

E91 Cloud DevOps Final Project

Group 1

Names & assigned task(s):

Item	Task(s)	Contributor Name
1	AWS setup, Dev & Stage, Logging/Monitoring Developer <ul style="list-style-type: none">Set up the AWS infrastructure and AWS client instances, create CFTSet up Dev and Staging container shell scripts and CFTs for Jenkins to usework with adjacent team members to integrate CFT creation/destruction of containers into Jenkins for dev and stage and push to ProdWork on Logging/monitoring for AWSDocumentation Editor	Christine Liu
2	Jenkins Developer <ul style="list-style-type: none">Define AWS Jenkins setup, and set up pipelines.Work with adjacent team members to set up webhooks.	Charles Mateer
3	Testing Developer <ul style="list-style-type: none">Create test scripts and tools potential on Jenkinsimplement Python HTML Parser	Jingjing Zheng
4	Production Instance (GCP Instance setup, Docker container, webhook server) <ul style="list-style-type: none">Prod Instance<ul style="list-style-type: none">Instance setupContainer & Site setupGithub webhook server setup	Stephen Akaeze
5	GCP setup, Ansible Server, Docker file, Logging/Monitoring Developer <ul style="list-style-type: none">Set up GCP environment and templatescreate and manage Ansible server on GCP with Ansible playbooksCreate Dockerfile to be used for Dev, stage containers.Work on logging/monitoring for GCPwork with adjacent team members to integrate CFT creation/destruction of containers for dev and stage into Jenkins and push to Prod	Philip Cunningham
6	Front end Developer <ul style="list-style-type: none">Create full scale website (HTML, CSS, JS) for public GIT repo to demonstrate real life CI/CD workloads.	Elias Falconi

Repos URLs:

- Public Repo: <https://github.com/cwmat/csci-91-final-phase1-public>
- Private Repo: <https://code.harvard.edu/cwm850/csci-91-final-phase1-private>

Submission Date: 12/19/18



Table of Content

PROJECT OBJECTIVE	3
PROJECT APPROACH	4
PLANNING	4
Roles	4
Architecture	5
Project Set Up Plan	6
Meeting frequency and goals	6
IMPLEMENTATION	6
Dockerfile - Philip Cunningham	7
AWS Infrastructure Setup - Christine Liu	7
AWS Base Environment CFT template	7
Dev EC2 CFT template	9
Staging EC2 CFT template	9
GCP Infrastructure Setup - Philip Cunningham and Stephen Akaeze	10
Ansible Server instance - Philip Cunningham	10
Prod Instance - Stephen Akaeze	14
Prod Server Instance, Container Setup and Webhook Configuration - Stephen Akaeze	15
Purpose	15
Implementation	15
Jenkins Pipeline and WebHooks - Charles Mateer	17
Initial Setup	17
Install Plugins and Configure Git Webhooks	19
Setup Pipeline	21
Freestyle Job Configurations	23
Test Cases - Jingjing Zheng	26
The site is up	26
The content is what you expected	26
Website - Elias Falconi	26
Background	27
Initial setup	27
Responsiveness	30
JavaScript	32
Miscellaneous info	33
Logging/Monitoring - Christine Liu & Philip Cunningham	34



Purpose	34
Logging/Monitoring in AWS	34
Logging/Monitoring in GCP	35
Checking Logs	35
FOLDER DESCRIPTIONS	37
Private Github	37
Public Github	37
FINAL PROJECT FUNCTIONALITY	38
INITIAL WEBSITE	38
Link to the website	38
Image of the initial website	38
OUTLINE OF CHANGES	39
Changing Files	39
Jenkins CI/CD process	41
RESULTS	43
Change results reflected in Github	43
Change results reflected on site	44
DOCUMENTATION FOR USER	45
E91 CLOUD DEVOPS FINAL PROJECT PHASE 1	45
MOTIVATION	45
HOW IT WORKS	45
CONTRIBUTING GUIDELINES	45
AUTHORS	46

PROJECT OBJECTIVE

Using DevOps techniques learned in class, group 1 will develop and maintain an open source website project that accounts for minimal/no funding and security in a single container. We will use CI/CD techniques to build a pipeline for a website that needs a seamless, end-to-end continuous delivery and deployment workflow. A continuous integration pipeline will support test, configuration, and deployment of a website and a continuous delivery pipeline will support automatic building, testing, and release of code changes to production environment.



PROJECT APPROACH

In the project approach section, group 1 will explain how we collaborated. We document our planning and implementation processes and results.

PLANNING

To ensure that DevOps was used properly and efficiently, we had several planning sessions to discuss the CI/CD requirements, architecture, roles, infrastructure build-out plan, and how to approach development, code reviews, and version control.

Roles

As a group, we brainstormed and decided the best way to organize and divide the roles to ensure efficient project buildout. Although we had roles, the group was in constant communication to ensure timely project development. All group members agreed and contributed to overall solution and troubleshooting promptly.

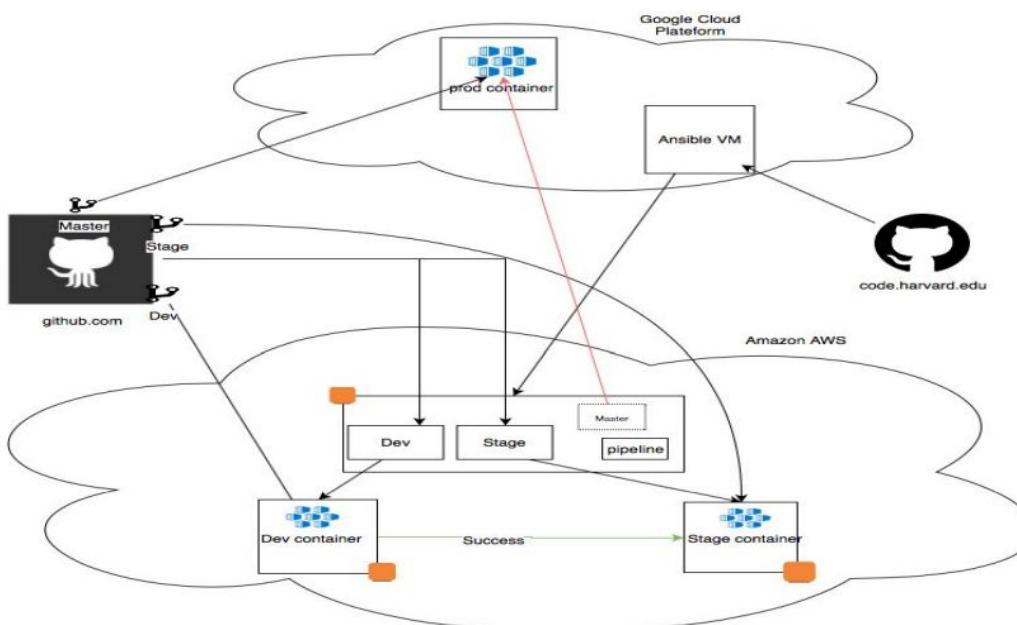
- **Testing Developer (Jingjing Zheng):**
 - Create test scripts and tools potential on jenkins
 - implement Python HTML Parser
- **Front end Developer (Elias Falconi):**
 - Create full scale website (HTML, CSS, JS) for public GIT repo that demonstrates a real CI/CD workload
- **GCP setup, Ansible Server, Docker file, Logging/Monitoring Developer (Philip Cunningham):**
 - Set up GCP environment
 - create and manage ansible server on GCP with ansible playbook
 - Create Dockerfile to be used for Dev, stage containers.
 - Work on logging/monitoring for GCP
 - work with adjacent team members to integrate Jenkins triggered/destroyed CFT templates for Dev and Staging and push to Prod
- **Production Instance (Stephen Akaeze):**
 - **Prod Instance**
 - Instance setup
 - Public repo & Container & Site setup
 - Github webhook server setup
- **Jenkins Developer (Charles Mateer):**
 - Define AWS Jenkins setup, and set up pipelines.
 - Work with adjacent team members to set up webhooks.
- **AWS setup, Dev & Stage, Logging/Monitoring Developer (Christine Liu):**
 - Set up the AWS infrastructure for AWS client instances, create CFT



- Set up Dev and Staging containers with CFT templates and shell scripts
- work with adjacent team members to integrate Jenkins triggered/destroyed CFT templates for Dev and Staging and push to Prod
- Work on Logging/monitoring for AWS
- Documentation Editor

Architecture

This was the architecture explained in Week 13 lecture to be used for the final project. We will be using CI/CD techniques to deploy software rapidly, repeatedly, and reliably. Testing and Staging will be hosted in AWS, while production will be hosted in GCP. The Ansible VM will set up the Jenkins. There will be a private repo to store our infrastructure templates (cloud formation and GCP templates) and any other sensitive data files, keeping confidential user data private. There will also be a public repo to store website code, test cases, and code that do not pose a security concern that other developers can use and improve. When a developer commits a change to the public repo dev branch, Jenkins will trigger the creation of a dev container and run through tests. If the tests pass, Jenkins will spin up a stage container and destroy the dev container. After running through staging tests, and if the tests pass, Jenkins will destroy the stage container and the changes will be committed to staging/master public repo and reflected in production. This architecture ensures automated testing and deployment of code that addresses the funding concern for dev and stage EC2 containers by only creating these test VMs when a change has been committed and tearing them down after.





Project Set Up Plan

To keep the project organized and efficient, we decided whose accounts we would use and who would set up the code repository and SSH key for us to use.

Accounts: As a team, we decided whose accounts to use in order to complete final project

- AWS account and roles - Charles Mateer
- GCP account and roles - Philip Cunningham

Code repositories and SSH key: Charles Mateer was responsible for setting up the code repos that the team will be using and generating the final project SSH key.

- Public Repo: <https://github.com/cwmat/csci-91-final-phase1-public>
- Private Repo: <https://code.harvard.edu/cwm850/csci-91-final-phase1-private>

*Infrastructure Development planning and implementation documented in Implementation Section

Meeting frequency and goals

In order to complete this project, we performed as a DevOps team.

- **Group Chat:** A group chat was set up so that constant, immediate communication was possible.
- **Scrum Calls:** Scrum calls were set up every other day to go through progress, raise concerns and dependencies, solve issues, and troubleshoot.
- **Code reviews/Version Control:** As a group, during our scrum calls, we would discuss our updates and ask for verbal approval of code changes. In addition, we learned how to implement git pull requests.
- **Pair programming:** We also practiced pair programming, where we would work with another person, checking before committing code.
- **Live Working sessions:** In addition to pair programming, we also held a lot of teamwide working sessions for holistic system development, integration, and testing via video conferencing.

IMPLEMENTATION

In this section, we detail how we implemented the architecture for this open source project. Each group member documented their assigned part of the project throughout this section. As described in the architecture section, our infrastructure spanned across AWS and GCP. Jenkins, Stage, and Dev are in AWS and prod and Ansible server are in GCP. When a developer pushes a commit to the dev branch of the public repo, the change will automatically trigger Jenkins to create a dev environment and run through some tests. If the tests pass dev, the changes will move onto staging and Jenkins will trigger creation of a staging environment, initiate some tests, and the destruction of the dev environment. If the tests pass staging, the

staging environment will terminate and the changes will be reflected in public repo staging branch, the master branch of public repo, and prod.

Dockerfile - Philip Cunningham

The Dockerfile is used in the shell scripts used to install docker containers on the VMs. The Dockerfile was tailored and configured to work with the container setup script to be used in all infrastructure setup. There are two docker files for the project, one based on the file from class and a second docker file for ubuntu based OS that creates an apache2 server.

```
1 # 1) For ubuntu GCP prod
2 FROM      ubuntu:latest
3
4
5 # 2)
6 RUN       apt-get update && apt-get install -y sudo && rm -rf /var/lib/apt/lists/* && sudo apt-get update -y && sudo apt-get install apache2
7
8
9 # 3)
10 EXPOSE   80
11
12
13 # 4)
14 CMD      ["/usr/sbin/apache2ctl","-D","FOREGROUND"]
15
16
17 # 5) may need to copy multiple files and dir
18 COPY      index.html /var/www/html/
19 COPY      csci-91-final-phase1-public/* /var/www/html/
```

The docker file installs apache2, exposes port 80, runs on the container and copies the open source public project and copies the website files and folder structure to the container.

AWS Infrastructure Setup - Christine Liu

As discussed and agreed within group 1, we decided to have a total of 3 cloud formation templates in the private repo for the AWS infrastructure. I contributed to mostly the private repo.

AWS Base Environment CFT template

The first CFT called 0-deploy-vpc-subnet-sg-jenkins.yaml contains the code to setup the VPC, Subnet, security group, and Jenkins EC2

(<https://code.harvard.edu/cwm850/csci-91-final-phase1-private/blob/master/cloud-formation/0-deploy-vpc-subnet-sg-jenkins.yaml>). It is located in the private repo.

- **VPC:** Group1-finalproject



VPC: vpc-05cdc2e67373c3896						
Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP options set	Main Route table
Group1-finalproject	vpc-05cdc2e67373c3896	available	10.0.0.0/16	-	dopt-c289f3b9	rtb-07af4922e94050a0d

Description CIDR Blocks Flow Logs Tags

VPC ID	vpc-05cdc2e67373c3896	Tenancy	default
State	available	Default VPC	No
IPv4 CIDR	10.0.0.0/16	Classic link	Disabled
IPv6 CIDR	-	DNS resolution	Enabled
Network ACL	acl-0e1d33de4c82a51c3	DNS hostnames	Disabled
DHCP options set	dopt-c289f3b9	ClassicLink DNS Support	Disabled
Route table	rtb-07af4922e94050a0d	Owner	531997612114

- **Subnet: Group1-finalproject Public Subnet**

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	IPv6 CIDR
Group1-finalproject Public Subnet	subnet-07fb8ab98f31d8fdb	available	vpc-05cdc2e67373c3896 Group1-finalproject	10.0.0.0/17	32759	-

Description Flow Logs Route Table Network ACL Tags Sharing

Subnet ID	subnet-07fb8ab98f31d8fdb	State	available
VPC	vpc-05cdc2e67373c3896 Group1-finalproject	IPv4 CIDR	10.0.0.0/17
Available IPv4 Addresses	32759	IPv6 CIDR	-
Availability Zone	us-east-1c (use1-a26)	Route Table	rtb-0ecd786a0bd60ae71 Group1-finalproject Public Routes
Network ACL	acl-0e1d33de4c82a51c3	Default subnet	No
Auto-assign public IPv4 address	Yes	Auto-assign IPv6 address	No
Owner	531997612114		

- **Security Group: csci91-phase1-group1-sg**

Create Security Group Actions ▾

search : csci	Add filter			
Name	Group ID	Group Name	VPC ID	Description
sg-04604303cb47d1020	csci91-phase1-group1-sg	vpc-05cdc2e67373c3896	Internet access to 80 and 443 and SSH on 22	

Security Group: sg-04604303cb47d1020

Description Inbound Outbound Tags

Edit

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	
Custom TCP Rule	TCP	8080	0.0.0.0/0	
SSH	TCP	22	0.0.0.0/0	
HTTPS	TCP	443	0.0.0.0/0	

- **Jenkins EC2: Public IP of 52.90.18.45**

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
Jenkins	i-03e8af31948c25b76	t2.micro	us-east-1c	running	2/2 checks ...	None		52.90.18.45

Dev EC2 CFT template

The second template called 1-deploy-dev.yaml is a CFT that Jenkins will trigger to create the dev container EC2 when a change is made to the public repo dev branch.

(<https://code.harvard.edu/cwm850/csci-91-final-phase1-private/blob/master/cloud-formation/1-deploy-dev.yaml>). It is located in the private repo. The template will:

- Create a CentOS dev EC2 in the base VPC, subnet, with same security group as the base AWS infrastructure described in the first template
- Installs AWS CLI and uses AWS commands to push its IP address to an S3 bucket so that Jenkins and testing scripts can use it as an endpoint.
- Install Docker Container on EC2 by:
 - Installing, starting, and enabling docker
 - yum install docker -y
 - systemctl enable docker
 - systemctl start docker
 - Creating a directory and download the Dockerfile by curling Phil's Dockerfile and cloning the dev branch of the public repo to get the index.html file
 - mkdir /tmp/build-docker-image && cd /tmp/build-docker-image
 - git clone -b dev <https://github.com/cwmat/csci-91-final-phase1-public.git>
 - curl -O <https://s3.amazonaws.com/final-e91/Dockerfilecentos>
 - mv /tmp/build-docker-image/Dockerfilecentos /tmp/build-docker-image/Dockerfile
 - Building apache httpd image
 - docker build -t centosapache .
 - Running a container from the image and naming it dev
 - docker run --name dev -d -p 80:80 centosapache

Staging EC2 CFT template

The third template called 2-deploy-staging.yaml is a CFT that Jenkins will trigger to create the staging container EC2 when change has passed dev tests and moves to staging

(<https://code.harvard.edu/cwm850/csci-91-final-phase1-private/blob/master/cloud-formation/2-deploy-staging.yaml>). It is located in the private repo. The template will:

- Create a CentOS dev EC2 in the base VPC, subnet, with same security group as the base AWS infrastructure described in the first template
- Installs AWS CLI and uses AWS commands to push its IP address to an S3 bucket so that Jenkins and testing scripts can use it as an endpoint.
- Install Docker Container on EC2 by:
 - Installing, starting, and enabling docker
 - yum install docker -y
 - systemctl enable docker
 - systemctl start docker



- Creating a directory and download the Dockerfile by curling Phil's Dockerfile and cloning the dev branch of the public repo to get the index.html file
 - mkdir /tmp/build-docker-image && cd /tmp/build-docker-image
 - git clone -b staging <https://github.com/cwmat/csci-91-final-phase1-public.git>
 - curl -O <https://s3.amazonaws.com/final-e91/Dockerfilecentos>
 - mv /tmp/build-docker-image/Dockerfilecentos /tmp/build-docker-image/Dockerfile
- Building apache httpd image
 - docker build -t centosapache .
- Running a container from the image and naming it dev
 - docker run --name stage -d -p 80:80 centosapache

GCP Infrastructure Setup - Philip Cunningham and Stephen Akaeze

To set up the GCP infrastructure, two templates were made that sets up the prod instance and ansible server instance

Ansible Server instance - Philip Cunningham

GCP deployment manager to deploy ansible server. The yaml files are in the private repository under GCP templates.

The screenshot shows the Google Cloud Platform Deployment Manager interface. The top navigation bar includes 'Google Cloud Platform' and 'Final Project'. The main area is titled 'Deployments' with a sub-section 'Deployments'. It shows a table with two rows:

Name	Created on
test	3 days ago
test2	1 day ago

Below the table, there is a Cloud Shell terminal window with the following content:

```
Welcome to Cloud Shell! Type "help" to get started.  
Your Cloud Platform project in this session is set to final-project-224917.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
philipcunningham@cloudshell:~ (final-project-224917)$ nano deploy2.yaml
```

The ansible server was configured to have the SSH keys public and private to perform GIT pulls, SSH into client instances and execute playbooks. The template will create the instance in the final project subnet with ubuntu 18.10 on the AZ us-east1-b and configured for a f1-micro machine type.

The screenshot shows a Cloud Shell terminal window with the following content:

```
Welcome to Cloud Shell! Type "help" to get started.  
Your Cloud Platform project in this session is set to final-project-224917.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
philipcunningham@cloudshell:~ (final-project-224917)$ nano deploy2.yaml  
philipcunningham@cloudshell:~ (final-project-224917)$ gcloud deployment-manager deployments create ansible --config deploy2.yaml
```



Unlock more of Google Cloud by upgrading now (\$247.77 credit and 347 days left in your free trial).

☰ Google Cloud Platform • Final Project ▾

Deployment Manager Deployments + DEPLOY MARKETPLACE SOLUTION ⚡ DELETE

Deployments Filter by label or name

Name Created on

test 3 days ago

(final-project-224917) x +

GNU nano 2.7.4

```
resources:
- name: ansible-server-1
  type: compute.v1.instance
  properties:
    zone: us-east1-b
    machineType: https://www.googleapis.com/compute/v1/projects/final-project-224917/zones/us-east1-b/machineTypes/f1-micro
  metadata:
    items:
      - key: ssh-keys
        value: |
          phase1:ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQAC6iUaH51orMeOPlfTuFFUIHpl0XWl3PwLuJN01mETYwW1rWrnsuYEvYTCy4bN918QDRxyGY09
      - key: startup-script
        value: |
          #! /bin/bash
          apt-get update
          apt-get install git -y
          sudo apt install ansible -y
          mkdir -p /root/.ssh
          cat << EOF > /root/.ssh/id_rsa
          -----BEGIN RSA PRIVATE KEY-----
MIIEpaIBAAKCAQEaUoLgh+daKzHj5X07hRVCB6zaFlJdz8C7iTaNZhE2MFta1q
S7ImBL2ewsuGzfdfNUAOAcmmNPZv1CxbgTj6raUubbqgDxkhSpk9EDMuUf0JQrxY
HS7xCEwmTe20YsDPirk019hdqdpRxvMPNqrxWJ21zPISqj7/fcuqeYUfOUg04Iw
ttudh2jJOZVUSByPmIJR+ZEg0x1u89QHIEC5mhgu8C2zJemb5zAHNm5uQAgpy9CI
uGitBPC2eie54yEW19GRRXG9DsCgaGheEY7Xkb3/CQa1DUh3cKNO/h98jsv6goPh
b7TAT4yG17u1VnbsKEWNu9LzdSkYjq89iW1YuwIDAQABaIBAQCH6Nkw0AmMiw7o
YTQR+1JxFgpkFhYazEYALD2LFxenznAsjOv1YV1lessu0qg/jUscsHUFqiN91zV1
2xe0lmqwNqf7m0Gps1F9ycnw4Vrs0WB0ETLDp092B3YjYzxz0KxxhPaQofRsNp
Dhc3/2UT0dkp1/5u0KEQ3cBoigpToopiypON/bcRensoyYA02NE+xSMq2NFGCOK
cog5aiySxqjyJTWjkmc6mC1vYqPh0YS7ThoVcpK+uyJRRqb8mw9TqhJrQ6ZBP0Gcm
N25KA6GsQbrwf6K/eFknVOIHCK0A6A66uY+fWTxIRdfeFKINK0Joc4zvDPDNY9h
DfLr21yBaoGBAPOT7ra/9sj0FVNuocRuis8+6L7qtzYnSlkNFW8f9szqdIxJttKr
lhbBl0BahFm79Ksg/i0gK1YfGMM79Ak8iqDDdiWrWfx8b545UF3liLz64CAmVK16
1fQdVdpOrhy1TT/ihdu3sUrK2qPELyC8/uZ4RBwfRrcz6tbvMA3TvhAoGBAMQM
odTpqXpq9W68yTDOwApOj5GjNS1x/MPwDmID4M6pym3Mvjfa+vB1qgdXB1USZrm3Mb
oVsH3bbd/kb5ShuipkLM+EXua7wQgYl2681cs/Z2qh/7WH6dP8oAOEJk/Vch/r2
oc0Wv7T7aTwQNGt/oGL2BkU5VpmiqAi85qXWQgbAoGBAMfhEP2ze5aa0QWeP7mg
TPqM0Un6KqaR0E8fYxtza3/0gJ+0zixLCe3QO3FoVjuc9caQ3U0259kgY52y0Svt
Np9+RrHaz1QA90eMGjChELij1tALJGnH51MPzlKax15Td2SLnghY0XBiztCuJPJk
px2orZu2PrjRRsfd4swe5W2BAoGAVGNjndGvwoaSDhmVgXcG8N8gNIR58zB/s1sQ
```

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to **final-project-224917**.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
philipkcunningham@cloudshell:~ (final-project-224917)\$ nano deploy2.yaml
philipkcunningham@cloudshell:~ (final-project-224917)\$ gcloud deployment-manager deployments create ansible --config deploy2.yaml
The fingerprint of the deployment is yboFy-bAVGg0iZ28qfFQlw==
Waiting for create [operation-154524677740-57d64cd470ae1-95e41128-20481af7]...done.
Create operation operation-154524677740-57d64cd470ae1-95e41128-20481af7 completed successfully.
NAME TYPE STATE ERRORS INTENT
my-second-vm compute.v1.instance COMPLETED []
philipkcunningham@cloudshell:~ (final-project-224917)\$

The ansible server can be launched in the GCP console by using deployment manager commands, naming the server, then referencing the deployment yaml.



gcloud deployment-manager deployments create ansible --config deploy2.yaml

GCP deployment manager also makes it extremely easy to modify changes to the configuration yaml and can be updated while the deployment is running.

gcloud deployment-manager deployments update ansible --config deploy2.yaml

After the deployment manager run successfully a new instance is created in the GCP compute engine

Ssh into the ansible instance and we can see that the Deployment manager kicked off the setup of ansible server, pulling the private git repo, moving the folder into /etc/ansible and ran the playbook to configure jenkins on the AWS instance based on the host file ip addresses in the private git repo.

```
phase1@ansible-server-1: ~
-
Ph1lx99@Ph1lx99 MINGW64 ~
$ cd Documents/
Ph1lx99@Ph1lx99 MINGW64 ~/Documents
$ ssh -i "final.pem" phase1@35.185.49.219
Welcome to Ubuntu 18.10 (GNU/Linux 4.18.0-1004-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * MicroK8s is Kubernetes in a snap. Made by devs for devs.
   One quick install on a workstation, VM, or appliance.

   - https://bit.ly/microk8s

 * Full K8s GPU support is now available!
```

```
root@ansible-server-1: ~
-
root@ansible-server-1:~# cd /root/
root@ansible-server-1:~# ls
bin  csci-91-final-phase1-private  Logging.sh  snap
root@ansible-server-1:~# |
```



```
root@ansible-server-1:/etc/ansible
root@ansible-server-1:~# cd /root/
root@ansible-server-1:~# ls
bin  csci-91-final-phasel-private  logging.sh  snap
root@ansible-server-1:~# cd /etc/ansible/
root@ansible-server-1:/etc/ansible# ls
README.md  ansible.cfg  cloudwatch.yaml  hosts      main.yml  roles
ansible     apache.yml    defaults       jenkins.yml  playbooks
root@ansible-server-1:/etc/ansible# |
```

Jenkins playbook can be re run at any time provided that the updated IP address is updated in the host file.

The screenshot shows the AWS EC2 console interface. At the top, there's a navigation bar with links like 'AWS', 'Services', 'Resource Groups', and a search bar. Below the navigation bar, the main area displays a table of EC2 instances. The table has columns for 'Name', 'Instance ID', 'Instance Type', 'Availability Zone', 'Instance State', 'Status Checks', 'Alarm Status', 'Public DNS (IPv4)', and 'IPv4 Public IP'. One instance is listed: 'Jenkins' with Instance ID 'i-03e8af31948c25b76', Type 't2.micro', in 'us-east-1c', 'running', with 'None' alarm status, and Public DNS '52.90.18.45'.

```
root@ansible-server-1:/etc/ansible
root@ansible-server-1:/etc/ansible# ansible-playbook jenkins.yml
PLAY [jenkins] ****
TASK [Gathering Facts] ****
ok: [52.90.18.45]
TASK [jenkins : add jenkins repo] ****
```



Not secure | <https://52.90.18.45/job/Group1%20pipeline%20copy%20test/>

Apps newest submissions: WCCFTech M4 335i f250 TechPowerUp

Jenkins > Group1 pipeline copy test ▾

Delete Pipeline

Configure

Full Stage View

GitHub

Rename

Pipeline Syntax

GitHub Hook Log

Recent Changes

Stage View

Average stage times:
(Average full run time: ~17min)

Build History trend —

Google Cloud Platform Final Project

Compute Engine VM instances CREATE INSTANCE IMPORT VM REFRESH START STOP RESET

VM instances

2 instances could be resized to save money or increase performance. [Learn more](#)

Filter VM instances

Name	Zone	Recommendation	Internal IP	External IP	Connect
ansible-server-1	us-east1-b		10.142.0.2 (nic0)	35.185.49.219	SSH

Prod Instance - Stephen Akaeze

Described in next section

Image of GCP infrastructure setup results

Name	Zone	Recommendation	Internal IP	External IP	Connect
ansible-server-1	us-east1-b		10.142.0.2 (nic0)	35.185.49.219	SSH
final-prod-server	us-west2-a		10.168.0.3 (nic0)	35.236.24.202	SSH



Prod Server Instance, Container Setup and Webhook Configuration - Stephen Akaeze

Purpose

As specified in the project requirements, the Prod server should meet the following requirements,

- Be located in the GCP infrastructure
- Setup a container running apache and sharing the website from the master branch of the public repo
- Set up an update mechanism which triggers the container site to be rebuilt whenever there is an update to public repo.

Implementation

Prod Instance Setup: The prod instance was setup using the GCP console as an n1-standard-1 (1 vCPU, 3.75GB memory) instance in us-west2-a availability zone. The selected operating system is Centos 07. SSH login was enabled using the group 1 shared public and private key pairs. The Prod public IP address is 35.236.24.202.

Prod Container Setup: The Prod container setup was done using a script ([ProdConSetup.sh](#)) which does the following,

- Updates the operating system
- Installs git and docker cli tools
- Stops the current docker container
- Deletes the current docker container and image
- Restarts Docker daemon
- Deletes the old public repo and clones a new public repo
- Creates the a directory for the docker container and moves the necessary apache files to the docker directory
- Creates a new docker container running apache with the [Dockerfile](#)

Based on the above steps, the script can repeatedly executed to successfully build new containers running apache.

Container Update Mechanism: This essentially refers the implementation approach used in updating the Prod container/site after Jenkins has merged stage branch into the master branch on the public repo. The easy approach would be to use a cron job but cron jobs are not efficient as it wastes cpu resources. In a real world environment, the docker container might demand high processing or memory resources which could be compromised by a cron job. Cron jobs are also not every resilient. So we decided to use the github webhooks for the Prod server.



Webhooks are efficient as they are remote notifications informing the prod machine about a new push in the public repo. The webhooks are sent as HTTP POST from github to any registered server in the repo settings. The webhook implementation was handled in two steps

- 1) Step1 - Webhook server: Step 1 is accomplished using a NodeJS server ([hookapp](#)) running on the Prod instance. It utilizes Express.js as a middleware to listen for HTTP POST requests on port 8080 with path “/payload” (as shown hookapp/bin/www and hookapp/app.js files). Whenever a HTTP POST request/webhook payload arrives on port 8080 with path “/payload” (essentially 35.236.24.202:8080/payload), the NodeJS server runs the [ProdConSetup.sh](#) which handles the repo update and docker container setup. The [forever](#) tool was used in running the server in the background as well as logging the [ProdConSetup.sh](#) output for reference as shown below,

```
[root@final-prod-server ~]# forever list
info:    Forever processes running
data:      uid  command          script           forever pid  id logfile              uptime
data: [0] z9b7  /usr/bin/node hookapp/bin/www  2891    2897  /root/.forever/z9b7.log 1:7:50:46.224
[root@final-prod-server ~]# exit
```

- 2) Step 2 - Webhook server registration on github: Step 2 involves registering the NodeJS server as a webhook recipient for github. This is accomplished by selecting add webhook from the public repo → settings ---> webhook page. The “Content Type” was set as Json and “Payload URL” set as “35.236.24.202:8080/payload”, pointing the NodeJS server in the Prod server. The Event was set to as “push” as shown below



Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

http://35.236.24.202:8080/payload/

Content type

application/json

Secret

Which events would you like to trigger this webhook?

- Just the push event.
- Send me everything.
- Let me select individual events.

Active

We will deliver event details when this hook is triggered.

[Update webhook](#)

[Delete webhook](#)

With both steps completed, NodeJS server successfully started receiving webhooks from github and rebuilds the prod container with every webhook payload received. By default, NodeJS is a single threaded and event-driven service so it is certainly more efficient than using a cron job.

Jenkins Pipeline and WebHooks - Charles Mateer

Initial Setup

Once Ansible has installed Jenkins, go to the public IP for the Jenkins instance on HTTPS (port 443).



Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/opt/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Next you will need to SSH to the Jenkins instance and:

```
sudo su -  
cat /opt/jenkins/secrets/initialAdminPassword
```

and use the resulting password to unlock the Jenkins admin.

Choose to install suggested plugins:

Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	Folders
✓ Timestamper	✓ Workspace Cleanup	⌚ Ant	⌚ Gradle	** JDK Tool ** Script Security ** Command Agent Launcher ** Structs ** bouncycastle API ** Pipeline: Step API ** SCM API ** Pipeline: API ** JUnit OWASP Markup Formatter ** Token Macro Build Timeout ** Credentials ** SSH Credentials ** Plain Credentials Credentials Binding
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline: Stage View	Timestamper ** Pipeline: Supporting APIs ** Durable Task ** Pipeline: Nodes and Processes ** Matrix Project ** Resource Disposer Workspace Cleanup
⌚ Git	⌚ Subversion	⌚ SSH Slaves	⌚ Matrix Authorization Strategy	
⌚ PAM Authentication	⌚ LDAP	⌚ Email Extension	⌚ Mailer	** - required dependency



Then setup an admin user:

Create First Admin User

Username:	phase1	
Password:	
Confirm password:	
Full name:	Phase1	
E-mail address:	cmateer@g.harvard.edu	

Use the default Instance Configuration.

Install Plugins and Configure Git Webhooks

The Git plugin was installed as part of the suggested plugins.

Also install, Deploy over SSH.

Go to Credentials > System > Global and add a Secret Text entry with your GitHub Access Key.

Go to Manage Jenkins > Configure System.

Scroll down to the GitHub Section and choose to Add a server:

The screenshot shows the Jenkins GitHub configuration page under the 'GitHub' section. It displays a single GitHub server entry named 'Phase1'. The configuration includes fields for 'Name' (Phase1), 'API URL' (https://api.github.com), and 'Credentials' (phase1github). A note below states 'Credentials verified for user cmat, rate limit: 4998'. There are buttons for 'Test connection', 'Advanced...', and 'Delete'. At the bottom, there's a link to 'Add GitHub Server' and another 'Advanced...' button.

Test the connection to make sure it works.



Go to the public repo > Settings > Web Hooks and add a webhook.

Set the payload URL to the Jenkins server plus “/github-webhook” and the content-type to application/json.

Check Active and then update.

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

Content type

Secret

SSL verification

By default, we verify SSL certificates when delivering payloads.

Enable SSL verification Disable (not recommended)

Which events would you like to trigger this webhook?

Just the push event.
 Send me everything.
 Let me select individual events.

Active
We will deliver event details when this hook is triggered.

Add webhook

You should see a success down below:



Recent Deliveries

✓ 0fd2a076-fbe9-11e8-906a-efd5f3595714 2018-12-09 14:31:44 ...

Request Response Redeliver Completed in 0.2 seconds.

Headers

```
Request URL: https://3.80.203.204/github-webhook/
Request method: POST
content-type: application/json
Expect:
User-Agent: GitHub-Hookshot/a1e410b
X-GitHub-Delivery: 0fd2a076-fbe9-11e8-906a-efd5f3595714
X-GitHub-Event: push
```

Setup Pipeline

Go to Jenkins > New Item and create a Pipeline Project:

Enter an item name

Pipeline » Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of parallel pipelines according to defined branches in one SCM repository.

Configure the pipeline to use the project's public repo and to be triggered by Git webhooks:



General Build Triggers Advanced Project Options Pipeline

Description Group 1 final project pipeline.

[Plain text] Preview

Discard old builds ?

Do not allow concurrent builds ?

Do not allow the pipeline to resume if the master restarts ?

GitHub project ?

Project url <https://github.com/cwmat/csci-91-final-phase1-public/> Advanced...

Pipeline speed/durability override ?

Preserve stash from completed builds ?

This project is parameterized ?

Throttle builds ?

Build Triggers

Build after other projects are built ?

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Disable this project ?

Quiet period ?

Under "Pipeline" you can now add the pipeline script from this repo `phase1-pipeline`.

Pipeline

Definition Pipeline script

Script

```
1- timestamps {
2-
3- node () {
4-
5-   stage ('Build - Checkout') {
6-     checkout([$class: 'GitSCM', branches: [[name: '**/dev']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: []])
7-   }
8-   stage ('Build - Build') {
9-     // Shell build step
10-    sh """
11-      cat index.html
12-    """
13-  }
14-  stage ('Test-Cases - Checkout') {
15-    Checkout([$class: 'GitSCM', branches: [[name: '**/dev']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: []])
16-  }
17-}
```

Use Groovy Sandbox ?

[Pipeline Syntax](#)

The pipeline will now run whenever a commit is made to the dev branch of the public repo.

The pipeline uses two stand alone jobs which utilize the Git Publisher plugin (which is not supported in pipelines) to merge changes to the staging and master branches of the public GH repo.

Additionally, the pipeline uses freestyle jobs using the **AWS Cloud Formation plugin** to create a CF stack from a CF template in the private repo.



Once the CF stack is stood up, a docker script is invoked to build the project.

Once the project is built, it is tested. The IPs for the EC2 instance that Jenkins stood up are pushed to an S3 bucket which the Python test scripts CURL from to perform the tests. Upon success, the CF stack is torn down and the code is merged and committed to the next branch (staging or master).

Freestyle Job Configurations

Commit to Staging

The screenshot shows the Jenkins Freestyle Job configuration page for a job named "Commit to Staging".

General Tab:

- Description: Git Staging
- Discard old builds: Checked
- Strategy: Log Rotation
 - Days to keep builds: []
 - If not empty, build records are only kept up to this number of days
 - Max # of builds to keep: 2
 - If not empty, only up to this number of build records are kept
- Advanced... button

Source Code Management Tab:

- Repository: Git
 - Repository URL: https://github.com/cwmat/csci-91-final-phase1-public/
 - Credentials: cscie91-jenkins/*****
 - Advanced... button
 - Add Repository button
- Branches to build:
 - Branch Specifier (blank for 'any'): */dev
 - Add Branch button
- Repository browser: (Auto)
- Additional Behaviours:
 - Merge before build
 - Name of repository: origin
 - Branch to merge to: staging
 - Merge strategy: default
 - Fast-forward mode: --ff
 - Add button



Build Triggers

- Trigger builds remotely (e.g., from scripts) (?)
- Build after other projects are built (?)
- Build periodically (?)
- GitHub hook trigger for GITScm polling (?)
- Poll SCM (?)

Build Environment

- Delete workspace before build starts (?)
- Use secret text(s) or file(s) (?)
- Abort the build if it's stuck (?)
- Add timestamps to the Console Output (?)
- Create AWS Cloud Formation stack (?)
- Execute shell script on remote host using ssh (?)
- With Ant (?)

Build

Execute shell

Command `echo "Committing to staging from dev."`

Post-build Actions

Git Publisher

Push Only If Build Succeeds (?)

Merge Results (?)

If pre-build merging is configured, push the result back to the origin

Force Push (?)

Add force option to git push

Tags Add Tag (?)

Branches Add Branch (?)

Branch to push: `staging`

Target remote name: `origin`

Notes Add Note (?)

Notes to push to remote repositories

Add post-build action ▾

Commit to Master

Same as above but the “Branch to Push” under Post-Build Actions > Git Publisher is “master”.

Create Dev CF



General Source Code Management Build Triggers Build Environment Build Post-build Actions

Description

[Plain text] [Preview]

Discard old builds ?

GitHub project ?

This build requires lockable resources ?

This project is parameterized ?

Throttle builds ?

Disable this project ?

Execute concurrent builds if necessary ?

Source Code Management

Git ?

Repositories

Repository URL: git@code.harvard.edu:cwm850/csci-91-final-phase1-private.git Advanced...

Credentials: group1 (ssh key for deployments) Add

Add Repository

Branches to build

Branch Specifier (blank for 'any'): */master X

Add Branch

Repository browser: (Auto) ?

Additional Behaviours: Add ▾

Subversion ?

Build Triggers

Trigger builds remotely (e.g., from scripts) ?

Build after other projects are built ?

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Build Environment

Delete workspace before build starts ?

Use secret text(s) or file(s) ?

Abort the build if it's stuck ?

Add timestamps to the Console Output ?

Create AWS Cloud Formation stack ?

Execute shell script on remote host using ssh ?

With Ant ?

Build

AWS Cloud Formation

Stack configuration

AWS Region	US East (Northern Virginia) Region
Cloud Formation recipe file/S3 URL (.json)	cloud-formation/1-deploy-dev.yaml
Stack name	jenkins-dev
Stack description	
Cloud Formation parameters	
Timeout (seconds)	0
AWS Access Key	AKIAIBTEWFW2KTBCT7IQ
AWS Secret Key
Sleep Time (seconds)	0

Add another AWS Stack

Add build step ▾

Create Staging CF

Same as above but invokes the staging CF script.

Test Cases - Jingjing Zheng

Two tests are created to deliver actionable feedback for each stage, make sure any changes/release are safe to proceed through the delivery pipeline.

The site is up

In this test, the script sends a GET request to the site, then the unittest will check the response status from the header. If the response status code is 200, the site is up, and it is reachable.

The content is what you expected

In this test, the script sends a GET request to the site hosting on dev / staging server. Then save the response data, which basically is a HTML formatted content. HTMLParser lib is used to extract content from a specific div tag. We use unittest verify whether the content is what we expected. Tests are ran in our project as below:

1. Testcase #1, content from title.
2. Testcase #2, content from section header

Website - Elias Falconi

The website developed is a fully functional site that sells ADT security (<http://35.236.24.202/>). In the public repo, Elias is both Usernamelias and Second Admin.



Background

This was a site I created for a client while briefly working as a web developer. The client wanted the site in WordPress. However, in this project, I recreated it as a static website. Since I no longer had the code for the site, everything was created from scratch. The only thing I had was the psd file, which showed exactly how the client wanted the site, with all the images, text, etc.

For this group project, we decided to create a full-blown website so that we can work under realistic CI/CD workloads and conditions. Having a website with far more content and code gives us more robust testing of the infrastructure, Jenkins pipeline, and test cases. The project team wanted to go above and beyond the requirements outlined in class that required an additional role for web development. This additional role required a lot of collaboration with other team members and ensured that our infrastructure solutions were as flexible, scalable, and robust.

Initial setup

Usually when creating a website, I first start with the HTML and CSS simultaneously, and then later add functionality (either JS or server side code) which is how I approached this project. Also, I usually always use Bootstrap for styling, and jQuery for the JavaScript. So one of the first things I did was include the link and script tags for them.

```
<head>
  <meta charset="utf-8">
  <meta http-equiv="x-ua-compatible" content="ie=edge">
  <title>CSCI-91 Public: Phase 1</title>
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <link rel="stylesheet" href="img/faucon.css">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFaoApmY8liuXoPKF0JwJ8ERdknLPM0" c
  <link rel="stylesheet" href="css/normalize.css">
  <link rel="stylesheet" href="css/main.css">
  <link href="https://fonts.googleapis.com/css?family=Montserrat:thin,regular,medium,bold,100,200,300,400,500,600"
    rel="stylesheet">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.5.0/css/all.css"
    integrity="sha384-B4dIHKNB188c12pWKCkhzcICo0wtJAoU8YZTYSqE0IdGSeTK6s+L3BLxeVIU"
    crossorigin="anonymous">
</head>
```

```
<script src="https://code.jquery.com/jquery-3.3.1.min.js" integrity="sha256-FgCb/KJQlLNf0u91ta32o/NMZltwRo8QtmkMRdAu8=">
  crossorigin="anonymous"</script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
  integrity="sha384-ChfqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ60W/JmZQ5stwEULTy"
  crossorigin="anonymous"></script>
<script type = "text/javascript"
  src = "https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-ui.min.js">
</script>
</body>
```

After including the tags for Bootstrap and jQuery, I worked on the HTML and CSS for each section of the site, one by one. That is, first the top section:



The screenshot shows the homepage of JAL Security. At the top left is the Harvard Extension School logo. The top right features contact information: email (info@jalsecurity.com) and phone (444-358-7656). Below this is a navigation bar with links: HOME, ABOUT, SERVICES, PACKAGES, and CONTACT. The main content area has a dark background with a large image of a house. Overlaid on the image is the text "GET ADT MONITORED SECURITY". Below this, a smaller text box says: "It's amazing how little it costs to have the peace of mind of ADT monitored security. A single ADT monitored security system can help protect you and your family from burglary, fire and high levels of carbon monoxide - 24 hours a day, seven days a week." At the bottom left are two buttons: "WHAT DO I GET?" and "LEARN MORE!". On the right side, there is a call-to-action box with a yellow header "CONTACT US FOR A FREE QUOTE". Inside the box, it says "Starting at \$36.99/mo" and includes a note about a 36-month monitoring contract. There are four input fields for Name*, Email*, Phone*, and Best Time to Reach Me*. At the bottom is a blue "SUBMIT FREE QUOTE!" button.

The top section includes everything in the screenshot above, and wrapped in a container (using Bootstrap's “container-fluid” class), with three rows. The first row contains the logo in the upper left corner and the contact info in the upper right corner. The second row is just the navbar. And the third row is everything else in the top section, divided into two columns. The first column has a grid width of eight, and contains the text “GET ADT MONITORED...” and the two buttons beneath it. The second column has a grid width of four, and contains the contact form. Lastly, I gave the whole top section a background image using CSS.

The next section I worked on was the “About” section:

ABOUT US

WE KNOW YOUR NEIGHBORHOOD

We are JAL Custom Home Security, your local ADT Authorized Dealer. We are the people to call if you want to put America's #1 name in security to work for you. We live here in Philadelphia, PA. We know your neighborhood. So we can help you determine what's best for you.

Our offices are located in the Philadelphia area. And our sales personnel and installers have all been trained by ADT. So we will take care of you quickly and with complete satisfaction. Give us a call at [215.900.6217](#) and let us show you what's available. You're going to like how it feels to have peace of mind.



Again I used Bootstrap, and wrapped the content in the “container-fluid” class, followed by a “row.” Within the row, I created two columns: one with a grid width of eight (the “ABOUT US...” text) and the other has a grid width of four (the image).

Then I worked on the “Smart Technology” (A.K.A. “Services”) section:



This whole section is wrapped in the “container-fluid” class as well, followed by a “row.” Within the row there is just one column, which spans the whole grid (width of 12). Like the top section, the background image was included via CSS.

Next, I worked on the “Packages” section:

CHOOSE THE ADT PLAN
FOR YOUR BUDGET AND **LIFESTYLE**

TRADITIONAL	REMOTE	VIDEO	CONTROL
\$36.99/mo	\$49.99/mo	\$58.99/mo	\$89.99/mo
24/7 Burglary & Theft Monitoring from ADT			
Voice Controlled App	Voice Controlled App	Voice Controlled App	Voice Controlled App
Custom Security Alerts	Custom Security Alerts	Custom Security Alerts	Custom Security Alerts
Remote Arm & Disarm			
	Live Video	Live Video	Live Video
			Remote Door Lock & Unlock
			Lights & Thermostat Control
			Garage Door Control

CONTACT TO ORDER **CONTACT TO ORDER** **CONTACT TO ORDER** **CONTACT TO ORDER**

First, I wrapped everything up in the container-fluid class. Then I included the section headings (“CHOOSE THE ADT PLAN...”). Next, right underneath the headings I have a row, which is divided into four columns of equal grid width (width of 3). Then in each column, I included a Bootstrap card for each of the packages (i.e. Traditional, Remote, Video, and Control), and then styled accordingly.

Then after the mid-sections were taken care of, I started on the footer:



The screenshot shows the footer area of the Harvard Extension School website. It includes contact information (123 Main Street, Bristol, PA 19007), a phone number (215)900-6217, an email address (info@alsecurity.com), and social media links. Below this is a row of four columns: 'QUICK LINKS' (with links to HOME, ABOUT, SERVICES, PACKAGES, and CONTACT US), 'LATEST NEWS' (with three blog post titles), and 'NEWSLETTER SIGNUP' (with an input field for 'Enter Your Email' and an 'OK' button). A disclaimer at the bottom explains ADT's monitoring services and terms.

*\$99.00 Customer Installation Charge. 36-Month Monitoring Agreement required at \$36.00 per month. (PA1394) 24-Month Monitoring Agreement required at \$36.00 per month. (PA1394) ADT Home Security Monitoring Services include the ADT Quality Service Plan (QSP) or ADT's Extended Limited Warranty. Form of payment must be by credit card or electronic charge to your checking or savings account. Offer applies to homeowners only. Up to \$1,000.00 in coverage is included. Higher coverage levels are available. Certain restrictions may apply. Offer valid for new ADT Authorized Dealer customers only and not on purchases from ADT LLC. Other rate plans available. Cannot be combined with any other offer.

ADT Pulse Interactive Solutions Services, which help you manage your home environment and family, require the exclusive installation of an ADT alarm system with integrated burglar service and a compatible telephone, cell phone, or FRS with Internet and email access. You must own or lease the premises that ADT services do not cover the operation or maintenance of any household equipment or systems that are connected to the ADT Pulse interactive Solutions Services equipment. ADT Pulse Interactive Solutions Services are not available with the telecommunications of ADT Pulse Interactive Solutions Services. All ADT Pulse Interactive Solutions Services are not available in all geographic areas. Standard message and data rates apply to texts sent. You may be required to pay additional charges to have those equipment required to utilize the ADT Pulse Interactive Solutions Services features you desire. Free Installation, ADT Pulse includes up to 5 wireless indoor/boundary cameras at no charge. You must redeem the wireless indoor/boundary cameras at time of install. Limit one per new ADT customer and cannot be combined with other offers or discounts. Burglary, Fire, Carbon Monoxide and Medical Alert monitoring requires purchase and/or activation of an ADT security system with integrated burglar, fire, carbon monoxide and medical alert sensors. Other ADT services (QSP & ADT's Extended Limited Warranty). All monthly monitoring contract is renewed from year to year. \$1,000.00 in coverage is included. Monitoring Agreement required for 36 months. (PA1394) 24-Month Monitoring Agreement required for 24 months. (PA1394) Monitoring fees are waived for the first 12 months. Additional monitoring fees apply thereafter. Monitoring fees are waived for the first 12 months. Additional monitoring fees apply thereafter. Prices may apply by market. Some insurance companies do not cover homeowner's insurance. Please contact your insurance company. Local zoning laws may be required. Satisfaction is not history. New and additional monitoring fees apply for remote monitoring services. Prices are for residential accounts only and may not reflect the actual production fee actually provided.

The footer contains three rows. The first row is split into four columns of equal grid width, and contains the contact info (address, phone number, email address, and social media links). The next row is also split into four columns of equal grid width, and contains the JAL logo and disclaimers, menu links, news, and a newsletter sign-up form. The last row contains one column, which spans the whole grid, and just contains additional disclaimers. Again, like the other sections, HTML was added first, followed by custom CSS.

Responsiveness

Bootstrap helps tremendously with creating a responsive website, but simply adding some Bootstrap classes isn't enough. So, I went through section by section again, adding additional custom CSS that will make the site more responsive, such as media queries.

```
@media only screen and (max-width: 767px){  
    .contact-info{  
        display: none;  
    }  
    .top-section-h1{  
        font-size: 2em;  
    }  
    .top-section-h2{  
        font-size: 3.5em;  
    }  
    ...  
}
```



```
@media (max-width:1556px){
    .top-heading .card{
        margin-bottom: 0;
        margin-right: 0;
    }
}
@media (max-width:991px){
    .top-heading .card{
        margin-bottom: 40px;
    }
    .mid-section-3 .card{
        height: 600px;
        width: 80%;
        margin-left: auto;
        margin-right: auto;
        margin-bottom: 20px;
    }
}
```

```
@media (max-width:1500px) and (min-width:992px){
    .mid-section-1 p{
        margin-left: 10px;
        margin-right: 10px;
    }
    .mid-section-1-left{
        margin-top: 10px;
    }
    .mid-section-1-heading{
        margin-bottom: 5px;
    }
    #mid-section-1-h2{
        margin-bottom: 10px;
    }
}
@media (max-width:1296px) and (min-width:992px){
    .mid-section-1 p{
        font-size: 0.8em;
        margin-left: 0;
        margin-right: 0;
    }
    .mid-section-1-left{
        margin-top: 5px;
    }
}
```



After adding the media queries, I moved on to adding some JavaScript.

JavaScript

First thing I did was create new div tags on the top of each section that will serve as a destination for the menu links. For example the “about” div below:

```
<div id="about"></div>
<div class="container-fluid mid-section-1">
```

This div with id=”about” will serve as destination for the About menu item.

```
<li class="nav-item">
  <a id="abt" class="nav-link abt" href="#about">About</a>
</li>
  . . .
```

Then, finally, I added the JavaScript for each menu item:

```
$(".abt").click(function(e) {
  $('html, body').animate({
    scrollTop: $($this, 'href')).offset().top
  }, 800);
});
$(".svcs").click(function(e) {
  $('html, body').animate({
    scrollTop: $($this, 'href')).offset().top
  });
});
$(".pkgs").click(function(e) {
  $('html, body').animate({
    scrollTop: $($this, 'href')).offset().top
  }, 800);
});
```

What this code does is gracefully move the view to the part of the page specified in the href of the menu item. By gracefully I mean you see it moving. Without the JavaScript, it would just jump to that part of the page, without any noticeable motion.



The same was done for the “Contact” link, except that after it’s clicked, the header of the contact form pulsates to place emphasis on it.

```
$(".contact").click(function(e) {  
  
    $('html, body').animate({  
        scrollTop: $($().attr(this, 'href')).offset().top  
    }, 800);  
    $(".top-heading .card-header").effect( "pulsate", {times:5}, 3000 );  
});
```

After the menu items were taken care of, I included a jQuery sticky navbar plugin, which will make the navbar follow the user while he/she scrolls the page. I included the JS for this in the js directory, and then changed some HTML and initiated it with the following code:

```
<script type="text/javascript">  
$(document).ready(function() {  
    $('.stickyMenu').smoothMenu({  
        stickyMenu: true,  
        slidingLine: true  
    });  
});  
</script>
```

I then turned my attention to all the buttons on the site. I gave them functionality by having them link to the appropriate places on the page (e.g. the “Contact to Order” button in the Packages section, which links to the contact form).

Miscellaneous info

Throughout developing the site, and again after everything was done, I changed any HTML, CSS, or JS that I felt would make the site better. For example, changing the font-size, font-weight, colors, the opacity of the navbar, adding HTML form validation, etc.

The font I used is from Google Fonts and it’s called Montserrat, which was included in the HTML with this link tag:

```
<link  
href="https://fonts.googleapis.com/css?family=Montserrat:thin,regular,medium,b  
old,100,200,300,400,500,600" rel="stylesheet">
```



While committing to the public repository, I always pushed to the dev branch, with the following commands:

```
git add -all  
git commit -m "example comment"  
git push origin dev
```

The hope would then be that it'll pass the Jenkins test, and get merged to staging. Then while in staging, it should pass its test and get merged to master.

Logging/Monitoring - Christine Liu & Philip Cunningham

Purpose

For production environments, security is a top priority. The team decided to set up logging and monitoring as an above and beyond, implementing DevSecOps. Logs are important for security to see who is accessing the system and required for audit purposes. It gives us the ability to track what is going on in all the instances and gives the team the ability to use logs to troubleshoot while retaining transparency of what is happening in the infrastructure.

Logging/Monitoring in AWS

To implement logging, we wanted to advantage of some of the great AWS resources that we learned about in class and decided to use CloudTrail and a S3 bucket due to its low cost and low overhead. CloudTrail was set up to monitor all the AWS instances and port logs into an S3 bucket. These logs show who logs in, what packages are installed, what packages are running, and general system data.

Steps to create a CloudTrail in AWS

AWS -> Trails -> Create Trail -> Trail name (e91-finalproject) -> Create a new S3 bucket (e91-cloudtrail)

Resulting image of Cloudtrail log created

Trails

Deliver logs to an Amazon S3 bucket. CloudTrail events can be processed by one trail for free. There is a charge for processing events with additional trails. For more information, see [AWS CloudTrail Pricing](#).

Create trail						
Name	Region	Organization trail	S3 bucket	Log file prefix	CloudWatch Logs Log group	Status
e91-finalproject	All	No	e91-cloudtrail		✓	



Logging/Monitoring in GCP

Then, to integrate the GCP infrastructure, research was done to figure out how to integrate the GCP instances into the S3 bucket. We used scripts to set up the prod and ansible instance in GCP to integrate with cloudtrail and port all the logs into the same S3 bucket.

Steps to create logging in GCP

AWS-integration.sh

The AWS integration script will install awscli so the to connect to a s3 bucket. First it will download and install packages unzip and python that will be used for installation of AWSCLI-bundle. Then the script will curl the installation, unzip, install and move the binaries to a specific folder for the second script to use.

GCP-ubuntu-logging.sh - sets up our ansible server to talk to S3 bucket

GCP-CentOs-loggingsetup.sh - sets up our ansible server to talk to S3 bucket

The GCP-ubuntu and centos logging scripts allow GCP instances can push their logs to their respective S3 buckets. The script also stores env variables for AWS configuration, AWS access key, AWS secret key, sets the endpoint and creates a cron job to log system activity every 5 minutes.

Checking Logs

Steps:

Amazon S3 -> Buckets -> select e91-cloudtrail -> select AWSLogs -> select 531997612114 -> select CloudTrail

Image below shows S3 with all the logs. GCP infrastructure logs are in GCP folder. AWS EC2 instances are within folder named after availability zone (scroll list)

Name	Last modified	Size	Storage class
GCP	--	--	--
ap-northeast-1	--	--	--
ap-northeast-2	--	--	--
ap-northeast-3	--	--	--
...			



Example of checking logs

For example, if we wanted to check the GCP logs of the prod server, we would select the GCP folder -> select prod -> select messages -> select download

The screenshot shows the AWS S3 console interface. The path in the top navigation bar is: Amazon S3 > e91-cloudtrail / AWSLogs / 531997612114 / CloudTrail / GCP. A sub-menu bar below the main navigation shows 'Overview' (selected), 'Actions', 'Download', '+ Create folder', and 'Upload'. The main content area displays a list of objects under the 'prod' folder. The columns are: Name, Last modified, Size, and Storage class. The objects listed are 'ansible-server' and 'prod'. Both have a last modified date of '--'. The storage class is also listed as '--'. A timestamp at the bottom indicates 'Dec 17, 2018 10:27:50'.

The screenshot shows the AWS S3 console interface for the 'prod' folder. The path in the top navigation bar is: Amazon S3 > e91-cloudtrail / AWSLogs / 531997612114 / CloudTrail / GCP / prod. A sub-menu bar below the main navigation shows 'messages' (selected), 'Latest version', 'Overview', 'Properties', 'Permissions', and 'Select from'. Below this, there are five buttons: 'Open', 'Download', 'Download as', 'Make public', and 'Copy path'. The 'Properties' section displays the following details:

- Owner:** cmateer
- Last modified:** Dec 18, 2018 4:25:03 PM GMT-0500
- Etag:** 51e39e2b6bb5ba570c0e4890fe5fea7
- Storage class:** Standard

Example of a downloaded log



```
messages
Dec 18 18:14:13 final-prod-server NetworkManager[2376]: <info> [1545156853.9474] dhcpc4
(eth0): gateway 10.168.0.1
Dec 18 18:14:13 final-prod-server NetworkManager[2376]: <info> [1545156853.9474] dhcpc4
(eth0): lease time 3600
Dec 18 18:14:13 final-prod-server NetworkManager[2376]: <info> [1545156853.9474] dhcpc4
(eth0): hostname 'final-prod-server.us-west2-a.c.final-project-224917.internal'
Dec 18 18:14:13 final-prod-server NetworkManager[2376]: <info> [1545156853.9474] dhcpc4
(eth0): nameserver '169.254.169.254'
Dec 18 18:14:13 final-prod-server NetworkManager[2376]: <info> [1545156853.9474] dhcpc4
(eth0): domain name 'us-west2-a.c.final-project-224917.internal'
Dec 18 18:14:13 final-prod-server NetworkManager[2376]: <info> [1545156853.9475] dhcpc4
(eth0): domain search 'us-west2-a.c.final-project-224917.internal.'
Dec 18 18:14:13 final-prod-server NetworkManager[2376]: <info> [1545156853.9475] dhcpc4
(eth0): domain search 'c.final-project-224917.internal.'
Dec 18 18:14:13 final-prod-server NetworkManager[2376]: <info> [1545156853.9475] dhcpc4
(eth0): domain search 'google.internal.'
Dec 18 18:14:13 final-prod-server NetworkManager[2376]: <info> [1545156853.9475] dhcpc4
(eth0): state changed bound -> bound
Dec 18 18:14:13 final-prod-server dbus[2226]: [system] Activating via systemd: service
name='org.freedesktop.nm_dispatcher' unit='dbus-org.freedesktop.nm-dispatcher.service'
Dec 18 18:14:13 final-prod-server systemd: Starting Network Manager Script Dispatcher
Service...
Dec 18 18:14:13 final-prod-server dhclient[2675]: bound to 10.168.0.3 -- renewal in 1371
seconds.
Dec 18 18:14:13 final-prod-server dbus[2226]: [system] Successfully activated service
'org.freedesktop.nm_dispatcher'
Dec 18 18:14:13 final-prod-server nm-dispatcher: req:1 'dhcpc4-change' [eth0]: new request
(3 scripts)
Dec 18 18:14:13 final-prod-server systemd: Started Network Manager Script Dispatcher
Service.
```

FOLDER DESCRIPTIONS

We have two Git repositories, public github and enterprise private code.harvard.edu:

Private Github

code.harvard.edu (enterprise github)

(<https://code.harvard.edu/cwm850/csci-91-final-phase1-private>) with one branch to accommodate:

- a. Branches
 - i. Master
- b. Folders
 - i. ansible folder to hold Ansible configuration files
 - ii. cloud-formation folder to hold CloudFormation files that will set up AWS infrastructure
 - iii. GCP-templates folder that holds files to set up GCP infrastructure
 - iv. docker folder that holds scripts to install docker on VMs
 - v. jenkins folder to hold jenkins pipeline files
 - vi. Security-logging folder that holds logging and monitoring setup for our AWS and GCP infrastructure

Public Github

Public Github (<https://github.com/cwmat/csci-91-final-phase1-public>) to perform version control for web site files and images:

- a. Branches
 - i. Dev
 - ii. Staging



iii. Master

b. Folders

- i. Index.html (file) - full scale web page
- ii. css - folder with css files
- iii. errors - folder that contains HTML files for server errors
- iv. img - image folder that holds all of the images of the website, including the favicon
- v. js - folder that contains js files for the JavaScript features
- vi. tests - folder with test cases
- vii. Prod_Server_Files - folder for the files used for prod
- viii. docker - folder with docker files that developers can use to set up their dev and stage containers (does not contain sensitive information)

FINAL PROJECT FUNCTIONALITY

This shows an example of the end to end testing that we did together.

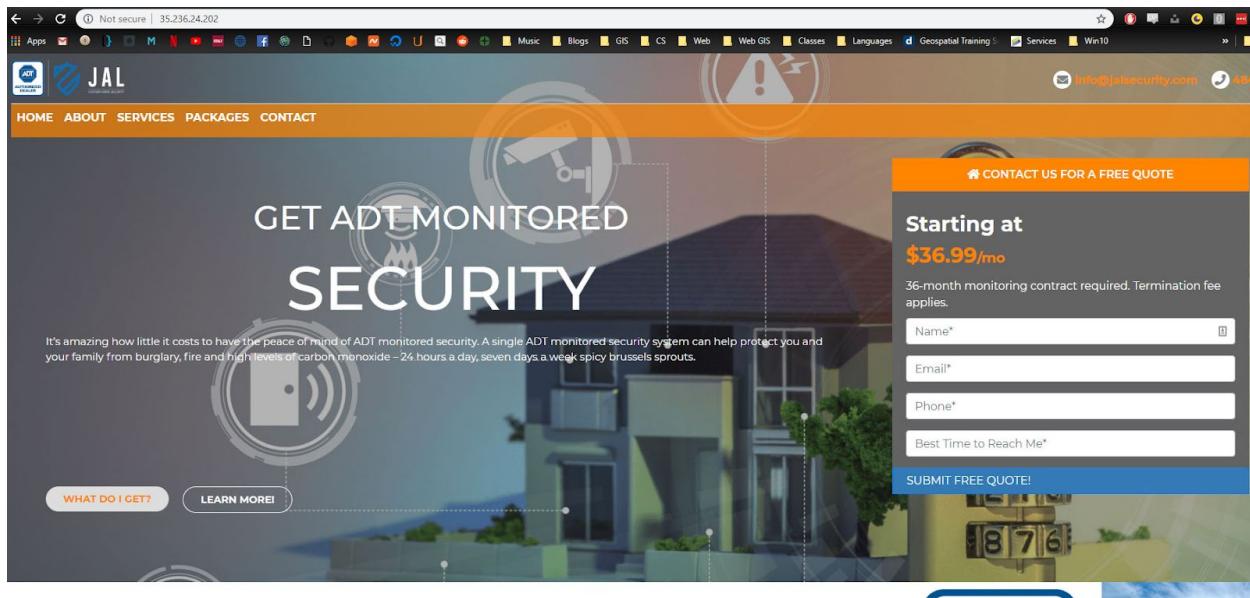
INITIAL WEBSITE

Link to the website

Link to Public Production Website: <http://35.236.24.202/>

Image of the initial website

Production





OUTLINE OF CHANGES

Changes made to the source code and pushed to the dev branch:
Removed text about spicy brussels sprouts and changed the navbar to green.

Changing Files



A screenshot of a code editor showing the file 'index.html'. The code is a snippet of HTML with line numbers from 71 to 97. Lines 71 through 74 show the closing of several HTML tags. Lines 75 through 81 define a row with a single column containing two header elements: 'GET ADT MONITORED' and 'SECURITY'. Line 82 contains a paragraph with descriptive text about ADT monitored security, followed by a note in brackets: '[Removed Text Here]'. Lines 83 through 86 show the opening and closing of a button container and two buttons labeled 'WHAT DO I GET?' and 'LEARN MORE!'. The code is syntax-highlighted with colors for different tags and attributes.

```
5 index.html •
1      </nav>
2      </div>
3      </div>
4      <div class="row">
5          <div class="col-lg-8 top-section">
6              <div class="top-section-h1">GET ADT MONITORED</div>
7              <div class="top-section-h2">SECURITY</div>
8          <p>
9              It's amazing how little it costs to have the peace of mind of ADT
10             monitored security. A single ADT monitored security system can help
11             protect you and your family from burglary, fire and high levels of
12             carbon monoxide - 24 hours a day, seven days a week. [Removed Text Here]
13         </p>
14         <div class="top-section-buttons">
15             <button id="whatdoiget">WHAT DO I GET?</button>
16             <button id="learn">LEARN MORE!</button>
17         </div>
```



```
main.css
93 |     padding: 0.2em 0;
94 |
95 |
96 |     /* =====
97 |     Author's custom styles
98 |     ===== */
99 |
100| {
101|     font-family: 'Montserrat', sans-serif;
102|
103|     img {
104|         max-width: 100%;
105|     }
106|
107|     .navbar {
108|         /* background-color: rgba(255, 132, 0, 0.7) !important; */
109|         background-color: #rgba(0, 255, 55, 0.7) !important;
110|         border-top: 1px solid #lightgray !important;
111|         padding-left: 0;
112|         padding-top: 0;
113|         font-weight: bold;
114|         margin-bottom: 30px;
115|         z-index: 9999;
116|         margin-left: 0 !important;
117|         margin-right: 0 !important;
118|     }

```

Committing changes to the public repo's dev branch:

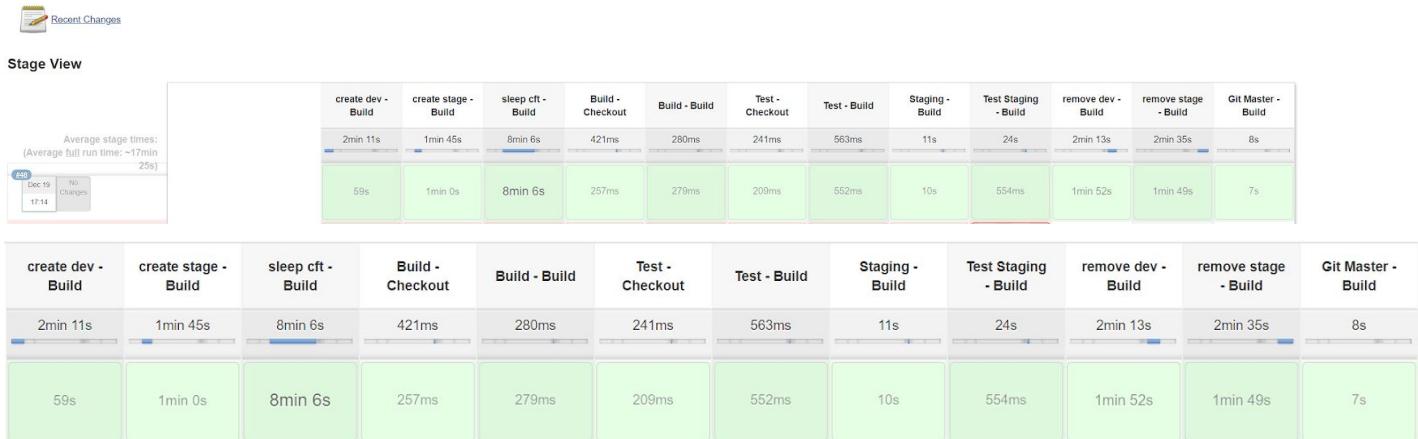
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\chazm\OneDrive\Documents\Projects\CSCI91_Final\temp new\csci-91-final-phase1-public> git add .
PS C:\Users\chazm\OneDrive\Documents\Projects\CSCI91_Final\temp new\csci-91-final-phase1-public> git commit -m"Change nav bar color for test."
[dev 225cf1d] Change nav bar color for test.
2 files changed, 312 insertions(+), 165 deletions(-)
PS C:\Users\chazm\OneDrive\Documents\Projects\CSCI91_Final\temp new\csci-91-final-phase1-public> git push origin dev
Enter passphrase for key '/c/Users/chazm/.ssh/id_rsa':
Counting objects: 5, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.19 KiB | 561.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:cwmat/csci-91-final-phase1-public.git
  0052a76..225cf1d  dev -> dev
PS C:\Users\chazm\OneDrive\Documents\Projects\CSCI91_Final\temp new\csci-91-final-phase1-public>
```

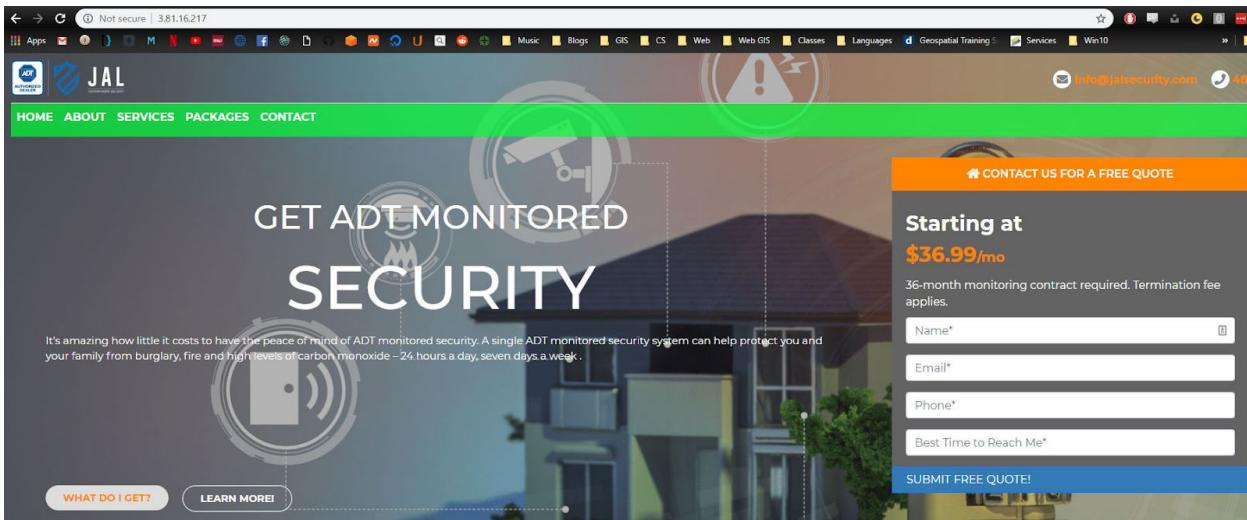


Jenkins CI/CD process

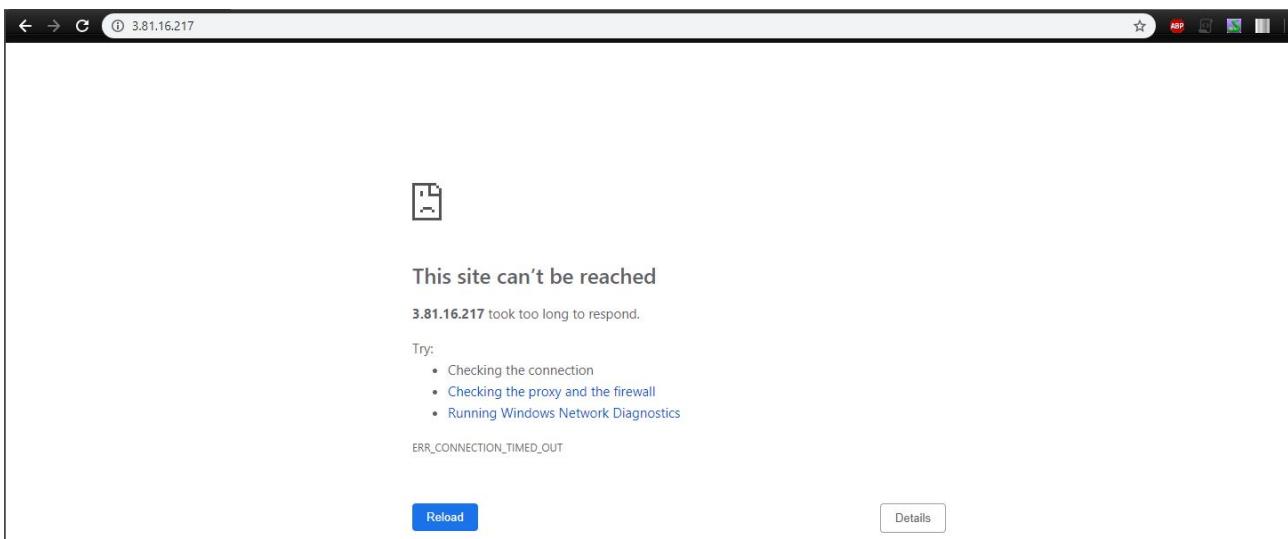
Jenkins Pipeline tests and builds in pushing code to GitHub staging and master branch after tests have passed. Jenkins also builds and tears down dev and stage environments when changes pass tests.



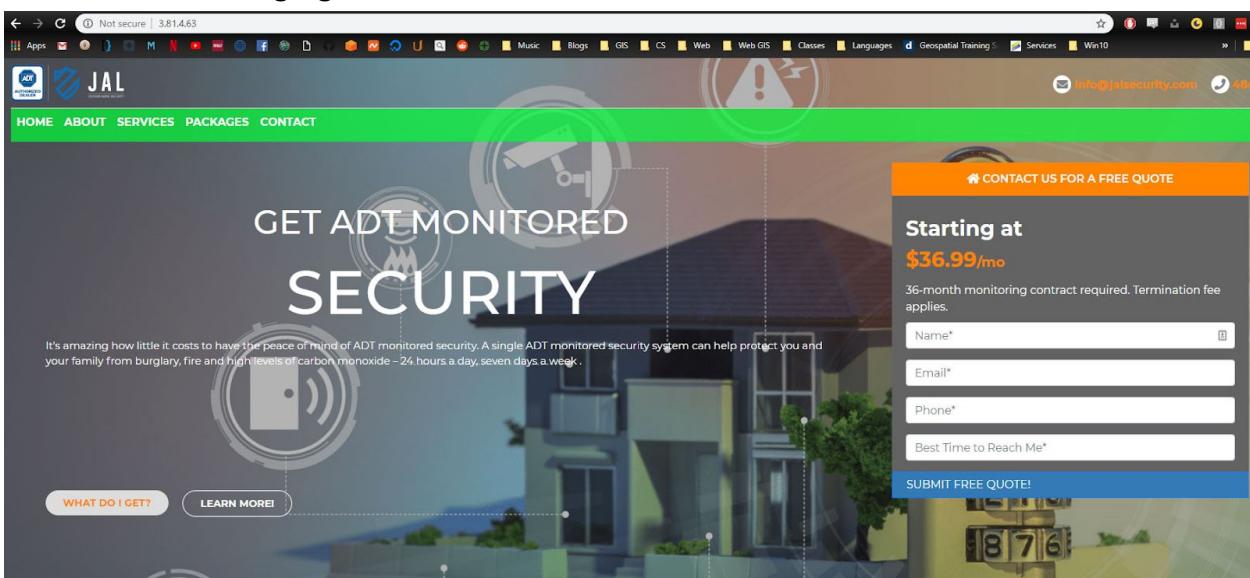
Jenkins created dev environment site:



Proof that Jenkins destroyed testing environment after tests passed:



Jenkins created staging site:



Proof that Jenkins destroyed staging environment after tests passed:



RESULTS

Change results reflected in Github

Images of GitHub repo branches. Dev was pushed to directly. Staging and master were updated by Jenkins after passing tests. Commits by Jenkins appear as cwmatt as Jenkins is using cwmatt's (Charles Mateer) access key.

cwmatt / csci-91-final-phase1-public

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: dev csci-91-final-phase1-public / css /

Create new file Upload files Find file History

This branch is 17 commits behind master.

Pull request Compare

cwmatt Change nav bar color for test. Latest commit 225cf1d 7 minutes ago

..

main.css Change nav bar color for test. 7 minutes ago

normalize.css Initial commit. 10 days ago



cwmat / csci-91-final-phase1-public

Unwatch ▾ 2 ⭐ Star 1 Fork 1

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Settings

Branch: staging

csci-91-final-phase1-public / css /

Create new file Upload files Find file History

This branch is 3 commits ahead, 17 commits behind master.

Pull request Compare

cwmat Change nav bar color for test.

Latest commit 225cf1d 8 minutes ago

..

main.css

Change nav bar color for test.

8 minutes ago

normalize.css

Initial commit.

10 days ago

cwmat / csci-91-final-phase1-public

Unwatch ▾ 2 ⭐ Star 1 Fork 1

Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

Settings

Branch: master

csci-91-final-phase1-public / css /

Create new file Upload files Find file History

cwmat Change nav bar color for test.

Latest commit 225cf1d 8 minutes ago

..

main.css

Change nav bar color for test.

8 minutes ago

normalize.css

Initial commit.

10 days ago

Change results reflected on site

Images of production website after Jenkins pipeline has finished:

Prod:

The screenshot shows a web browser window displaying the JAL Security website. The URL in the address bar is 'Not secure | 35.236.24.202'. The page features a green header with links for HOME, ABOUT, SERVICES, PACKAGES, and CONTACT. Below the header, there's a large image of a house with security cameras and a central monitoring hub. Text on the page includes 'GET ADT MONITORED SECURITY' and 'It's amazing how little it costs to have the peace of mind of ADT monitored security. A single ADT monitored security system can help protect you and your family from burglary, fire and high levels of carbon monoxide – 24 hours a day, seven days a week.' There are two buttons at the bottom left: 'WHAT DO I GET?' and 'LEARN MORE!'. On the right side, there's a form for contacting them: 'CONTACT US FOR A FREE QUOTE', 'Starting at \$36.99/mo', and fields for Name*, Email*, Phone*, and Best Time to Reach Me*. A 'SUBMIT FREE QUOTE!' button is at the bottom of the form. The number '876' is visible in the bottom right corner of the image.



DOCUMENTATION FOR USER

This section provides documentation for a developer to start contributing to our open source project.

E91 CLOUD DEVOPS FINAL PROJECT PHASE 1

Welcome to Group 1's open source project called Phase 1! The link to the production site can be found here: <http://35.236.24.202/>. This site is a full scale site that sells ADT security.

MOTIVATION

This project is a final group project for the E91 DevOps class. Utilizing basic and modern Cloud DevOps techniques that we learned from class, we were challenged to develop an open source project that packages our requirements in a single container on public github that accounts for no funding and special treatment of sensitive information. We will use CI/CD techniques to build a pipeline for a website that needs a seamless, end-to-end continuous delivery and deployment workflow. A continuous integration pipeline will support test, configuration, and deployment of a website and a continuous delivery pipeline will support automatic building, testing, and release of code changes to production environment.

HOW IT WORKS

When a user commits to dev, this will trigger our CI/CD pipeline to kickoff. Jenkins will build a dev container in a EC2 instance and run tests. If the tests pass, Jenkins will set up stage EC2 and test, tearing down the dev container. If the tests pass, Jenkins will tear down stage and the changes will reflect in prod and the public github. The building of dev and stage VMs and containers only when a change is initiated ensures that we do not spend money on a development and staging environment.

CONTRIBUTING GUIDELINES

We invite you to contribute! If you wish to contribute to this open source project, follow these steps:

1. Clone the repo if you are a contributor. Else, fork master <https://github.com/cwmat/csci-91-final-phase1-public> .
2. Create a feature branch and make edits .
3. Commit edits to your feature branch and create pull request to dev branch
4. Include these reviewers



- a. Usernamelias
- b. cpmat
- c. diasyzheng2012
- d. cml2197
- e. philc171
- f. scakaeze
5. Once request is approved and merged by reviewer, it will trigger dev branch and it will go through the jenkins CI/CD process
6. If success, you will be able to see the changes you made in the github repos and on the prod site

AUTHORS

Thanks to all the people who created this project

- Christine Liu
- Charles Mateer
- Jingjing Zheng
- Stephen Akaeze
- Philip Cunningham
- Elias Falconi