Auditwatch

Spring 2023

Team Members:
Elliston Kieng
Jason Copson
Chun Yu
Sundari Bista
Kevin Hui

Table of Contents

Project Mission

The purpose of the project for IT485 is to demonstrate the ability to integrate a backend language and front end language then present it in a usable way to a consumer. The project will consist of 5 students with unique skill sets that can contribute to the release of our product. The semester-long project consists of software development skills, system administration skill, presentation skill and project management abilities. The 5 students also known as the Auditwatch group will meet weekly to start and end sprints, discuss roadblocks, update reports, and discuss schedule availability. Upon completion, the Auditwatch group will live demonstrate the product with other groups and deliver a report to the stakeholders. The Auditwatch group will consist of the following members:

- Elliston Kieng
  - A senior with strengths in system administration,communication and project management experience.
- Jason Copson
  - Senior student, with strengths in system administration, networks, communication and project management.
- Chun Yu
  - Junior student with strengths in HTML and CSS.
- Sundari Bista
  - Senior student with strengths in HTML and CSS
- Kevin Hui
  - Senior student with strengths in Python, HTML and CSS

Versions

1.0

- Hosted on Amazon AWS EC2 instance.
- Responsive HTML5 code.
- Bar graph charts for query results.
- HTML5 calendar picker.
- Provide attribute value for attribute Key.
- Can limit # of returns from the query.

- Submit a question to the developer.
- Can only query CloudTrail Events.

Project Information

Project Title: Auditwatch

Prepared By: Elliston Kieng, Jason Copson, Chun Yu, Sundari Bista, and Kevin Hui

Program / Project URL: https://www.auditwatch.site

Service Description: Auditwatch can display results from AWS Cloudtrail

Problem Statement

While companies need access to Amazon's AWS Cloudtrail service console. The complexity of AWS console, permissions and training can be overwhelming for some users. Using Auditwatch's web application the users are presented with a simplified interface to query AWs's cloudtrail event logs while not having to login to AWS console. This application also helps the IT department from providing user access to the AWS console while delivering query services to end users.

Mission Statement and Goals

The mission of Auditwatch's web application is to deliver a web application for end users to query information from Amazon's AWS Cloudtrail service without having access to the AWS Console. By not having access to the AWS console means a more secure environment for the IT department, reduced service request for AWS services and less training requirements for using AWS services.

Goals

1. Simple responsive HTML5 webpage compatible on desktop using FireFox.
2. Query AWS Cloudtrail events that include: Username, Event Source, Event Name and Access Key ID.

3. Reduce the need to access the AWS console for CloudTrail Events.

Planning and Scheduling

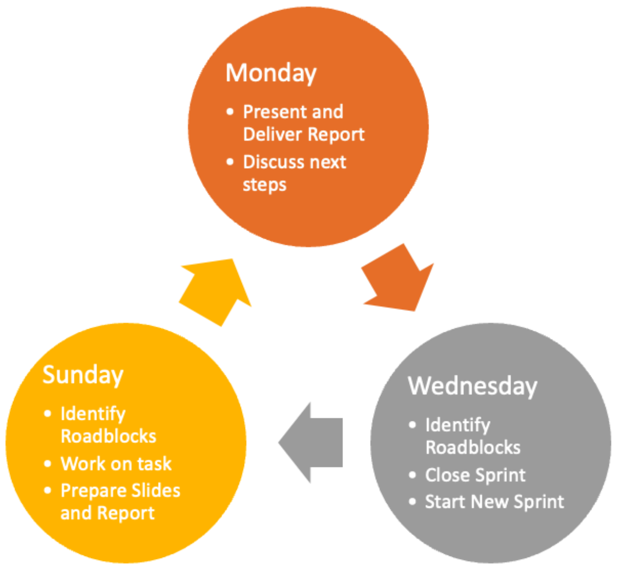
Presentation and Reports

The project started January 23rd 2023 and will finish on May 8th. During that time the Auditwatch group completed 6 in-class presentations and 6 status reports. The class presentations allowed a public presentation of the progress the team was making over time. The reports were provided to the stakeholders for in-depth analysis such as each student's contribution.

Meetings

Throughout the semester the Auditwatch group met every Wednesday and Sunday evenings around 6:30pm EST lasting between 30-60 minutes. The purpose of the Wednesday stand up meeting was to identify road-blocks, complete sprint backlogs, start new sprint and assign tasks to resources. Sunday evenings stand-ups concentrated primarily on developing reports, preparing presentation materials and clearing roadblocks for engineering work to be completed the following Wednesday See example below.

Meeting Cadence



Resource Allocations

Example 1

Task	Resource(s)
Restore main site after AWS shutdown; configure services to persist upon reboots/after shutdowns	Jay, Elliston
AWS Cognito and AWS Identification Access Management now functioning from live auditwatch.site	Chun, Jay
Auditwatch web pages updated	Chun, Sundari
Json data can be displayed from Boto3 to HTML with user input for data parameters	Kevin, Elliston
Cloud9 and AWS Workspace deployed for further development	Kevin, Elliston
General management of project	Kevin, Elliston, Jay

Project Deliverables by Sprints (Kevin)

Helps a company quickly find CloudTrail logs they might want, such as:

- Overview of what happened the last few days
- What did a specific user do between a couple of days
- Look into unusual activity, such as if a user has been failing to log in or failing to access things outside of their assigned privileges

Front End Development: (chun/sundari)

Summary:

- In Auditwatch our main goal is to optimize navigation which helps to make easier for users to find what they want from the site
- After designing most of our frontend is basic HTML and CSS we needed to find a way to implement this into our backend properly.
- We wanted to make the frontend more interactive giving users the ability to see general log data but also more detailed data if needed. With this we used HTML/CSS which allows us to formating/ design and JavaScript that helps to make dynamic changes.
- Amazon Cognito offers users pools and identity pools. We allow our users to login from our web application and access AWS services.
- Cloudtrail Log Search page is also useful for users to retrieve the results by entering attribute values.

Description:

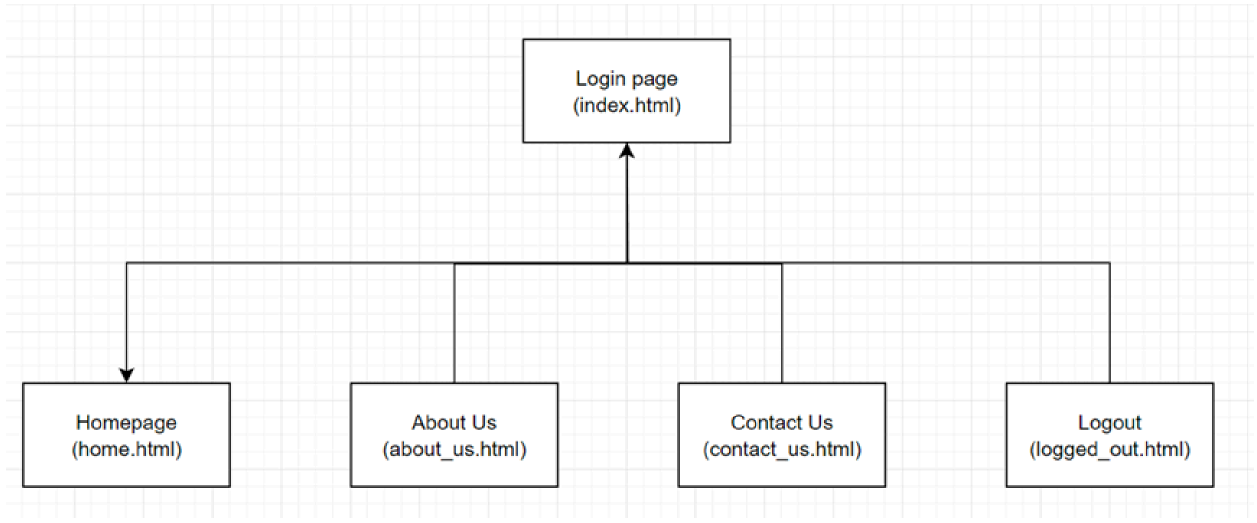
At Auditwatch, we understand the importance of monitoring and analyzing cloud infrastructure data to ensure the security and compliance of your business. That's why we created a powerful tool that captures AWS CloudTrail logs and presents them in an easy-to-use website dashboard and graphs for analytics. With our web application, companies can quickly and easily identify potential security risks and track user activity. We also implement Amazon Cognito which offers users pools, identity pools and access to other AWS services.

Key Features:

- Amazon Cognito integration for secure login and logout
- Retrieval and display of Amazon CloudTrail logs containing users' query information
- Data visualization capabilities
- Contact form for troubleshooting and website improvement purposes

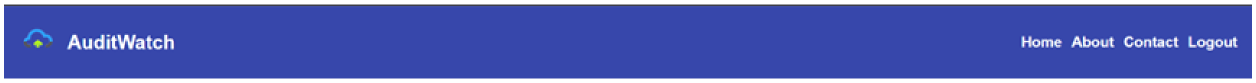
Design and Layout:

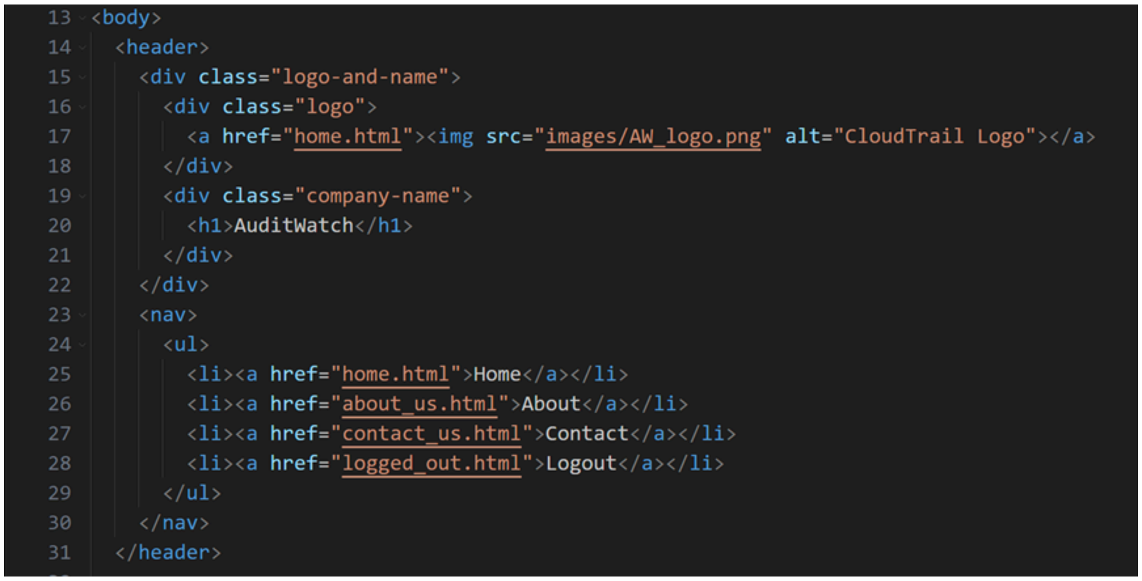
Our website has been designed with simplicity, modernity, and practicality in mind. Navigation has been made intuitive and easy, enabling users to query the information quickly and efficiently by entering attribute values on the homepage. The color scheme features a combination of dark and blue shades, imparting a professional and straightforward look to the website.



Navigation:

The header of our website features a user-friendly navigation bar that facilitates easy access to the desired content. Our top menu comprises various categories such as Home, About Us, Contact Us, and Logout. Additionally, our company icon serves as a link to the homepage, providing users with a convenient way to return to the starting point.

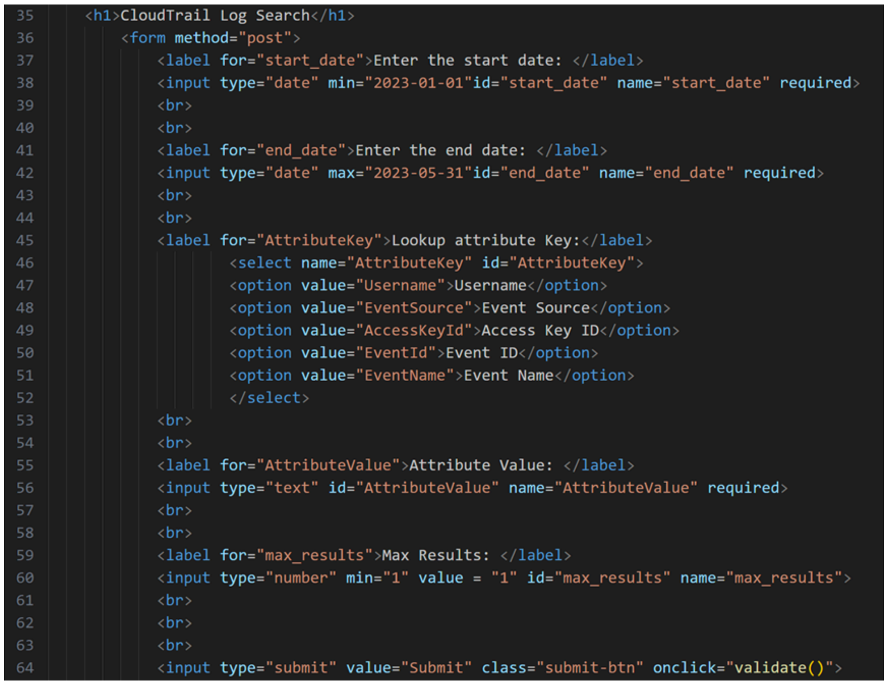




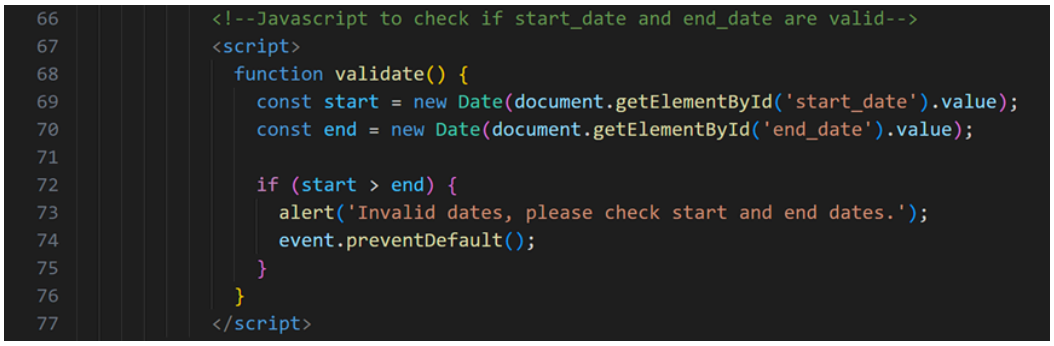
```html
<body>
  <header>
    <div class="logo-and-name">
      <div class="logo">
        <a href="home.html"><img src="images/AW_logo.png" alt="CloudTrail Logo"></a>
      </div>
      <div class="company-name">
        <h1>AuditWatch</h1>
      </div>
    </div>
    <nav>
      <ul>
        <li><a href="home.html">Home</a></li>
        <li><a href="about_us.html">About</a></li>
        <li><a href="contact_us.html">Contact</a></li>
        <li><a href="logged_out.html">Logout</a></li>
      </ul>
    </nav>
  </header>
```

Content:

The primary goal of our web application is to query information from Amazon's AWS CloudTrail service and present it on our website. This form requires users to enter the start date and the end date of the events. Users can also search through different attribute keys such as username, event source, access key ID, event ID and event name to find the information they are looking for. Additionally, this form also implements a JavaScript which will do a validation test to check if the start date and end date are valid or not. If any error is detected, a message prompt will appear to prevent users from proceeding further.



```html
<h1>CloudTrail Log Search</h1>
    <form method="post">
        <label for="start_date">Enter the start date: </label>
        <input type="date" min="2023-01-01"id="start_date" name="start_date" required>
        <br>
        <br>
        <label for="end_date">Enter the end date: </label>
        <input type="date" max="2023-05-31"id="end_date" name="end_date" required>
        <br>
        <br>
        <label for="AttributeKey">Lookup attribute Key:</label>
                <select name="AttributeKey" id="AttributeKey">
                <option value="Username">Username</option>
                <option value="EventSource">Event Source</option>
                <option value="AccessKeyId">Access Key ID</option>
                <option value="EventId">Event ID</option>
                <option value="EventName">Event Name</option>
                </select>
        <br>
        <br>
        <label for="AttributeValue">Attribute Value: </label>
        <input type="text" id="AttributeValue" name="AttributeValue" required>
        <br>
        <br>
        <label for="max_results">Max Results: </label>
        <input type="number" min="1" value = "1" id="max_results" name="max_results">
        <br>
        <br>
        <br>
        <input type="submit" value="Submit" class="submit-btn" onclick="validate()">
```

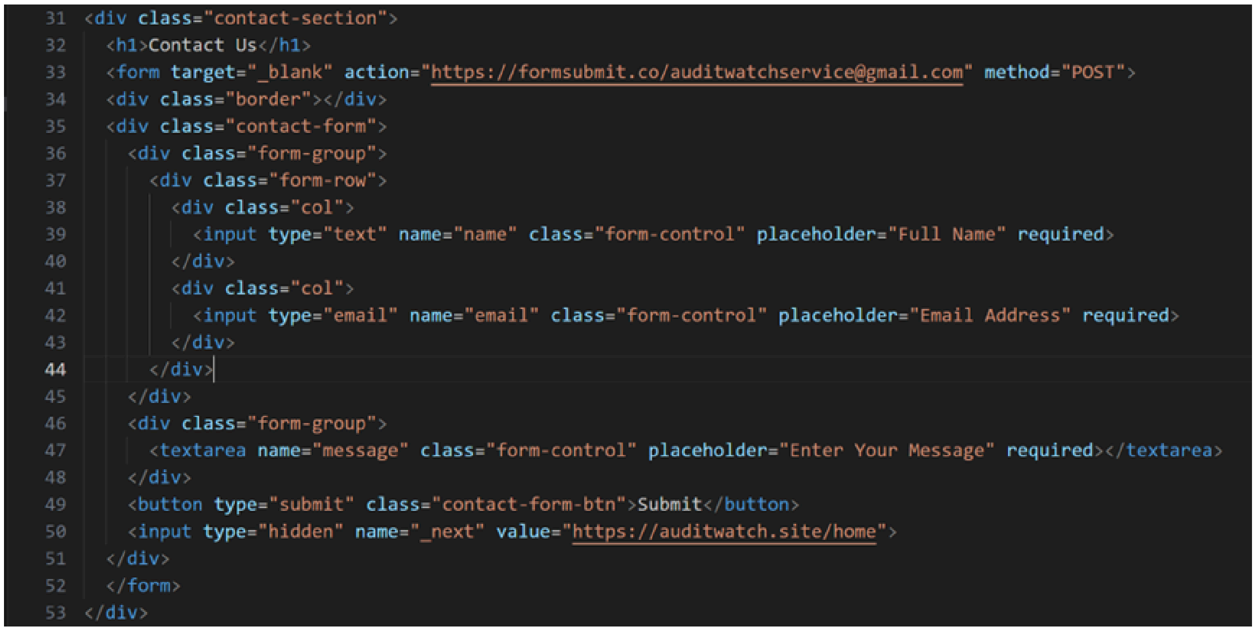


```html
<!--Javascript to check if start_date and end_date are valid-->
<script>
  function validate() {
    const start = new Date(document.getElementById('start_date').value);
    const end = new Date(document.getElementById('end_date').value);

    if (start > end) {
      alert('Invalid dates, please check start and end dates.');
      event.preventDefault();
    }
  }
</script>
```

Contact Information:

In order to provide efficient assistance to our users and continuously enhance our products based on their valuable feedback, we have integrated an online service called "FormSubmit" into our website. This feature allows users to complete and submit a form that is automatically sent to our business email for review. “FormSubmit” can not only be used as a ticket function for users to report problems or difficulties while using our product. It can also act as a platform for our company to receive feedback from our users and improve from it.



```html
<div class="contact-section">
  <h1>Contact Us</h1>
  <form target="_blank" action="https://formsubmit.co/auditwatchservice@gmail.com" method="POST">
  <div class="border"></div>
  <div class="contact-form">
    <div class="form-group">
      <div class="form-row">
        <div class="col">
          <input type="text" name="name" class="form-control" placeholder="Full Name" required>
        </div>
        <div class="col">
          <input type="email" name="email" class="form-control" placeholder="Email Address" required>
        </div>
      </div>
    </div>
    <div class="form-group">
      <textarea name="message" class="form-control" placeholder="Enter Your Message" required></textarea>
    </div>
    <button type="submit" class="contact-form-btn">Submit</button>
    <input type="hidden" name="_next" value="https://auditwatch.site/home">
  </div>
  </form>
</div>
```

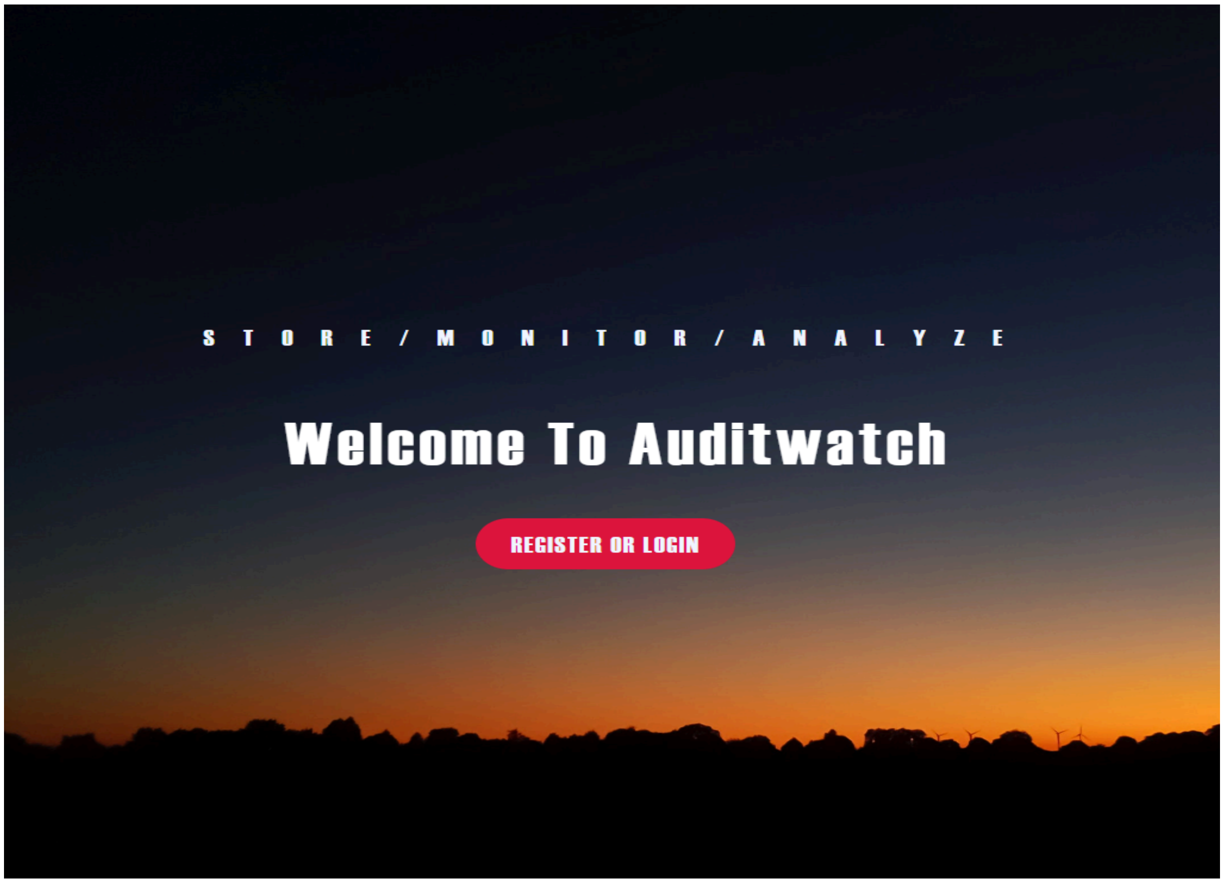
The <form action> attribute directs the submission of our form to the designated URL, allowing us to receive submissions to our business email. To facilitate future communication and feedback from users, we have included three required input fields: Full Name, Email Address, and Message. After submitting the form, users will be redirected to the home page, ensuring a smooth and seamless user experience.

CSS Code:

With the use of CSS and HTML in Auditwatch we are able to shine the front end. As a Web Developer we are familiar with CSS and HTML. Our main goal for using CSS and HTML is to make Auditwatch looks well organized, structured and interactive. CSS helps to make a website beautiful with sizing, decoration and transformation while HTML prepares the structure of the page.

Key Features:

- Successfully created the welcome page with a beautiful background image and login page which allows users to access and interact with our site.
- Fast, secure and easy for the user.



CSS and Frontend:

With the help of HTML and CSS we are able to create frontend of Auditwatch that include welcome page, header, and CloudTrail Log Search Form. To make the simple front end we use all the CSS features (margin, padding, font, text-decoration, position, etc.).

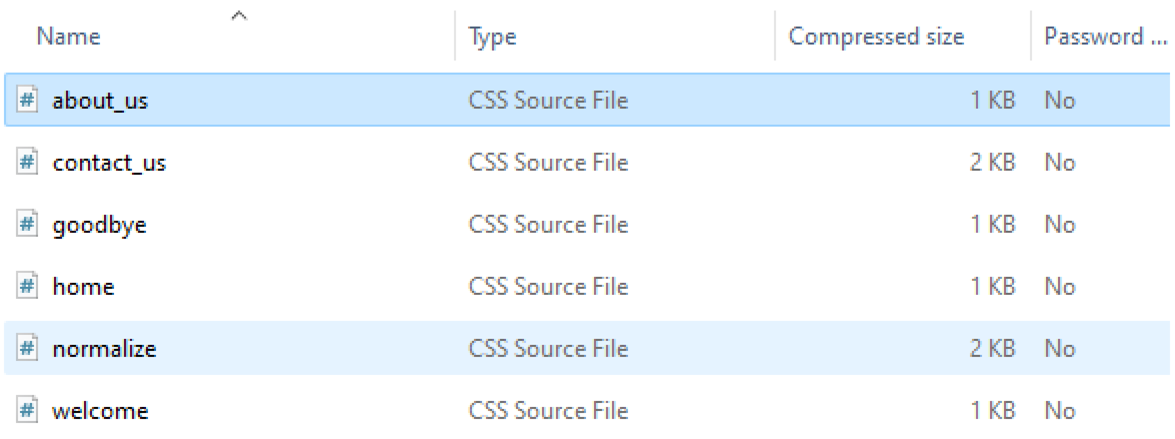
CSS Features:

- Start set up the css to give the margin and padding value of all default html elements.
- Set the background image to the body.
- By using the background property and url method pass the images inside the method.
- For the “WELCOME PAGE” we set the background image for the whole cover so that image can occupy the whole cover page.

- The sticky class is added to the header with JS when it reaches its scroll position.
- Add some top padding to the page content to prevent sudden quick movement.

CSS Codes:

Here is the image of CSS codes that we use for Auditwatch:



Name	Type	Compressed size	Password ...
about_us	CSS Source File	1 KB	No
contact_us	CSS Source File	2 KB	No
goodbye	CSS Source File	1 KB	No
home	CSS Source File	1 KB	No
normalize	CSS Source File	2 KB	No
welcome	CSS Source File	1 KB	No

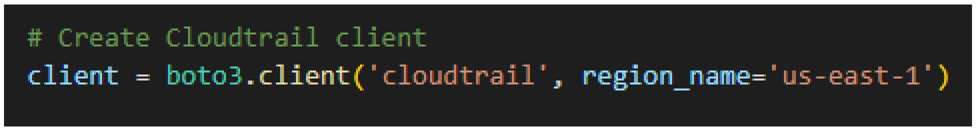
Back End Development: (KEVIN)

Summary:

- For the back end development, we chose to use the Python programming language as that was the language that the group had the most experience in and was incredibly adaptable with its wide array of modules and libraries we could use in our application.
- By combining multiple python libraries together, we are able to create a working application for users to retrieve, parse, and display AWS CloudTrail logs onto a webpage in an effective manner.

AWS Boto3 Library:

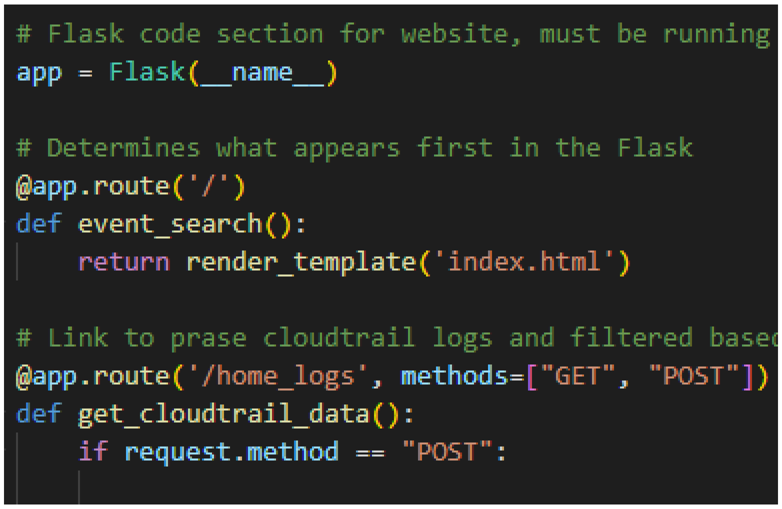
By using the boto3 library, provided by Amazon Web Services (AWS), our application is able to communicate with AWS CloudTrail services much simpler than through other means. We can then read the CloudTrail logs that are generated by AWS and the details of those events, such as the event name, the user that created it, the time it occurred, and the Access Key ID of the user's account. From this we are able to work with CloudTrail logs in our python program. We are using the paginate method with LookupAttributes in order to search for the desired logs with StartTime, EndTime, and PaginationConfig as arguments. Below is assigning the boto3.client for cloudtrail to variable 'client' for the program.



```
# Create Cloudtrail client
client = boto3.client('cloudtrail', region_name='us-east-1')
```

Flask Web Framework:

Flask is then used to generate a webpage that allows a user to input their values to be used in our python program to retrieve CloudTrail logs, such as the start and end dates, attribute values, and the maximum number of logs to retrieve. As well as, setting up the links and pages that appear when the website is accessed, such as the index page, home_logs, about us, contact us, and logout pages. Below shows the home page and the home_logs page for user input.



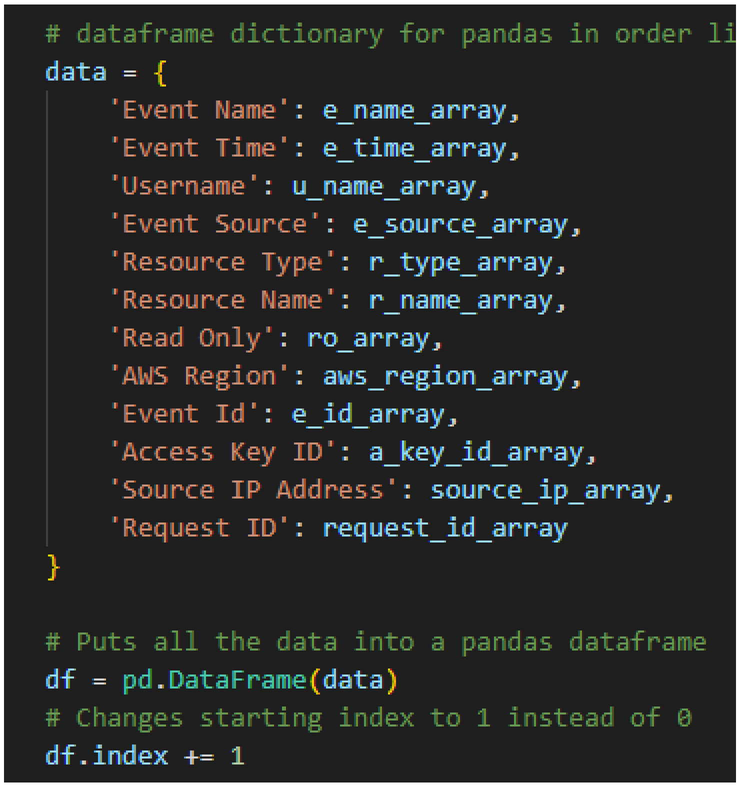
```
# Flask code section for website, must be running
app = Flask(__name__)

# Determines what appears first in the Flask
@app.route('/')
def event_search():
    return render_template('index.html')

# Link to prase cloudtrail logs and filtered basec
@app.route('/home_logs', methods=["GET", "POST"])
def get_cloudtrail_data():
    if request.method == "POST":
```

Pandas library:

The Pandas library is used for data analysis and here it is used to convert the data from the CloudTrail logs into a workable DataFrame, a two-dimensional table with rows and columns, to be displayed as a table on our webpage. As shown below, for our purposes our rows are the number of times a particular event is created, and for our columns we are having them be the usernames of the user who created those events. We also had the index start from 1 instead of the normal 0 index that is set by default.

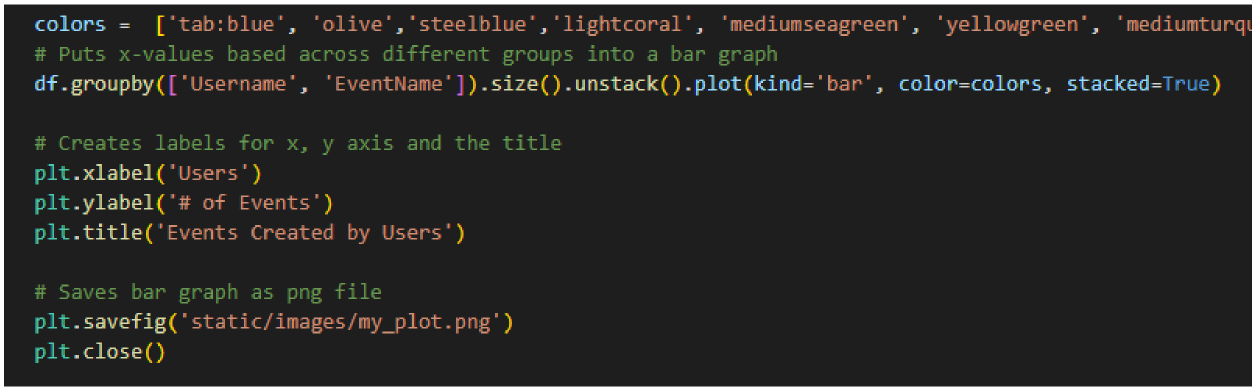


```
# dataframe dictionary for pandas in order li
data = {
    'Event Name': e_name_array,
    'Event Time': e_time_array,
    'Username': u_name_array,
    'Event Source': e_source_array,
    'Resource Type': r_type_array,
    'Resource Name': r_name_array,
    'Read Only': ro_array,
    'AWS Region': aws_region_array,
    'Event Id': e_id_array,
    'Access Key ID': a_key_id_array,
    'Source IP Address': source_ip_array,
    'Request ID': request_id_array
}

# Puts all the data into a pandas dataframe
df = pd.DataFrame(data)
# Changes starting index to 1 instead of 0
df.index += 1
```

Matplotlib library:

Matplotlib is a plotting library for python that allows us to create graphs and help visualize our data in a more visually pleasing way. For our purposes, Matplotlib takes the Pandas DataFrame we created earlier to plot a bar graph based on attributes queried by the user. As shown below, the X-axis is the usernames that had created events and the Y-axis will be the number of events that those users created within that period of time. The legend describes the name of the event and its corresponding color.



```
colors =  ['tab:blue', 'olive','steelblue','lightcoral', 'mediumseagreen', 'yellowgreen', 'mediumturq
# Puts x-values based across different groups into a bar graph
df.groupby(['Username', 'EventName']).size().unstack().plot(kind='bar', color=colors, stacked=True)

# Creates labels for x, y axis and the title
plt.xlabel('Users')
plt.ylabel('# of Events')
plt.title('Events Created by Users')

# Saves bar graph as png file
plt.savefig('static/images/my_plot.png')
plt.close()
```

Infrastructure System Design (Jason)

System Requirements

From the outset of the project, we wanted to build our application on fully production-ready infrastructure, live, and in the cloud. We found this could be accomplished at very little cost. Granted, the resources provisioned for the current setup would need to be dialed up for a true production environment, but all of the components chosen could be readily and easily scaled.

To run the web and application server, along with the Python code itself, we selected a small t2.micro instance on AWS EC2. This provided only a single CPU core and 1GB RAM, but also provided for burstable usage when needed, which made for an ideal fit both in terms of cost and our transient demand for greater resources.

Git repository

One of the first services we established was a git repository. All team members set up accounts at Github, which is a well-known cloud service that hosts git repositories (free to use for small teams). All members then installed a local copy of the repo on their home computers, meaning the root path of the repository contained a ".git" file. The repo tracked all changes to files made over time, so that everyone could be assured of having the latest copy of the code (python, HTML, static files, template files, as well as Apache and Gunicorn configuration files). For instance, where a change was made to CSS, another user of the HTML file affected could retrieve it easily through the repo instead of needing to request a copy via email. When a change needed to be reverted, the repo also made it easy to fall back to the previous working copy. Finally, by checking out branches off the main repo, a developer could build on an existing file and test it before merging the new code back into the main source.

Team members used both the git UI and the command line interface to work with the repository.

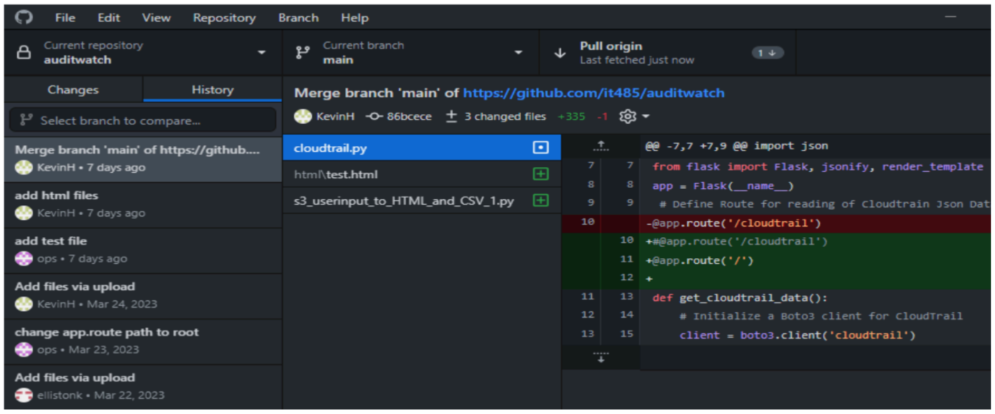


Fig. S1, The git graphical user interface

Merge branch 'main' of https://github....
KevinH • 7 days ago
add html files
KevinH • 7 days ago
add test file
ops • 7 days ago
Add files via upload
KevinH • Mar 24, 2023
change app.route path to root
ops • Mar 23, 2023
Add files via upload
ellistonk • Mar 22, 2023

Merge branch 'main' of https://github.com/it485/auditwatch
KevinH 86bcece 3 changed files +335 -1

cloudtrail.py
html\test.html
s3_userinput_to_HTML_and_CSV_1.py

```
@@ -7,7 +7,9 @@ import json
 from flask import Flask, jsonify, render_template
 app = Flask(__name__)
  # Define Route for reading of Cloudtrain Json Dat
-@app.route('/cloudtrail')
+#@app.route('/cloudtrail')
+@app.route('/')
+
 def get_cloudtrail_data():
     # Initialize a Boto3 client for CloudTrail
     client = boto3.client('cloudtrail')
```

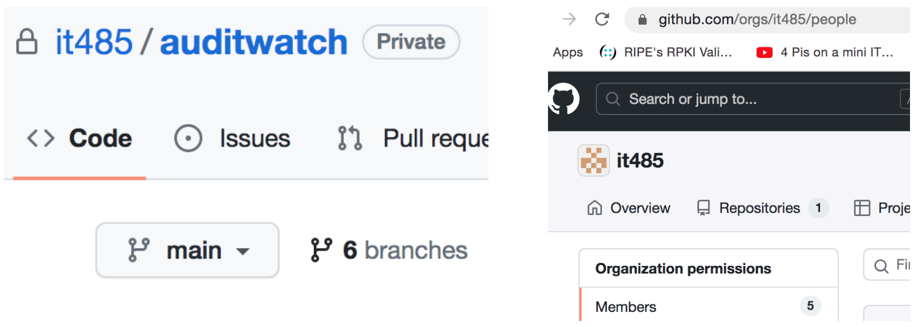


Fig. S2, Our github.com/it485/auditwatch code repository

Domain, DNS, and SSL

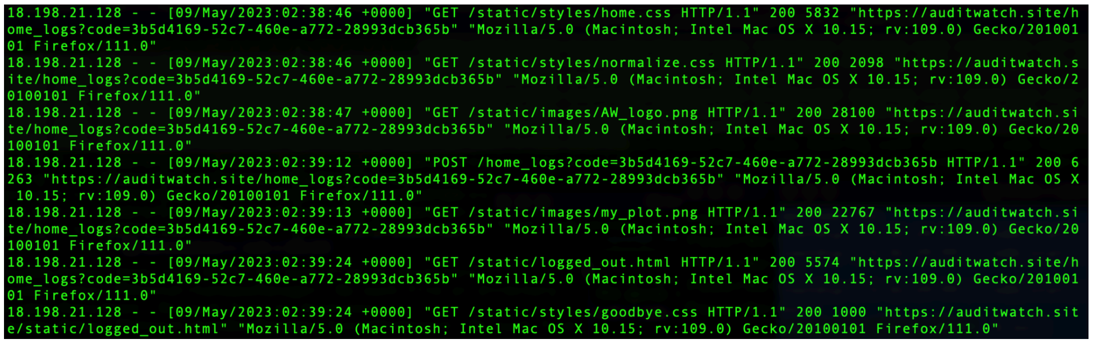
With our public IPv4 address allocation from AWS EC2, we registered a domain “auditwatch.site” and set up DNS records to resolve this domain (and its www subdomain) to the EC2 instance.

As the vast majority of reputable websites now use some version of SSL/TLS for public access, we wanted to similarly provide such security. HTTPS-enabled websites also are important for SEO rankings and gaining the trust of site visitors; over 85% of web users avoid insecure websites.[1] SSL certificates can vary greatly in annual fees, but simple domain-validated certs can run as low as dollars per month. A free alternative to purchasing a certificate is to obtain one via Let’s Encrypt, which is quickly capturing a growing share of the installed certificate market–up to 24% by recent estimates.[2] With budget in mind, we chose Let’s Encrypt. One disadvantage of the Let’s Encrypt certs, is that they require renewal every 60 days (although this also adds to their security appeal). However, an open source “certbot” tool is available that provides the ability to autorenew these which can be installed on any server via command line access and root privileges.[3] This tool removed much of the initial hesitation heretofore often associated with Let’s Encrypt.

Apache Web Server

Apache is a public-facing web server. It can serve many web pages and web applications simultaneously, across as many domain names as needed, while also providing SSL offloading. Apache provides a number of add-on modules that provide for optimum page delivery times, including caching features, HTTP/2, URL rewriting and redirects, security, and extensive logging. The access and error logs provided by Apache were used extensively in troubleshooting this project.

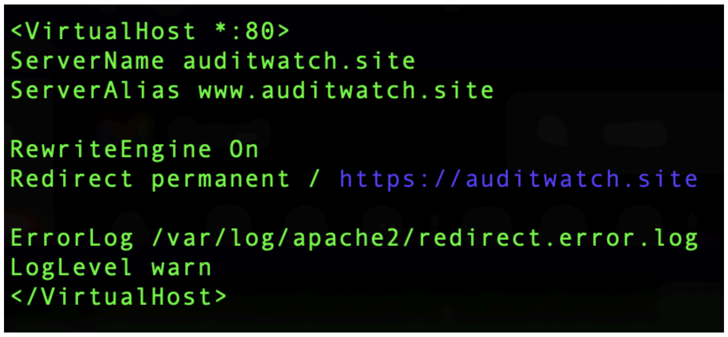
Fig. S3, Successful HTTP 200 responses shown in the Apache access log



```
18.198.21.128 - - [09/May/2023:02:38:46 +0000] "GET /static/styles/home.css HTTP/1.1" 200 5832 "https://auditwatch.site/h
ome_logs?code=3b5d4169-52c7-460e-a772-28993dcb365b" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/201001
01 Firefox/111.0"
18.198.21.128 - - [09/May/2023:02:38:46 +0000] "GET /static/styles/normalize.css HTTP/1.1" 200 2098 "https://auditwatch.s
ite/home_logs?code=3b5d4169-52c7-460e-a772-28993dcb365b" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/2
0100101 Firefox/111.0"
18.198.21.128 - - [09/May/2023:02:38:47 +0000] "GET /static/images/AW_logo.png HTTP/1.1" 200 28100 "https://auditwatch.si
te/home_logs?code=3b5d4169-52c7-460e-a772-28993dcb365b" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20
100101 Firefox/111.0"
18.198.21.128 - - [09/May/2023:02:39:12 +0000] "POST /home_logs?code=3b5d4169-52c7-460e-a772-28993dcb365b HTTP/1.1" 200 6
263 "https://auditwatch.site/home_logs?code=3b5d4169-52c7-460e-a772-28993dcb365b" "Mozilla/5.0 (Macintosh; Intel Mac OS X
 10.15; rv:109.0) Gecko/20100101 Firefox/111.0"
18.198.21.128 - - [09/May/2023:02:39:13 +0000] "GET /static/images/my_plot.png HTTP/1.1" 200 22767 "https://auditwatch.si
te/home_logs?code=3b5d4169-52c7-460e-a772-28993dcb365b" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20
100101 Firefox/111.0"
18.198.21.128 - - [09/May/2023:02:39:24 +0000] "GET /static/logged_out.html HTTP/1.1" 200 5574 "https://auditwatch.site/h
ome_logs?code=3b5d4169-52c7-460e-a772-28993dcb365b" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/201001
01 Firefox/111.0"
18.198.21.128 - - [09/May/2023:02:39:24 +0000] "GET /static/styles/goodbye.css HTTP/1.1" 200 1000 "https://auditwatch.sit
e/static/logged_out.html" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101 Firefox/111.0"
```

Our Apache server listens on both ports 80 and 443. Any traffic coming in for auditwatch.site or www.auditwatch.site on port 80 (plain HTTP) will be redirected to port 443 (HTTPS). This forced redirect of HTTP to HTTPS is as widespread as the use of the SSL certificates themselves. We chose not to force redirect or rewrite the URL for www.auditwatch.site to the apex domain auditwatch.site (or vice-versa) which is another common redirect found with popular websites, but upon login to the site, all subsequent pages loaded do fall under the apex domain.

Fig. S4, Apache virtual host file forcing HTTP to HTTPS



```
<VirtualHost *:80>
ServerName auditwatch.site
ServerAlias www.auditwatch.site

RewriteEngine On
Redirect permanent / https://auditwatch.site

ErrorLog /var/log/apache2/redirect.error.log
LogLevel warn
</VirtualHost>
```

In order to serve the site and application over HTTPS, a certificate and corresponding private key must be installed in the Apache configuration tree. This is where our Let's Encrypt cert and key were accordingly installed.

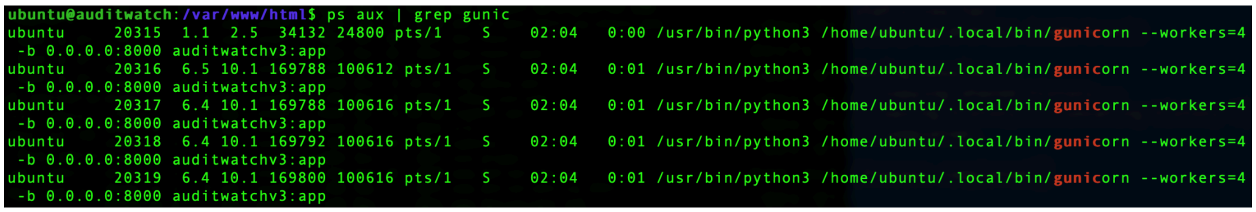
Gunicorn Application Server

This project employs Python running with the Flask framework to facilitate web delivery. The purpose of Flask however is to serve as an add-on module to Python; it serves as a gateway between the application and an application server using a standard interface protocol called the "Web Server Gateway Interface" (WSGI).

While you can run an application in Flask and connect to it via browser on your local machine on port 5000–which is handy for coding and testing purposes–that's as far as Flask should be used for any kind of server. It's not meant for production use.[4] Flask can only handle one request at a time, and will drop the Python interpreter from memory once that session is complete. There's also no security provided by Flask; on the contrary, for debugging purposes, Flask will dump to a privileged shell when hitting an error in the application code. Finally, Flask is inefficient about delivering static resources (e.g., images and binaries, stylesheets/CSS, and javascript).

Enter Gunicorn, an application server built for web applications written in Python. Gunicorn picks up right where Flask leaves off. The app server loads the Python interpreter and application into memory, and can scale copies of the running code by means of "workers". Essentially, Gunicorn forks clones of itself into as many child processes as requested. Of course, this is limited by the amount of CPU and RAM resources available, as well as disk I/O and max file handles, but the server can handle a great number of concurrent requests/second at scale.

Fig. S5, Multiple Gunicorn worker processes, as set at startup

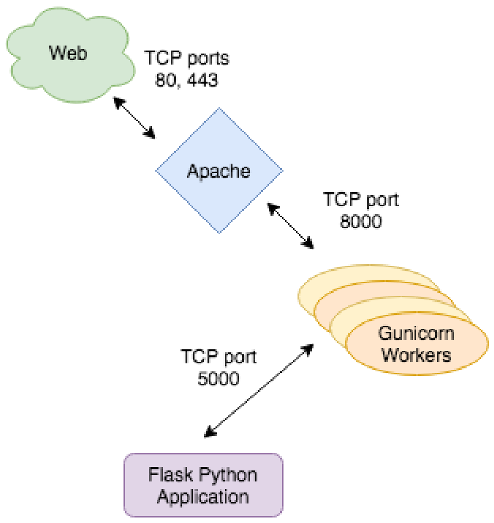


```
ubuntu@auditwatch:/var/www/html$ ps aux | grep gunic
ubuntu     20315  1.1  2.5  34132 24800 pts/1    S    02:04   0:00 /usr/bin/python3 /home/ubuntu/.local/bin/gunicorn --workers=4
 -b 0.0.0.0:8000 auditwatchv3:app
ubuntu     20316  6.5 10.1 169788 100612 pts/1   S    02:04   0:01 /usr/bin/python3 /home/ubuntu/.local/bin/gunicorn --workers=4
 -b 0.0.0.0:8000 auditwatchv3:app
ubuntu     20317  6.4 10.1 169788 100616 pts/1   S    02:04   0:01 /usr/bin/python3 /home/ubuntu/.local/bin/gunicorn --workers=4
 -b 0.0.0.0:8000 auditwatchv3:app
ubuntu     20318  6.4 10.1 169792 100616 pts/1   S    02:04   0:01 /usr/bin/python3 /home/ubuntu/.local/bin/gunicorn --workers=4
 -b 0.0.0.0:8000 auditwatchv3:app
ubuntu     20319  6.4 10.1 169800 100616 pts/1   S    02:04   0:01 /usr/bin/python3 /home/ubuntu/.local/bin/gunicorn --workers=4
 -b 0.0.0.0:8000 auditwatchv3:app
```

So Gunicorn takes the Python app much closer to production web delivery, but leaves the final step to an actual web server, such as nginx or Apache. While Python has become the language of choice for many developers, the concept of using an application server like Gunicorn isn't new; if we had built the application in Java, we would have simply used Tomcat instead–the application server that powers Java applications on the web, like Gunicorn does for Python.

Put all together, the top-level view of the system architecture is best graphically shown here:

Fig. S6, High level architecture view

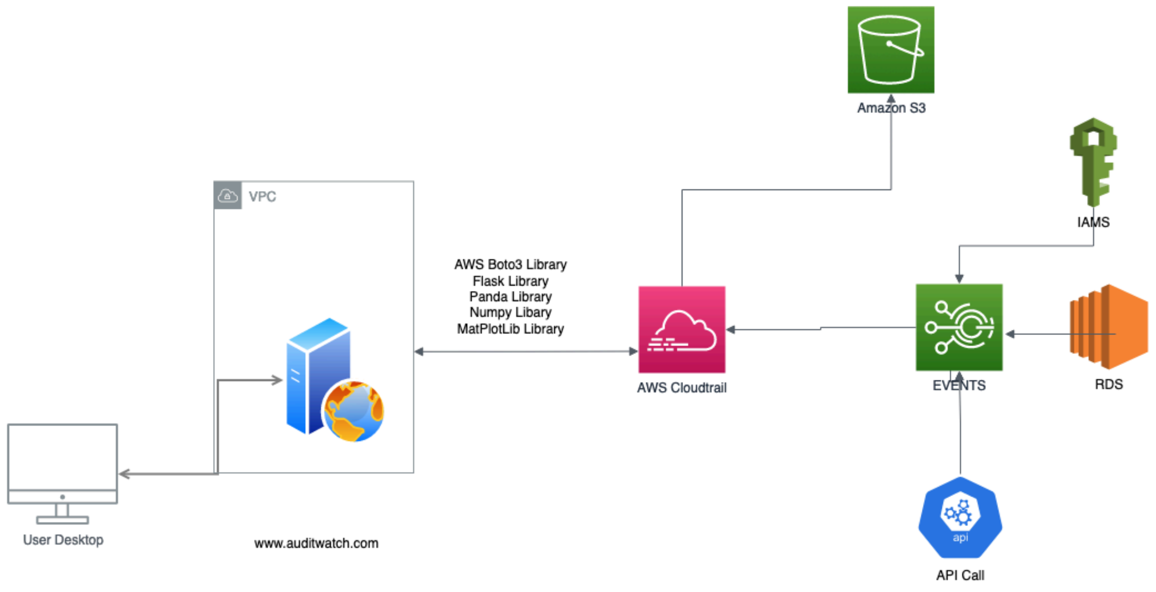


Application Architecture:

When the user enters the URL for our application on auditwatch.site, they are greeted by the index page that prompts them to login with their account. After being authenticated, the user is then brought to the search function of our application, with links in the top right allowing them to view our About Us page, containing our mission statement, and a Contact Us page that allows a user to email us with questions and/or comments about the application.

When the user wants to search their CloudTrail logs, going to the Home page allows them to choose their desired attributes for what logs they would like to retrieve, including the time period of those logs, the attribute keys and values those logs have, and how many they wish to retrieve. If the values are valid then the page loads a table containing the logs that fit the filters from the previous page into the format of a table using columns that contains the log number, Event Name, Event Time, Username, Event Source, Resource Type, Resource Name, if it was Read Only, AWS Region, Event ID, Access Key ID, Source IP Address, and the Request ID.

Below is a diagram of the application architecture. (Elliston)



Appendices

System Deployment Checklist

- ☑ ~~Does website work https://www.auditwatch.site~~
- ☑ ~~Does Register or Login Work~~
- ☑ ~~Does Sign in work~~
- ☑ ~~Does Sign Up~~
- ☑ ~~Does Start Date and End work?~~
- ☑ ~~Does date validation date work?~~
- ☑ ~~Does LookupEvents Work?~~
- ☑ ~~Does the chart work?~~
- ☑ ~~Does Username Work?~~
- ☑ ~~Does EventSource work?~~
- ☑ ~~Does Access Key Work?~~
- ☑ ~~Does Event ID work?~~
- ☑ ~~Does About link work?~~
- ☑ ~~Does Contact work?~~
- ☑ ~~Does contact information send the data?~~
- ☑ ~~Does logged out work?~~
- ☑ ~~Does Home Icon Work~~

References:

AWS. 2023. Boto3 1.26.131 documentation. https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

Matplotlib. February, 2023. Matplotlib 3.7.0 documentation. https://matplotlib.org/

Pandas. April 24, 2023. Pandas 2.0.1 documentation. https://pandas.pydata.org/

NumPy. April 22, 2023. NumPy 1.24.3 documentation. https://numpy.org/

Flask. April 25, 2023. Flask 2.3.0 documentation. https://flask.palletsprojects.com/en/2.3.x/

Footnotes:

[1] https://www.enterpriseappstoday.com/stats/ssl-statistics.html

[2] https://www.clickssl.net/blog/ssl-statistics

[3] https://certbot.eff.org/pages/about

[4] https://flask.palletsprojects.com/en/2.3.x/tutorial/deploy/