The students will install wordpress and create a CLI client to interact with it's API. A CLI based blogging client that will be a part of a CI pipeline, triggered by code changes, performing tests of the code.

[General Requirements](#_3vszknp7vl5a)

[Capstone](#_nqqw84jvujvd)

[0. Setup supporting infrastructure](#_kxn0xzczeg08)

[Run Bamboo on Infrastructure node](#_6qmpsniyjqbf)

[1. Create a pipeline to deploy wordpress app](#_u9c92exmjixi)

[a) Write Docker Compose file for Wordpress](#_ky36aoan5nx4)

[b) Bamboo Build Plan Requirements](#_exkliu3cr8kn)

[c) One-time install REST API Authentication plugin for Wordpress](#_r81yrmio3g4t)

[2. Python client for Wordpress](#_pp6357gwrb3q)

[a) Read configuration from the following ENV variables](#_q2a77b4pmwtt)

[b) Unit tests](#_5qd5m8xd8kl3)

[c) Containerized](#_6twn43591aa)

[3. Create a build pipeline for pyblog](#_nf8nuckxkv6g)

[b) Bamboo Build Plan requirements](#_8wulvnuvuhee)

[4. System and Application Monitoring](#_9qljdhka2vnf)

[a) Metric gathering](#_iuh1n4x2w6hx)

[a) Dashboards](#_lh1iuuvvmru)

[Python CLI Detailed Requirements](#_kf7zapiv1fo6)

[Required Functions](#_9d23zcf3x8to)

[Example Usage](#_g0txfo1gplfb)

[Requirements Matrix](#_32cbt23dds27)

# General Requirements

* Put all "code" (that is, anything you write) into a Git Repository on Github.
* Use this repository to collaborate. See details below for configuration.
  + If you use Pull Requests, reviewers should be your teammate and jrrickerson@redrivetstudios.com. You only need approval from a team member to merge to master though.
  + If you do mob programming or work off of a single repo, please invite [jrrickerson@redrivetstudios.com](mailto:jrrickerson@redrivetstudios.com) as a collaborator to the repo (read only access is fine).
* Automate whenever possible – we want our steps to be fast and repeatable.
* Keep your repo well maintained and organized - no files that aren’t being used, generated files, etc.
* Do not store passwords or API keys in code
* Please tag all resources to make them easier to locate and clean up after the course (“course=IEA”, “cohort=8”)
* Consider disaster recovery when designing things – could you recreate everything you need from what is in your repo?

# Capstone

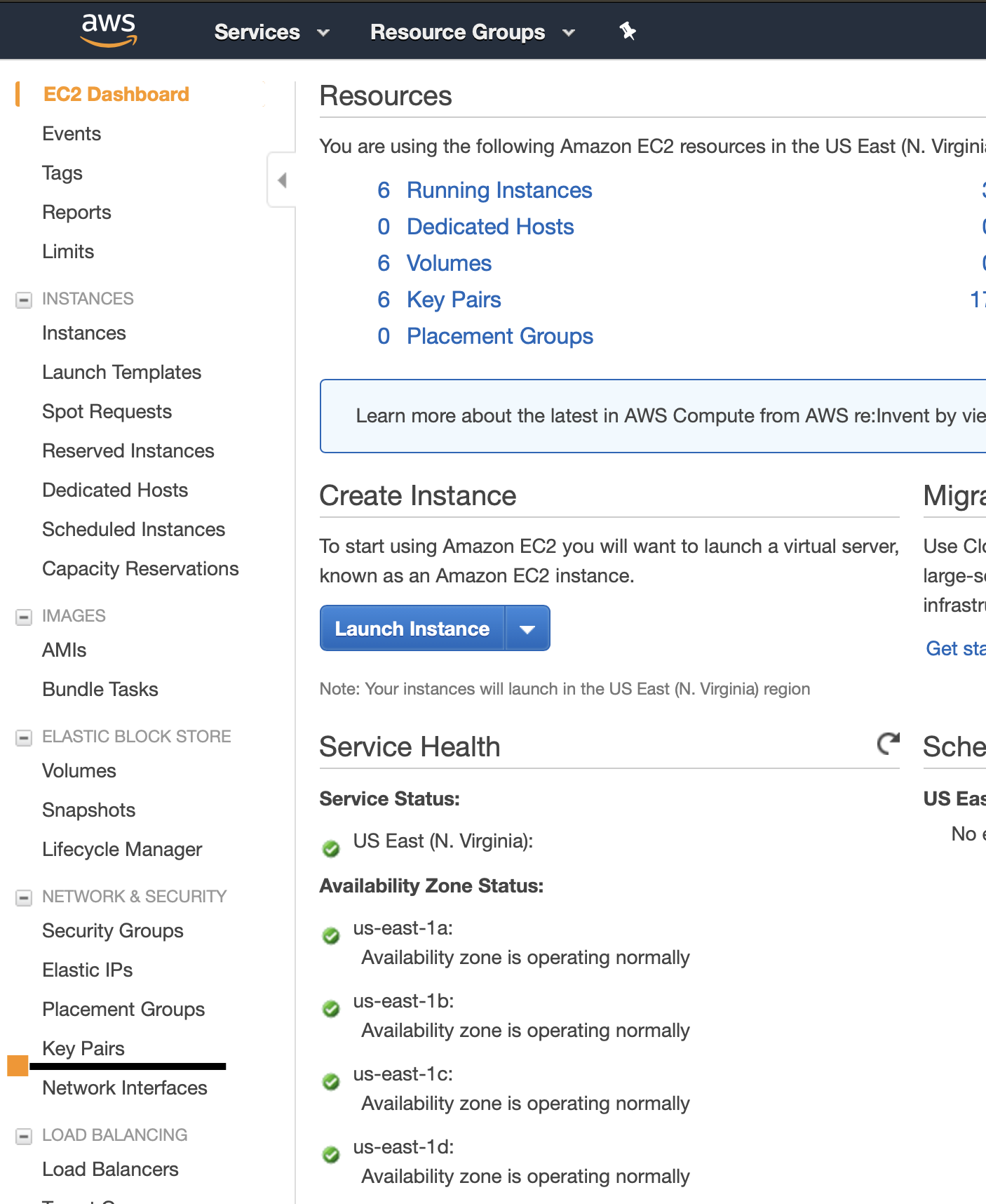
### 0. Setup supporting infrastructure

* If it is not already installed, install **Ansible** on your AWS workstation. Use the –user flag or install into a virtualenv

|  |
| --- |
| pip3 install --user ansible |

* Ensure **Docker** is installed and configured on your AWS Workstation
  + For Amazon workspaces, we need to configure docker to properly perform DNS lookups
    - → Create file */etc/docker/daemon.json* with the following contents

|  |
| --- |
| {  "dns": ["192.10.0.2", "8.8.8.8"]  } |

* Create a Cloud Formation Template defining 3 EC2 instances and their security groups**:**
* Important Note:
  + When creating the EC2 you must specify a key if you ever want to login to it. You must generate that key (PEM file) before running Cloudformation. You need to keep this PEM file safe as you will need it below for configuring Bamboo
  + You can only generate a PEM file using AWS console.
    - Log into the console
    - Goto EC2 services
    - Select Key Pairs and generate one.
* Here are the **EC2 instance** specs:
  + - <team name>-infrastructure
      * Instance type: t2.medium
      * AMI: Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0b4577d77dac11b84 (64-bit x86)
      * Inbound ports: 22, 80, 8080, 443, 8085
        + This instance will host Bamboo and all other supporting services
        + You may host Splunk here if you are not using a Cloud-hosted instance, but Splunk requires a lot of storage space, so you may want to adjust your instance settings to accomodate it.
    - <team name>-staging
      * Instance type: t2.micro
      * AMI: Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0b4577d77dac11b84 (64-bit x86)
      * Inbound Ports: 22, 8088
        + This is the test environment. The pipelines run by Bamboo will use this instance. Commits to Github will trigger a deployment to this environment and run tests.
    - <your name>-prod
      * Instance type: t2.micro
      * AMI: Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0b4577d77dac11b84 (64-bit x86)
      * Inbound ports: 22, 8088
    - NOTE: The AMI ids specified above are specific to the us-east-2 region. You may need a different id if you use a different region or different version of Ubuntu. See also: https://cloud-images.ubuntu.com/locator/ec2/
    - You should be able to ssh to your instances using the 'ubuntu' user and your PEM file as your private key.
* Create an **Ansible** playbook and add the hosts file on your AWS Workstation (as we did in the previous labs) to install docker and docker-compose on each EC2 created above.

"hosts" Inventory file skeleton

|  |
| --- |
| [docker]  [all:vars]  ansible\_connection=ssh  ansible\_user=ubuntu  ansible\_ssh\_private\_key\_file= |

#### **Run** Bamboo **on Infrastructure node**

a) Bamboo can easily be run via docker using the [custom bamboo image](https://hub.docker.com/repository/docker/jrrickerson/capstone-bamboo) created for the capstone. You may use the following commands, or create an appropriate docker-compose file to configure Bamboo. Mounting the “docker.sock” file will allow Bamboo to run Docker commands.

|  |
| --- |
| docker volume create --name bambooVolume  docker run --group-add $(getent group docker | cut -d ":" -f 3) -v /var/run/docker.sock:/var/run/docker.sock -v bambooVolume:<bamboo\_home> --name="bamboo" --init -d -p 54663:54663 -p 8085:8085 jrrickerson/capstone-bamboo |

i) Use or create an Atlassian account to get a trial license for bamboo

ii) Go to Bamboo in your browser to finish installation on your infrastructure node.

[infrastructure-node-address]:8085

d) Setup a container registry

i) Create an account on hub.docker.com or setup an ECR registry in AWS

e) Create capstone Git repo in Github

i) Its content should look similar to the following directory structure by the end of the capstone.

|  |
| --- |
| capstone/  hosts  infra-ansible/  cft.yaml  playbook.yml  vars.ini  monitoring-ansible/  playbook.yml  vars.ini  wordpress/  docker-compose.yml  pyblog/  Dockerfile  requirements.txt  pyblog.py  test\_pyblog.py |

### 1. Create a pipeline to deploy wordpress app

Write a Bamboo pipeline that can be run to deploy the latest Wordpress image. Ensure you can deploy wordpress using your docker-compose.yml before attempting to make the Bamboo plan. Configure Bamboo with EC2 PEM that was generated when you create the EC2 infrastructure instance.

#### a) Write Docker Compose file for Wordpress

i) Have wordpress exposed on 8088

ii) Wordpress data (/var/www/html) should be stored in a docker volume

iii) Ensure you deploy the required mysql db for wordpress

#### b) Bamboo Build Plan Requirements

0) Triggered on any change in Github

i) [Plan Stage] Upload docker-compose.yml to target machine

ii) [Plan Stage] Deploy wordpress

iii) [Plan Stage] Smoke test

* A smoke test could be as simple as sending a request to the http endpoint for wordpress to ensure it has started properly.

#### c) One-time install REST API Authentication plugin for Wordpress

0) Search for a freely available REST API Authentication plugin within Wordpress (NOTE: Avoid the “miniOrange” plugin, as it requires license fees)

i) Install the plugin

ii) Activate / Configure the plugin

### 2. **Python** client for Wordpress

Utilizing the **RESTful API** provided by Wordpress ([docs](https://developer.wordpress.org/rest-api/)), write a python CLI tool to read and write blog posts.

#### a) Read configuration from the following ENV variables

WORDPRESS\_USERNAME

WORDPRESS\_PASSWORD

WORDPRESS\_URL

#### b) Unit tests

* Ensure you unit test as many code paths as you can.
* Include error handling for if the Wordpress API is unavailable or incorrect credentials are provided.

#### c) Containerized

* Your CLI tool should be able to run on any machine running docker via a docker container.
  + NOTE, a docker container cannot access files on the host machine, so "upload -f <filename>" may need to be tweaked. See “CLI Detailed Requirements” below.

### 3. Create a build pipeline for pyblog

Write a build pipeline that will rebuild the docker image for your CLI tool on any new commit to your Github repository. Ensure you have a manual process of each step before attempting to automate.

#### b) Bamboo Build Plan requirements

0) Triggered by code commit

ii) [Plan Stage] Static code analysis (flake8)

Extra credit - code coverage:<https://pypi.org/project/coverage/>

iiI) [Plan Stage] Execute unit tests

iv) [Plan Stage] Build container image

- **<your docker hub repo>**/pyblog:latest

v) [Plan Stage] Run container (as a test) Acceptance tests (Execute commands against wordpress in staging env)

vii) [Plan Stage] Push image to registry

### 4. System and Application Monitoring

#### a) Metric gathering

The health of the system should be monitored, this can be done with Splunk Enterprise or Splunk Cloud.

#### a) Dashboards

Create a simple dashboard monitoring the infrastructure health and application performance.

## Python CLI - Detailed Requirements

### Required Functions

* Show latest blog post
* Upload a blog post

### Example Usage

python3 pyblog.py <command> [options]

python3 pyblog.py read

- Outputs the contents of the latest blog post to stdout

python3 pyblog.py upload -f <filename or - >

- Uploads the contents of the specified file as a new blog post

- Uses the current time for the post

- Format of the post file is as follows

- NOTE: The literal character '-' as the filename signifies reading file contents from stdin instead of opening a file

|  |
| --- |
| <title>  <contents> |

Example file contents:

|  |
| --- |
| Latin is a fun language  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.  Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur? |

# 

# 

# Requirements Matrix

|  |  |  |
| --- | --- | --- |
| **Component** | **Basic Requirements** | **Advanced Ideas** |
| **Infrastructure** | * Created via a Cloud Formation Template or Terraform | * CFT stack has outputs for accessing services * CFT stack created via ansible playbook |
| **Application data** | * MySQL and Wordpress data is stored external from running container on the host | * MySQL and Wordpress persistent data is stored on dedicated EBS volumes |
| **PyBlog client** | * Show latest blog post * Upload a blog post * Unittest coverage > 80% * Dockerized   + docker run pyblog read | * Additional features   + Add comment to blog post   + Edit existing post   + Search for post by keywords |
| **CI/CD** | * Pipeline that deploys wordpress to production * CI Pipeline for testing pyblog and building it's Docker image | * Create a pipeline to provision and configure Staging and Prod servers from scratch * pyblog docker image versioning strategy (:v1, :v2, etc) for "cutting a release" |
| **Development Process** | * Code is peer reviewed before merge * All Code is stored in a centralized Git repo |  |
| **Application Monitoring** | * Wordpress health   + Request health | * Monitoring agent(s) deployed via Ansible * Wordpress logs to Splunk (consider *Splunk Universal Forwarder*) |
| **Infrastructure Monitoring** | * Basic Bamboo and Server metrics tracked | * Monitoring agents deployed via Ansible |