Project 3 : EAT (AKA: BEATS)

Brian Sandoval (Lead)
Erik Blomquist
Alexander Enaceanu
Tyler Kluszczynski
Shahid Karim

Project Overview

Main Objectives.

1. Set up Docker and dockerfile so that we may use it for our images.
2. Setup ECR and ECS to define tasks and roles
3. Configure Lambda to be able to deploy our staging site and production site.

Team Overview

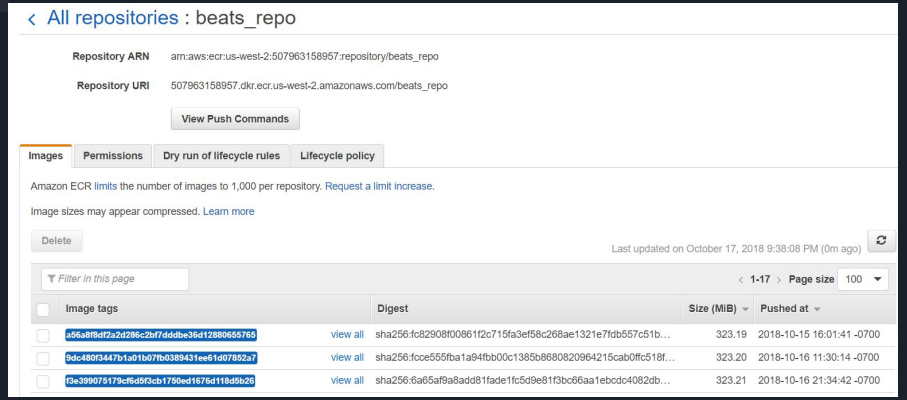
Tasks allocated to each team member:

Name	Tasks
Brian	ECS,ECR
Erik	ECS, ECR, ALB, Route 53
Alex	Docker
Tyler	Lambda, CircleCI
Shahid	Dockerfile

Brian - Tasks (ECR)

```
resource "aws_ecr_repository" "beats_repo" {
  name = "beats_repo"
}

output "beats-repository-URL" {
  value = "${aws_ecr_repository.beats_repo.repository_url}"
}
```



Brian - Tasks (ECS - Tasks & Services)

```
resource "aws_ecs_task_definition" "beats-staging" {
  family                = "beats-staging"
  container_definitions = "${file("task-definitions/staging.json")}"
  volume {
    name      = "beats-staging"
    host_path = "/ecs/beats_service"
  }
  placement_constraints {
    type       = "memberOf"
    expression = "attribute:ecs.availability-zone in [us-west-2a, us-west-2b]"
  }
}

resource "aws_ecs_service" "beats-staging" {
  name            = "beats-staging"
  cluster         = "${aws_ecs_cluster.beats-cluster.id}"
  task_definition = "${aws_ecs_task_definition.beats-staging.arn}"

  load_balancer {
    target_group_arn = "${aws_alb_target_group.HTTP-Group.arn}"
    container_name   = "beats-staging"
    container_port   = 80
  }

  placement_constraints {
    type       = "memberOf"
    expression = "attribute:ecs.availability-zone in [us-west-2a, us-west-2b]"
  }
}
```

```
resource "aws_ecs_task_definition" "beats-production" {
  family                = "beats-production"
  container_definitions = "${file("task-definitions/production.json")}"
  volume {
    name      = "beats-production"
    host_path = "/ecs/beats_service"
  }

  placement_constraints {
    type       = "memberOf"
    expression = "attribute:ecs.availability-zone in [us-west-2a, us-west-2b]"
  }
}

resource "aws_ecs_service" "beats-production" {
  name            = "beats-production"
  cluster         = "${aws_ecs_cluster.beats-cluster.id}"
  task_definition = "${aws_ecs_task_definition.beats-production.arn}"

  load_balancer {
    target_group_arn = "${aws_alb_target_group.HTTP-Group.arn}"
    container_name   = "beats-production"
    container_port   = 80
  }

  placement_constraints {
    type       = "memberOf"
    expression = "attribute:ecs.availability-zone in [us-west-2a, us-west-2b]"
    }
```

Erik - Tasks (ECS - Launch Configuration)

```
# ECS launch configuration
resource "aws_launch_configuration" "ecs-launch-configuration" {
name_prefix           = "ecs-cluster-launch"
image_id              = "ami-00430184c7bb49914"
security_groups       = ["${aws_security_group.NATSG.id}"]
instance_type         = "t2.micro"
key_name              = "${aws_key_pair.deployer.key_name}"
iam_instance_profile  = "${aws_iam_instance_profile.ecs-instance-profile.id}"
user_data             = "#!/bin/bash\necho 'ECS_CLUSTER=beats-cluster' > /etc/ecs/ecs.config\nstart ecs"

root_block_device {
  volume_type = "standard"
  volume_size = 8
  delete_on_termination = true
}

lifecycle {
create_before_destroy = true
}
associate_public_ip_address = "false"
```

Erik - Tasks (ECS ASG and Cluster)

```
# ECS ASG
resource "aws_autoscaling_group" "ecs-autoscaling-group" {
  name = "ecs-autoscaling-group"
  max_size = "2"
  min_size = "1"
  desired_capacity = "2"
  vpc_zone_identifier = ["${aws_subnet.privsubnet1.id}"]
  launch_configuration = "${aws_launch_configuration.ecs-launch-configuration.name}"
  health_check_type = "ELB"
  target_group_arns = ["${aws_alb_target_group.HTTP-Group.arn}"]
tag {
  key = "Name"
  value = "-beats-ecs-asg"
  propagate_at_launch = true
}
}

resource "aws_ecs_cluster" "beats-cluster" {
  name = "beats-cluster"
}
```

Erik - Tasks (ECS, ALB & Route 53)

Load Balancing

Target Group Name	Container Name	Container Port
HTTP-Group	beats-staging	80

beats-cluster >

FARGATE

0	0	0
Services	Running tasks	Pending tasks

EC2

2	2	0	0.06%	0.51%	2
Services	Running tasks	Pending tasks	CPUUtilization	MemoryUtilization	Container instances

Alex - Tasks (Docker)

Setup Docker on local machine

These instructions will be for Ubuntu.

Update the apt package index:

```
$ sudo apt-get update
```

Install packages to allow apt to use a repository over HTTPS:

```
$ sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
Software-properties-common
```

Add Docker's official GPG key:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
```

Verify key with fingerprint:

```
$ sudo apt-key fingerprint 0EBFCD88
```

Should return:

```
pub   4096R/0EBFCD88 2017-02-22
      Key fingerprint = 9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid                  Docker Release (CE deb) <docker@docker.com>
sub   4096R/F273FCD8 2017-02-22
```

Run command to setup stable repo:

```
$ sudo add-apt-repository \
 "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \stable"`
```

Actual install:

```
$ sudo apt-get update
```

Install the latest version of Docker CE

```
$ sudo apt-get install docker-ce
```

Verify that Docker CE is installed correctly by running the hello-world image:

```
$ sudo docker run hello-world
```

Alex - Tasks (Docker)

Search

Dashboard Explore Organizations Create ae637

ae637 Repositories Stars Contributed

Private Repositories: Using 0 of 1 Get more

Repositories

Create Repository +

Type to filter repositories by name

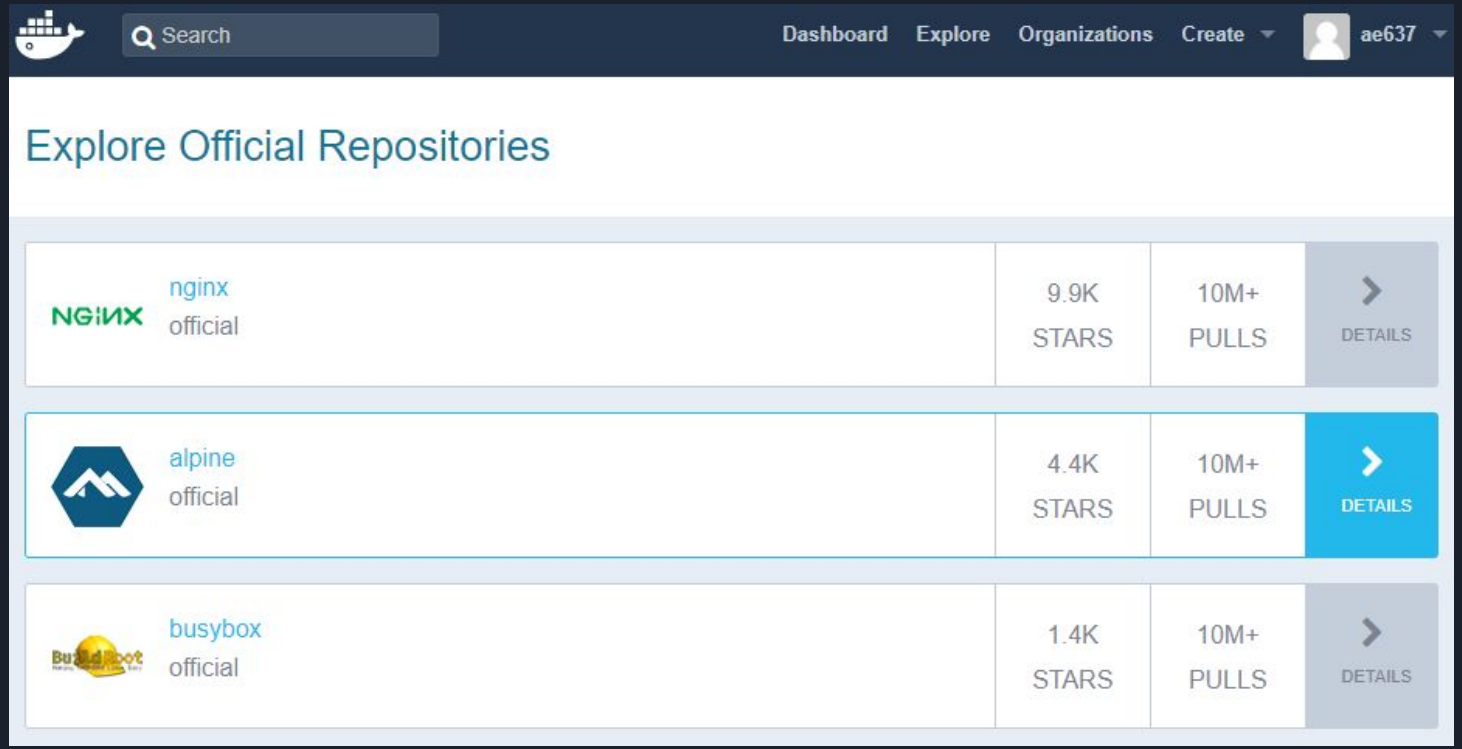
ae637/blogserver2 public	0 STARS	7 PULLS	DETAILS
ae637/testrepo public	0 STARS	2 PULLS	DETAILS

Alex - Tasks (Docker)

Docker Pull Command

```
docker pull ae637/blogserver2
```

Alex - Tasks (Docker)



Tyler - Tasks (CircleCI - Docker)

Setting up docker and building the sites.

40 lines (36 sloc) | 1.3 KB

Raw | Blame | History

```yaml
version: 2
jobs:
  build:
    machine: true
    steps:
      - checkout
      - run:
          command: |
            # Setting up docker here.
            cd fa480.club-dockerfile/

            # Build 1
            docker build --rm=true -t fa480.club-base .
            cd ..
            cd production-site-dockerfile/

            # Build 2
            docker build --rm=true -t finalbuild .

            # Tagging
            docker tag finalbuild:latest 507963158957.dkr.ecr.us-west-2.amazonaws.com/beats_repo:$CIRCLE_SHA1
            cd ..
```

Tyler - Tasks (CircleCI - Docker & ECR)

Uploading docker images to ECR

```
- run:
    command: |
      # Upload image to ECR here.
      sudo $(aws ecr get-login --no-include-email --region us-west-2)
      sudo docker push 507963158957.dkr.ecr.us-west-2.amazonaws.com/beats_repo
```

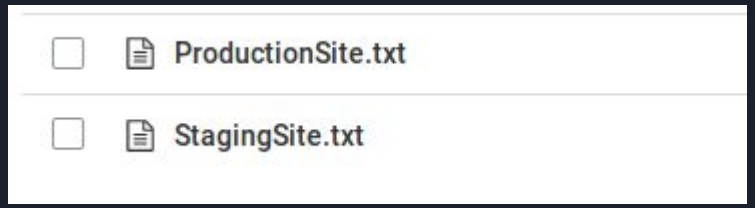
Tyler - Tasks (CircleCI - .txt files)

Uploading txt files to S3 with commit SHA to use.

```
- run:
    command: |
      # This uploads the ProductionSite.txt to s3.
      sudo pip install awscli
      aws s3 cp /home/circleci/project/ProductionSite.txt s3://csuneat-project-2/
      sudo rm /home/circleci/project/ProductionSite.txt
- run:
    command: |
      # Adding Staging Site to s3.
      touch StagingSite.txt
      echo $CIRCLE_SHA1 > StagingSite.txt
      aws s3 cp StagingSite.txt s3://csuneat-project-2
```

Tyler - Tasks (CircleCI - Results)

.txt files on S3



Example ECR image.

Tyler - Tasks (Lambda - Example Response)

Example Response when a .txt file is uploaded to S3

```
event = {'Records': [{
        'eventVersion': '2.0',
        'eventSource': 'aws:s3',
        'awsRegion': 'us-west-2',
        'eventTime': '2018-10-11T19:08:26.592Z',
        'eventName': 'ObjectCreated:Put',
        'userIdentity':
                {'principalId': 'AWS:ID HERE'},
        'requestParameters':
            {'sourceIPAddress': 'IP ADDRESS HERE'},
        'responseElements':
            {'x-amz-request-id': 'AMZ REQUEST ID HERE',
            'x-amz-id-2': 'ID 2 HERE (LONGER)'},
        's3':
            {'s3SchemaVersion': '1.0',
            'configurationId': 'LAMBDA TF CONFIGURATION ID',
            'bucket':
                {'name': 'NAME OF BUCKET',
                'ownerIdentity':{'principalId': 'ID OF OWNER'},
                'arn': 'ARN OF BUCKET'},
            'object':
                {'key': 'FILENAME', 'size': 48, 'eTag': 'ETAG HERE', 'sequencer': 'SEQUENCER NUMBER HERE'}}}]}
'''
```

Tyler - Tasks (Lambda - Setup)

Changing the task definition, happens whenever file is uploaded to S3.

```python
def update(tag,containername,imagename):
    imagename = "507963158957.dkr.ecr.us-west-2.amazonaws.com/beats_repo:"+imagename
    imagename = imagename.rstrip()

    print("Image name is:##" +imagename +"##.")
    response = client.register_task_definition(
        family=containername,
        #taskRoleArn='string',
        networkMode='bridge',
        containerDefinitions=[
            {
                'name': containername,
                'image': imagename,
            #'cpu': 123,
                'memory': 300,
            #'memoryReservation': 123,
            #'links': [
            #    'string',
                'portMappings': [
                    {
                        'containerPort': 80,
                        'hostPort': 80,
                        'protocol': 'tcp'
                    },
                    {
                        'containerPort': 443,
                        'hostPort': 443,
                        'protocol': 'tcp'
                    }
                ],
                'essential': True,
            },
        ],
    )
```

- Imagename/tag: SHA1 of commit.
- ContainerName: Staging or Production.

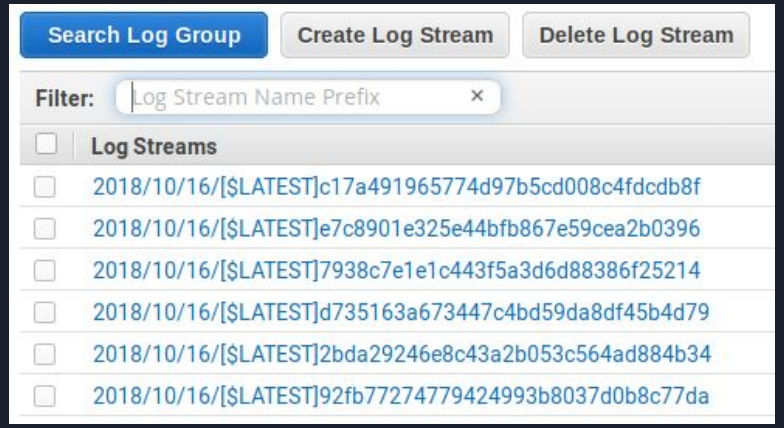
Tyler - Tasks (Lambda - Update)

Updating the cluster, service and task definition.

```
    taskDefinitionRev = response['taskDefinition']['family'] + ':' + str(response['taskDefinition']['revision'])
#print taskDefinition
    response = client.update_service(
        cluster='beats-cluster',
        service=containername,
        desiredCount=1,
        taskDefinition=taskDefinitionRev,
        deploymentConfiguration={
            'maximumPercent': 100,
            'minimumHealthyPercent': 40
        }
    )
```

Tyler - Tasks (Lambda - Cloudwatch)

Logging to test the output of the function. (using CloudWatch)



Tyler - Tasks (Lambda - Cloudwatch)

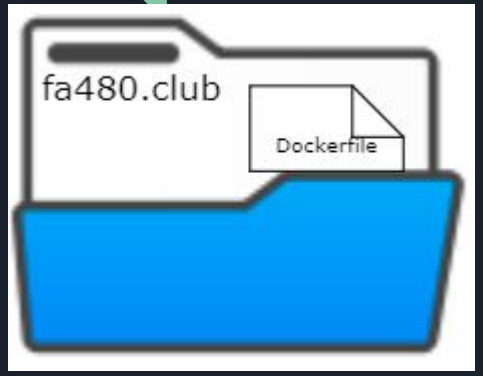
Example successful log.

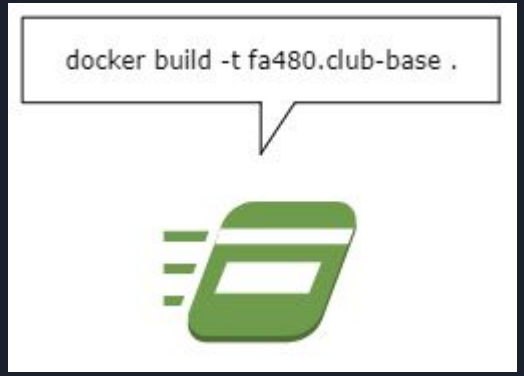
		No older events found at the moment. Retry.
22:26:55	service updated	
22:26:55	START RequestId: 90004f91-d192-11e8-ab4f-1f8cc592af06 Version: $LATEST	
22:26:56	32d1a93a43c9f0b0ec98b68391c303f93f3b8b4f	Image SHA
22:26:56	Image name is:##507963158957.dkr.ecr.us-west-2.amazonaws.com/beats_repo:32d1a93a43c9f0b0ec98b68391c303f93f3b8b4f##.	

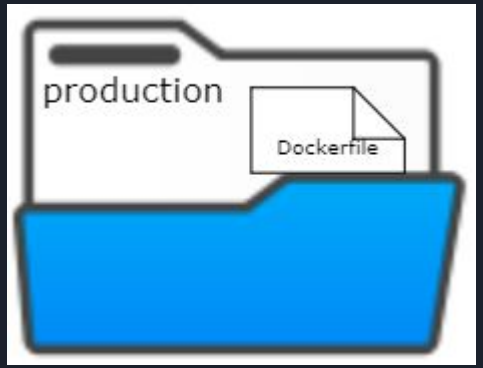
Shahid - Tasks(Dockerfile)

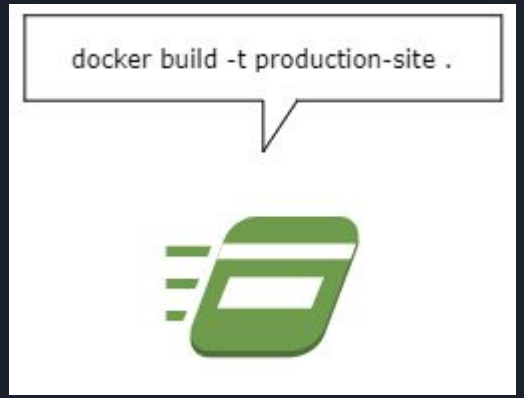
1. Similar to Packer and Ansible in that there are a series of steps that the machine follows that is written to our unique specifications.
2. Specifically our blog requires that we install all our software then get our blog and hugo theme from our blog repo on github. Then we have to do some pre hugo file movement then post hugo file movement in order to properly display our website.
3. We took advantage of docker layering in order to speed up our build times from over 5 minutes to about 25 seconds.
4. Used https://www.fromlatest.io/ : A dockerfile linter in order to help refactor the dockerfile

Shahid - Tasks(Dockerfile)









fa480.club-base Image

Ubuntu 14.04.5 base image

Update package list

Install apache2

Install python-pip

Install dpkg

Install curl

Install git

Install wget

About 430MB before refactor

About 265MB after refactor

production-site Image

fa480.club-base base image

Grab blog files

Configure hugo + theme

Remove misc files

Run hugo

Move created website

About 870 MB before refactor

About 335 MB after refactor

Shahid - Tasks(Dockerfile)

```
FROM ubuntu:14.04.5

# First cd into the folder this docker image is in.
# Then build this dockerfile image by running: docker build --rm=true -t fa480.club-base .
# Tell the image that it's noninteractive so there wont be any waiting for prompts.
RUN echo 'debconf debconf/frontend select Noninteractive' | debconf-set-selections

# Installing minimum software to get the website running.
RUN apt-get update && \
apt-get install --no-install-recommends -y apache2 curl dpkg git python-pip wget && \
rm -rf /var/lib/apt/lists/*
```

Shahid - Tasks(Dockerfile)

```
# Get the website from github
RUN git clone https://github.com/CSUN-SeniorDesign/eat-blog.git

# Pre hugo configuration
RUN hugo new site beats && \
mv /eat-blog/arabica /beats/themes/arabica && \
rm -rf /beats/themes/arabica/exampleSite/* && \
mv /eat-blog/config.toml /beats/config.toml && \
mkdir /beats/content/post/ && \
mv /eat-blog/* /beats/content/post/ && \
# Pre hugo clean up
rm /beats/content/post/BEATS-Uploader.py && \
rm /beats/content/post/ProductionSite.txt && \
rm /beats/content/post/S3-Fetch.py && \
rm -rf eat-blog && \
# Use hugo to create the website
cd /beats && hugo && \
# Move the website to /var/www/html
mv /beats/public/* /var/www/html && \
rm -rf beats

EXPOSE 80/tcp
EXPOSE 443/tcp
```

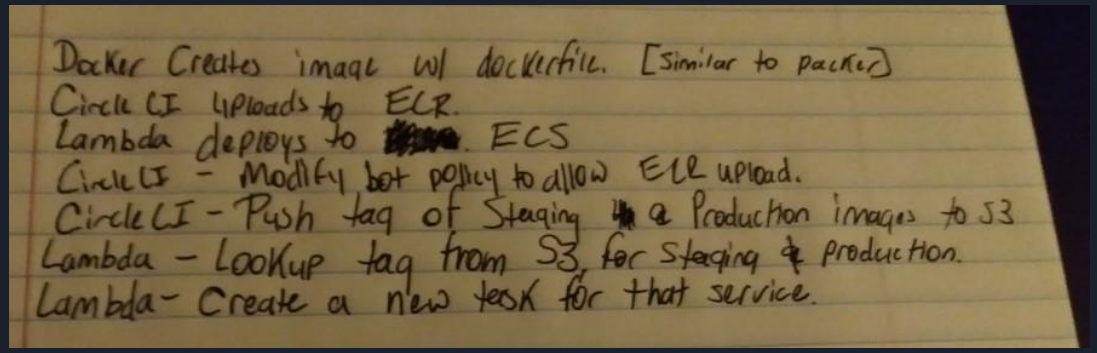
Issues and Lessons Learned

We encountered 4 major issues throughout Project 3.

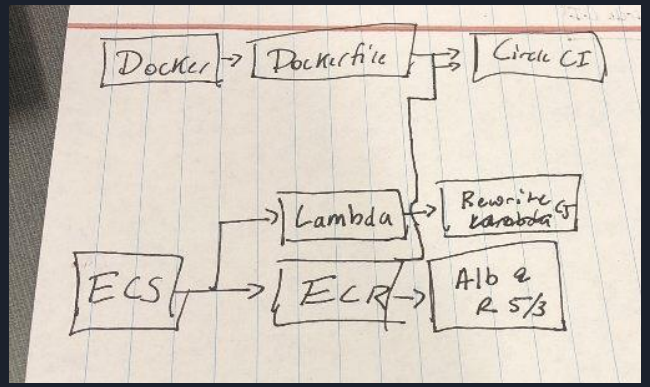
1. Old ASG prevented new ASG to function properly.
2. Services did not attach to the ALB.
3. Volumes had a size of 100 GB.
4. Long build time (Docker)

Project 3 - New Project Management System

Day 1 Documentation

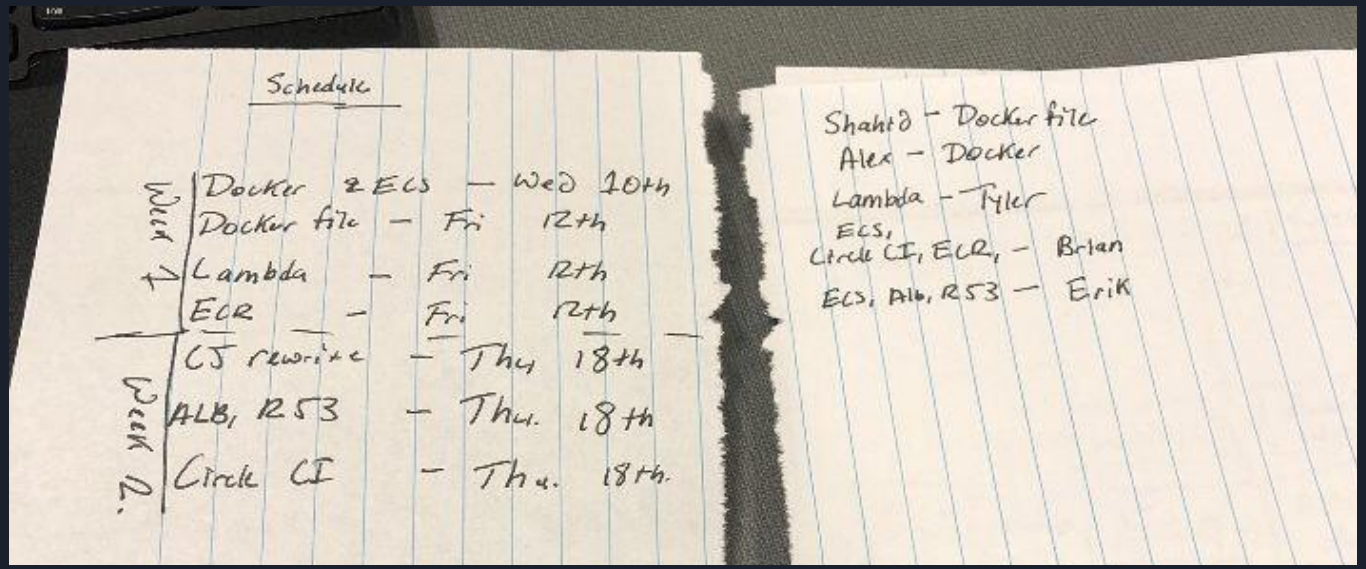


Docker Creates image w/ dockerfile. [Similar to packer]
Circle CI uploads to ECR.
Lambda deploys to [redacted]. ECS
Circle CI - Modify bot policy to allow ECR upload.
Circle CI - Push tag of Staging & Production images to S3
Lambda - Lookup tag from S3 for Staging & production.
Lambda - Create a new task for that service.



Project 3 - New Project Management System

Day 1 Task Assignment and Scheduling



Motivation


YOU NEVER HESITATE TO TACKLE
THE MOST DIFFICULT PROBLEMS.
PANDA EXPRESS

Project Demonstration