.169.254/latest/meta-data/public-hostn
    "echo DNS=${DNS}",
    "export BASEROOT=http://${DNS}:8080",
    "sed -i -e s/localhost/${DNS}/ add-ons/angular-tools/environment.c
]
if (pr != ""):
    commands = [
        "cd /home/ec2-user/",
        "git clone https://github.com/DSpace/DSpace.git",
```

UserData - Set Environment

```
    "DNS=`curl -s http://169.254.169.254/latest/meta-data/public-hostn
    "echo DNS=${DNS}",
    "export BASEROOT=http://${DNS}:8080",
    "sed -i -e s/localhost/${DNS}/ add-ons/angular-tools/environment.c
]
if (pr != ""):
    commands = [
        "cd /home/ec2-user/",
        "git clone https://github.com/DSpace/DSpace.git",
        "cd DSpace",
        "git checkout " + branch,
        "git pull",
        "export DSPACE_SRC=$PWD",
        "curl -o /tmp/pr.patch -L https://github.com/DSpace/DSpace/pul
        "git apply /tmp/pr.patch",
    ]

commands.append("cd /home/ec2-user/DSpace-Docker-Images")
```

UserData - Clone DSpace Code for PR

```
        "export DSPACE_SRC=$PWD",
        "curl -o /tmp/pr.patch -L https://github.com/DSpace/DSpace/pul
        "git apply /tmp/pr.patch",
    ]

commands.append("cd /home/ec2-user/DSpace-Docker-Images")
commands.append("git pull origin")
commands.append("cd docker-compose-files/dspace-compose")

if (pr != ""):
    commands.append("docker-compose -p dspace -f docker-compose.yml "
    commands.append("docker-compose -p dspace -f docker-compose.yml "
    commands.append("docker-compose -p dspace -f docker-compose.yml "
else:
    commands.append("docker-compose -p dspace -f docker-compose.yml "
    commands.append("docker-compose -p dspace -f docker-compose.yml "

return "#!/bin/bash\nsudo su -l ec2-user -c '" + ";".join(commands) +
```

UserData - Clone Docker Compose Files

```
        InstanceType=INSTTYPE,
        UserData=getUserData(pr, branch),
        KeyName=KEYNAME
    )
    ids=[]
    for instance in instances['Instances']:
        ids.append(instance['InstanceId'])
    ec2.create_tags(Resources=ids,Tags=getTags(pr, branch, title))
    return ids

# Verify that the number of DSpace instances does not exceed the maiximum
def checkRunningInstances():
    return len(getInstanceObjects()) < MAX_INSTANCE

# API Gateway interface for the lambda that starts an instance
def lambda_startInstances(event, context):
    body = json.loads(event['body'])
    pr = body['prnum']
```

Check that running instances does not exceed max allowed

```
# Start and EC2 instance that will run a specific branch/pull request for
# Set the EC2 tags to convey dashboard metadata for the instance.
# Set the userdata for the EC2 instance to bootstrap the call to docker-c
def startInstance(pr, branch, title):
    # https://boto3.amazonaws.com/v1/documentation/api/latest/reference/se
    # https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-met
    ec2 = getEC2()
    instances = ec2.run_instances(
        MaxCount=1,
        MinCount=1,
        ImageId=AMI,
        InstanceType=INSTTYPE,
        UserData=getUserData(pr, branch),
        KeyName=KEYNAME
    )
    ids=[]
    for instance in instances['Instances']:
        ids.append(instance['InstanceId'])
    ec2.create_tags(Resources=ids,Tags=getTags(pr, branch, title))
```

Start Instance Lambda -> ec2.run_instances

```python
    id = qp['id'] if 'id' in qp else ''
    ids = stopInstance(id)
    return {
        'statusCode': 200,
        'headers': { 'Access-Control-Allow-Origin': '*'},
        'body': json.dumps(ids)
    }

# Stop a specific DSpace instance
def stopInstance(id):
    objarr = getInstanceObjects()
    ids = getObjIdsByVal(objarr, id)
    if (len(ids) > 0):
        ec2 = getEC2().terminate_instances(InstanceIds=ids)
    return ids

# Lambda invoked by cron/schedule - will run every 5 min to stop instnaces
# ------------------------------------------------------------------
def lambda_stopQuotingInstances(event, context):
```

Stop Instance Lambda -> ec2.terminate_instances

```python
def lambda_stopOvertimeInstances(event, context):
    ids = stopOvertimeInstances()
    return {
        'statusCode': 200,
        'headers': { 'Access-Control-Allow-Origin': '*'},
        'body': json.dumps(ids)
    }

# Stop any DSpace instance over its allotted time
def stopOvertimeInstances():
    objarr = getInstanceObjects()
    ids = getObjIdsByDate(objarr)
    if (len(ids) > 0):
        ec2 = getEC2().terminate_instances(InstanceIds=ids)
    return ids

# ==================================================
# Common Functions
# ==================================================
```

Stop Overtime Instance Lambda -- Cloud Watch Rule

```python
# TODO: make this smarter  and more portable
def getEC2():
    return boto3.client('ec2', region_name=REGION)

# Filter EC2 Reservations instances by tags to find instances started fro
def getReservations():
    return getEC2().describe_instances(
        Filters=[
            {
                'Name': 'tag:' + DSPACE_TAG_NAME,
                'Values': [DSPACE_TAG_VALUE]
            },
            {
                'Name': 'instance-state-name',
                'Values': ['running', 'pending', 'stopping', 'shutting-do
            },
        ]
    )
```

Filter Running Instances

Lambda GitHub API

```python
## get PRs

# https://developer.github.com/v3/pulls/#list-pull-requests

#- open pulls are grabbed by default
#- should the base url be pulled
#- should branch be inferred from pull?

#https://api.github.com/repos/DSpace/DSpace/pulls

import boto3
import urllib2
import json
import re

# Grab the first 2 pages of DSpace Pull Requests using the GitHub API
def getPRs():
    prs = []
```

```
for branch in ['master', 'preview','dspace-6_x','dspace-5_x','dspace-4
    prs.append({
        'prnum': '',
        'base': branch,

        'title': 'Branch'
    })
for page in range(1, 2):
    req = urllib2.Request('https://api.github.com/repos/DSpace/DSpace/
    req.add_header('accept', 'application/json')
    response = urllib2.urlopen(req)
    jsondata = json.load(response)

    for pr in jsondata:
        match = re.match(r".*/(\d+)$", pr['url'])
        prnum = match.group(1) if match else ""
        prs.append({
            'prnum': prnum,
```

Call GitHub API

```
                'prnum': prnum,
                'title': pr['title'],
                'base': pr['base']['ref'],
            })
    return prs

# Lambda Handler wrpping getPRs()
def lambda_handler(event, context):
    prs = getPRs()
    return {
        'statusCode': 200,
        'headers': { 'Access-Control-Allow-Origin': '*'},
        'body': json.dumps(prs)
    }

# Testing function
def test_handler(event, context):
    prs = [1,2,3,4]
```

Get PRs Lambda

```
    return {
        'statusCode': 200,
        'headers': { 'Access-Control-Allow-Origin': '*'},
        'body': json.dumps(prs)
    }

# Testing function
def test_handler(event, context):
    prs = [1,2,3,4]
    return {
        'statusCode': 200,
        'headers': { 'Access-Control-Allow-Origin': '*'},
        'body': json.dumps(prs)
    }

# CLI interface for testing the lambda code
prs=getPRs()
for pr in getPRs():
```

CLI Tester

Webapp JavaSciprt

```javascript
// API_BASE will be set in dspaceLauncher.init.js
var PRS;

/*
Refresh the DSpace dashboard
*/
$(document).ready(function(){
  $("#refresh").on("click", function(){refresh();});
  refresh();
  loadPRs();
  $("#startInstance")
    .on("click", function(){
      startInstance();
    });
});

/*
Get the list of running DSpace instances and present as a table
```

```
var PRs;

/*
Refresh the DSpace dashboard
*/
$(document).ready(function(){
  $("#refresh").on("click", function(){refresh();});
  refresh();
  loadPRs();
  $("#startInstance")
    .on("click", function(){
      startInstance();
    });
});

/*
Get the list of running DSpace instances and present as a table
*/
function refresh() {
```

Load Resources

```
    });
});

/*
Get the list of running DSpace instances and present as a table
*/
function refresh() {
  $.getJSON(API_BASE+"/projListInstances", function(data){
    if (data == null) return;
    $("#instances tr.data").remove();
    for(var i=0; i<data.length; i++) {

      var tr = $("<tr class='data'/>");
      $("#instances table").append(tr);
      var obj = data[i];
      var tddns = $("<td/>").text(obj['dns'].replace(/\..*$/,""))
      for(var j=0; j<data[i].services.length; j++){
        var sv = data[i].services[j];
        var url = "http://"+obj['dns']+sv.path;
```

Call Get Instances Lambda

```
    var sv = data[i].services[j];
    var url = "http://"+obj['dns']+sv.path;
    tddns.append($("<span> </span>"));
    tddns.append($("<a/>").text(sv.name).attr("href", url));
  }
  var tdact = $("<td/>")
    .append(
      $("<button class='stop'/>")
        .text("Stop")
        .attr("id", obj['id'])
        .on("click", function(){
          $(this).attr("disabled", true);
          stopInstance($(this).attr("id"))
        })
    );

  tr.append($("<td/>").text(obj['id']))
    .append($("<td/>").text(obj['pr']))
```

Create stop button

```
        .on("click", function(){
          $(this).attr("disabled", true);
          stopInstance($(this).attr("id"))
        })
    );

  tr.append($("<td/>").text(obj['id']))
    .append($("<td/>").text(obj['pr']))
    .append($("<td/>").text(obj['branch']))
    .append($("<td/>").text(obj['name']))
    .append($("<td/>").text(obj['state']))
    .append(tdact)
    .append(tddns)
    .append($("<td/>").text(obj['endTime']));
}

/*
TODO - the maximum number of instances should be pulled from AWS SSM
rather than hard-coding
```

Show table

```javascript
/*
Call the Get PRs lambda and update a select widget with the values
*/
function loadPRs() {
  $.getJSON(API_BASE+"/projgetprs", function(data){
    if (data == null) return;
    PRS=data;
    $("#pr option").remove();
    for(var i=0; i<data.length; i++) {
      var row=data[i];
      var str = row.prnum + "; " + row.base + "; " + row.title;
      var opt = $("<option/>")
        .attr("value", i)
        .text(str);
      $("#pr").append(opt);
    }
  });
}
```

Call Get PRs Lambda

```
      $("#pr").append(opt);
    }
  });
}

/*
Call stop instance lambda
*/
function stopInstance(id) {
  $.getJSON(API_BASE+"/projstopinstances?id="+id, function(data){
    setTimeout(refresh(), 2000);
  });
}

/*
Call start instance lambdaPerms
*/
function startInstance(){
```

Call Stop Instance Lambda

```
function startInstance(){
  $("#startInstance").attr("disabled", true);
  var data = {}
  var i = $("#pr").val();
  if (i>=0 && i<PRS.length) {
    data = PRS[i];
  }
  $.ajax({
    type: "POST",
    url: API_BASE+"/projcreateinstance",
    data: JSON.stringify(data),
    dataType: "json",
    contentType: 'application/json',
    success: function(){
      setTimeout(function(){refresh()}, 2000);
    },
    failure: function() {
      alert("Instance Start Failed");
```

Call Start Instance Lambda

Thank You

- https://github.com/terrywbrady/Cldaws230
- https://github.com/terrywbrady/info