

Pixplore – Group 1

Amaan Ahmed – 25100127

Muhammad Shayan – 25100164

Faizan Ullah – 23030015

Ibrahim Ahmed Khan – 25100112

Abdul Ahad Bin Ali - 25100016

Amaan – Lambda, Documentation, Slides, SCRUM Master

Shayan – API Gateway, Cognito, Frontend, Integration

Faizan – ECR, ECS, Cognito, Load Balancer, Autoscaling, FastAPI, Rekognition, Integration

Ibrahim – SQS, EventBridge, Integration

Ahad – Testing, S3 Archive and CDN, Integration

Collective effort for integration and functionality of all modules.

Tasks allocated and tracked through Trello.

Table of Contents

1. Introduction
2. Functional Scope
3. Cloud Design Scope
4. Components
5. Architecture Diagram
6. AWS Services
7. Design Excellence
8. Load Testing
9. Disaster Recovery
10. Cost Calculator
11. User Guide

Introduction

Pixplore is a Pinterest-like image gallery application designed to provide users with a seamless way to upload and search for images based on tags or visual content. It leverages AWS cloud services to ensure a serverless, scalable, and secure environment that can handle high traffic and provide efficient, on-demand search functionality. The platform aims to cater to dynamic user demands and supports scalability for variable traffic, making it ideal for both individual and business use cases. For businesses, Pixplore offers practical applications like efficient organization of extensive product catalogs, quick retrieval and advanced content management and analysis. The goal is to create a system that's both powerful and user friendly, solving everyday challenges in digital image management efficiently.

Functional Scope

Pixplore offers a wide array of features designed to cater to both the requirements of individuals and businesses. For an individual user, we offer:

1. **User Authentication & Account Management:**

Secure sign-up and login using AWS Cognito, ensuring that users can safely create and manage their accounts. This feature supports password recovery and user verification to maintain data security.

2. **Image Upload and Storage:**

Users can upload images securely to the cloud using Amazon S3. The upload process is streamlined with the use of pre-signed URLs, which allows direct uploads from the user's device to the cloud without overloading the backend server.

3. **Metadata Extraction, Image Tagging and Search Functionality:**

Once images are uploaded, Pixplore automatically extracts metadata (such as image labels, objects, and tags) using AWS Rekognition. This metadata is stored in Amazon Aurora and enables more complex searches for the users. Users can search for images based on tags, labels, or even visual content, thanks to the image analysis provided by Rekognition. The system uses advanced querying capabilities on the metadata stored in Aurora to deliver fast and accurate search results.

4. **Scalability and Performance:**

Pixplore is designed to scale seamlessly based on traffic and user demand. The backend automatically adjusts to handle increasing loads, ensuring that users always experience fast and responsive performance.

Similarly, for businesses, we offer:

1. Advanced Content Management:

Businesses can use Pixplore to organize and manage large image libraries, for example product catalogs. The ability to automatically tag and categorize images based on their content helps businesses conserve time, resources, and improve workflow efficiency.

2. Image Processing:

Automated extraction of image metadata and real-time image analysis using Rekognition provides businesses with valuable insights into the content of their image collections. This feature can be especially beneficial for marketing campaigns or product catalogs that need to quickly filter and search for specific images.

3. Customizable Search for Business Needs:

Pixplore allows businesses to tailor the search functionality to fit their specific use cases, such as filtering product images by tags, categories, or other metadata. This is particularly useful for e-commerce platforms or digital asset management systems.

4. Scalable Infrastructure for Growing Needs:

As businesses scale, Pixplore is capable of handling increasing volumes of images and users. The system's cloud-based, serverless architecture ensures that it can support high traffic and large image volumes without compromising performance or security.

Cloud Design Scope

Pixplore's design is based on a serverless architecture, utilizing multiple AWS services to build a flexible and high-performance solution that can dynamically adjust to varying user loads. Some key principles of our cloud design are:

1. Serverless Architecture

Pixplore leverages AWS's serverless offerings such as Lambda and API Gateway to minimize operational overhead, reduce costs and improve scalability. This architecture ensures that the system can automatically scale based on user demand without the need for manual intervention or resource management. Lambda handles backend processing tasks like image metadata extraction and search queries, while API Gateway serves as the interface for handling incoming requests.

2. Data Storage and Management

Amazon S3 is used for storing uploaded images, providing a secure, scalable, and cost-effective solution for large volumes of data. Pixplore employs Amazon Rekognition that is utilised for analyzing images and extracting metadata such as labels, objects, and text, which is then stored in Aurora for easy search and retrieval. Amazon Aurora then stores metadata extracted from the uploaded images, offering a highly available relational database that can scale as metadata increases.

3. Networking and Isolation

Pixplore utilises Amazon Virtual Private Cloud (VPC) to ensure network isolation and secure communication between services. The VPC design includes **public subnets** for resources that require internet access, such as API Gateway and static web pages, and **private subnets** for sensitive resources like Aurora, ECS with Fargate, and Lambda, which do not require direct internet access. Security Groups and Network Access Control Lists (NACLs) are configured within the VPC to control inbound and outbound traffic to services, ensuring that only authorized requests can interact with sensitive parts of the system.

4. Autoscaling and Load Balancing

The system is designed to scale automatically with varying levels of traffic. An Elastic Load Balancer (Application Load Balancer) is used to distribute incoming requests evenly across backend resources (ECS containers and Lambda functions). This ensures optimal performance and availability, even during traffic spikes. Moreover, Autoscaling adjusts the number of ECS tasks and Lambda functions based on demand. For example, when user traffic increases, ECS containers scale up to meet the demand for image searches and metadata processing.

5. Event-driven Architecture

Pixplore employs an event-driven architecture using Amazon EventBridge and Amazon SQS. EventBridge routes events from services like S3 (when images are uploaded) to trigger relevant workflows, such as metadata extraction using AWS Rekognition. SQS is used to decouple tasks, allowing asynchronous processing of image metadata and preventing bottlenecks during high traffic periods.

6. Monitoring and Observability

Amazon CloudWatch is used to monitor the health, performance, and resource usage of the system. CloudWatch Logs, Metrics, and Alarms are configured to track key performance indicators (KPIs), such as latency, error rates, and resource utilization. Enables proactive scaling and troubleshooting to maintain satisfied user experience and optimal performance of the system. Custom metrics in CloudWatch help the

development team identify potential bottlenecks, such as slow image processing times or throttling issues, and adjust the system's scaling rules accordingly.

7. Security and Compliance

Security is a key design consideration for Pixplore, particularly with user data and images. AWS Cognito is used to manage user authentication, registration, and access control. It ensures that only authorized users can upload images or perform searches. Further, IAM roles and policies restrict access to specific AWS resources based on the principle of least privilege. This ensures that only the necessary services and users can interact with sensitive data. We also implement encryption for the data stored in S3 (and Aurora) and data in transit, where encryption is completed using AWS-managed encryption keys (AES-256 for S3 and TLS for data in transit).

8. Disaster Recovery and Fault Tolerance

Designed for high availability and disaster recovery, where multi-AZ deployments for Amazon Aurora provide automatic failover in case of a database failure. Additionally, autoscaling and load balancing ensure that the system can handle failures of individual resources without affecting overall system performance. Regular snapshots of the Aurora database and S3 bucket backups ensure that data is protected and can be restored quickly in case of a failure.

9. Cost Optimization

The serverless architecture and dynamic scaling help keep costs low by only using resources when needed. For example, AWS Lambda charges only for the compute time used during function execution, while S3's pay-per-use model ensures that storage costs are aligned with usage. Additionally, by using services like Aurora and ECS with autoscaling, Pixplore avoids over-provisioning and minimizes idle resource costs.

Components

Pixplore comprises of several key components that work together to provide a seamless and scalable image recognition and search platform. Each component serves a specific function and is integrated to ensure efficient processing, security, and user experience.

These components are as follows:

1. Security and Access Control (IAM & VPC)

This component is responsible for controlling and managing access to resources and ensuring secure inter-component communication. **AWS IAM (Identity and Access Management)** roles and policies are used throughout the system to control access to AWS resources, where each service or user is granted the minimum necessary permissions, ensuring that sensitive data is protected. The system is deployed in a **Virtual Private Cloud (VPC)** to isolate backend services and control network access. Security groups and NACLs within the VPC restrict inbound and outbound traffic based on service requirements.

2. User Authentication and Access Control (AWS Cognito)

This component is responsible for user management and authentication, it handles user registration, login, and session management. **AWS Cognito** enables secure authentication and authorization, ensuring that only registered users can access the system's features. It manages user identities, password recovery, and access tokens for API requests, providing a secure and seamless user experience.

3. Image Upload and Storage (Amazon S3)

This component is responsible for handling large image volumes by storing images, uploaded by users, securely and efficiently. **AWS S3** offers a secure, scalable, and cost-efficient way to manage image data. When a user uploads an image, a Lambda function generates a pre-signed URL to allow the image to be uploaded directly to S3, bypassing the backend servers and reducing load.

4. Metadata Extraction and Image Analysis (AWS Rekognition)

This component is responsible for analysing image content and extracting relevant metadata to aid the search functionality. Once images are uploaded, **AWS Rekognition** automatically analyzes the content, extracting metadata such as image labels, objects detected, and even facial attributes, if applicable. This metadata is then stored in **Amazon Aurora** and used to enhance the search functionality, enabling users to find images based on these artifacts.

5. Image Metadata Storage (Amazon Aurora)

This component is responsible for storing and querying image metadata. **Amazon Aurora** is used to store the metadata generated by Rekognition. Aurora is a highly available and scalable relational database that stores metadata such as image tags, labels, and object detection results. The structured data in Aurora is queried by the image search service to quickly retrieve relevant images based on user queries.

6. Image Search and Query (Amazon ECS with Fargate)

This component is responsible for handling image search queries, retrieving images based on the queries, and sending back the results. Built on **Amazon ECS with Fargate**, this component runs containerized services that execute search operations using metadata stored in Aurora. ECS with Fargate offers a serverless container solution that scales based on traffic and performs complex search queries in real-time.

7. Event-Driven Processing (Amazon EventBridge & SQS)

This component is responsible for the event-driven coordination in Pixplore and queuing messages for an efficient decoupled processing workflow. When an image is uploaded, **Amazon EventBridge** triggers downstream workflows, such as initiating metadata extraction with Rekognition and updating the Aurora database. **Amazon SQS** is used to handle asynchronous tasks like queuing image metadata for further processing, allowing for smoother scaling and increased system resilience.

8. API Gateway and RESTful APIs (Amazon API Gateway)

This component is responsible for exposing the RESTful APIs Pixplore utilises, for user interaction. **Amazon API Gateway** acts as the entry point for all incoming requests to the system. It routes user requests to the appropriate backend services, including image upload, metadata processing, and image search. API Gateway also handles security through integration with AWS Cognito to ensure that only authenticated users can access the APIs. It also supports rate-limiting and request throttling to ensure stable system performance.

9. Serverless Compute (AWS Lambda)

This component is responsible for the backend logic execution of processes that are event-driven and require quick execution of smaller workloads. **Lambda** functions are used for backend logic and serverless compute tasks, where these functions are triggered by events, such as image uploads or user requests. Lambda handles tasks like generating pre-signed URLs for image uploads, processing metadata, or triggering events in the system. It scales automatically based on traffic, reducing overhead and ensuring cost-effective operation.

10. Monitoring and Logging (Amazon CloudWatch)

This component is responsible for monitoring Pixplore's performance and troubleshooting. **Amazon CloudWatch** monitors the health and performance of all services in the system. It tracks key metrics such as function execution time, API request latency, and resource utilization, while CloudWatch Logs are used to record detailed logs of system activities, providing real-time visibility into errors or

performance bottlenecks. Custom CloudWatch Alarms notify the development team when thresholds are exceeded, enabling proactive issue resolution.

11. Load Balancing (Elastic Load Balancer – ALB)

This component is responsible for improved distribution of traffic and ensuring consistent and high availability. The **Elastic Load Balancer (ALB)** distributes incoming API traffic across multiple backend services, ensuring that requests are routed to available ECS containers and Lambda functions. It enables horizontal scaling of the system and optimizes performance by balancing the load during traffic spikes. ALB also helps maintain high availability by automatically rerouting traffic in case of failures.

Architecture Diagram

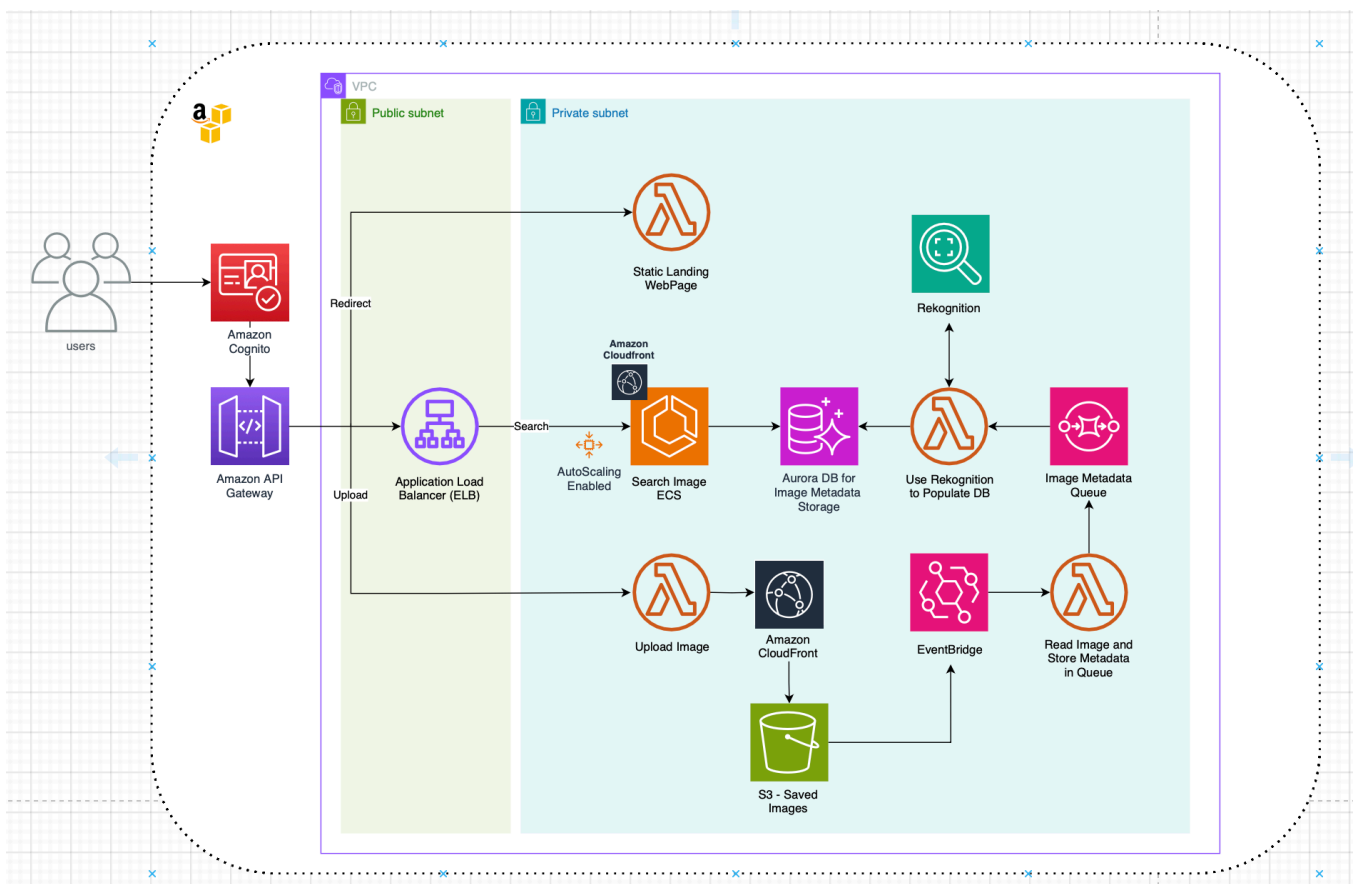


Fig 1. Pixplore's High-Level Architecture Diagram

AWS Services

Each AWS service was meticulously selected to ensure scalability, ease of integration, and cost-efficiency while minimizing the complexity of managing multiple tools or cross-cloud integrations. The alternatives considered would have required more management, lacked some necessary features, or added unnecessary complexity to Pixplore's architecture.

Following are the AWS Services we chose:

1. Amazon S3

- **Purpose:** Store images securely and manage large volumes of data.
- **Alternative:** Amazon EFS (Elastic File System)
- **Reason:** S3 is more cost-effective and scalable for handling static data like images. EFS is better for shared file systems but unnecessary for image storage in Pixplore's architecture. S3 also integrates more efficiently with other AWS services like Lambda and Rekognition.

2. AWS Lambda

- **Purpose:** Serverless compute for event-driven processing and backend logic.
- **Alternative:** Amazon EC2 (Elastic Compute Cloud)
- **Reason:** Lambda provides automatic scaling and cost savings in a serverless environment, charging only for compute time. EC2 would require manual scaling and provisioning, leading to higher operational costs and complexity. Lambda was a more cost-effective and scalable choice for Pixplore's proposed event-driven architecture.

3. Amazon API Gateway

- **Purpose:** Expose and manage RESTful APIs for user interactions.
- **Alternative:** AWS App Runner
- **Reason:** API Gateway is better suited for API management and integrates seamlessly with Lambda, while App Runner is optimized for containerized applications and continuous deployment workflows, which is not necessary for Pixplore's API requirements and making it less ideal.

4. Amazon Rekognition

- **Purpose:** Analyze image content and generate metadata (for e.g., labels, objects).
- **Alternative:** Amazon SageMaker (Custom ML models)
- **Reason:** Rekognition provides ready-made image recognition capabilities, which are sufficient for Pixplore's needs. SageMaker requires custom model development and training, adding complexity that's not needed for Pixplore.

5. Amazon Aurora

- **Purpose:** Relational database for storing image metadata.
- **Alternative:** Amazon RDS (MySQL/PostgreSQL)
- **Reason:** Aurora offers better performance and automatic scaling for larger datasets and high-performance queries. RDS is viable but doesn't scale as efficiently as Aurora for high-demand metadata processing.

6. Amazon ECS with Fargate

- **Purpose:** Containerized compute for running search and query operations.
- **Alternative:** Amazon ECS with EC2 instances
- **Reason:** ECS with Fargate is serverless and automatically scales, reducing overhead and management. ECS with EC2 would require more management and manual scaling of the underlying infrastructure.

7. Amazon EventBridge

- **Purpose:** Event-driven coordination for triggering workflows.
- **Alternative:** AWS Step Functions, Amazon SNS
- **Reason:** EventBridge is ideal for routing events between AWS services. Step Functions is better suited for complex state-based workflows, while SNS is more for messaging and lacks the advanced event routing of EventBridge.

8. Amazon SQS

- **Purpose:** Queue-based messaging for decoupling processing tasks.
- **Alternative:** Amazon MQ
- **Reason:** SQS is fully managed, scalable, and integrates easily with other AWS services. Amazon MQ is more complex and is better suited for legacy applications requiring a message broker, which Pixplore doesn't need.

9. Elastic Load Balancer (ALB)

- **Purpose:** Distribute incoming traffic across backend services.
- **Alternative:** Classic Load Balancer (CLB), Network Load Balancer (NLB)
- **Reason:** ALB offers advanced routing capabilities for HTTP/HTTPS traffic, making it ideal for Pixplore's application-level traffic. CLB is outdated, and NLB is optimized for TCP traffic, not HTTP/HTTPS.

10. Amazon Cognito

- **Purpose:** Manage user authentication, registration, and access control.
- **Alternative:** AWS IAM Identity Center (formerly AWS SSO)
- **Reason:** Cognito is designed for managing user-facing authentication and integrates well with APIs. IAM Identity Center is focused on managing AWS access, not for user authentication in custom applications.

11. Amazon CloudWatch

- **Purpose:** Monitor system performance, logs, and alerts.
- **Alternative:** AWS X-Ray
- **Reason:** CloudWatch provides a comprehensive monitoring solution for system metrics, logs, and alarms, essential for Pixplore's observability needs. X-Ray is more suited for detailed tracing of distributed applications and isn't necessary for general monitoring.

12. AWS Identity and Access Management (IAM)

- **Purpose:** Control access to AWS resources securely.
- **Alternative:** AWS Organizations
- **Reason:** IAM provides fine-grained control over access to individual resources. AWS Organizations is more focused on managing access across multiple AWS accounts, which isn't required for Pixplore's single-account based user model.

13. Amazon Virtual Private Cloud (VPC)

- **Purpose:** Network isolation and security for backend services.
- **Alternative:** AWS PrivateLink
- **Reason:** VPC allows for a fully customizable, isolated network environment. PrivateLink is specialized for secure access to AWS services over private networks, but VPC offers the necessary isolation without the added complexity.

Design Excellence

We designed Pixplore with the priority to adhere to the AWS Well-Architected Framework, ensuring it meets its key pillars. Below is a breakdown of how Pixplore satisfies each pillar with specific design considerations:

- **Operational Excellence:**

- i. *Monitoring and Observability* → CloudWatch is used to monitor performance, track Lambda execution times, and capture logs for troubleshooting. Alarms are set up to alert of performance issues, ensuring fast response times.
- ii. *Automated Scaling* → AWS Lambda automatically scales based on demand, while Amazon ECS with Fargate scales containers dynamically to handle varying levels of image search traffic, providing cost-efficient and effective scaling.
- iii. *Continuous Improvement* → Amazon EventBridge is used for event-driven workflows, allowing for flexibility and easy adaptation as business needs evolve. It ensures Pixplore can scale and evolve over time without disrupting operations.

- **Security:**

- i. *Authentication and Access Control* → Amazon Cognito manages secure user authentication with support for MFA. IAM roles ensure that services and users have only the minimal permissions needed to interact with the system.
- ii. *Data Protection* → Amazon S3 uses AWS-managed encryption keys to protect stored data. All inter-service communication is encrypted using TLS to secure data in transit.
- iii. *Network Security* → Amazon VPC isolates backend services, placing databases and internal services in private subnets. Security Groups and Network ACLs restrict access to sensitive parts of the system.

- **Performance:**

- i. *Scalable Compute* → AWS Lambda dynamically scales based on demand, ensuring efficient use of compute resources without overprovisioning. Amazon ECS with Fargate scales containerized workloads automatically, adjusting to traffic fluctuations while maintaining performance.
- ii. *Storage and Content Delivery* → Amazon S3 handles high-volume image uploads with low latency. Amazon Rekognition automatically scales to meet demand, ensuring efficient image analysis without overloading the system.
- iii. *Optimized Database* → Amazon Aurora supports automatic scaling and read replicas to handle query loads, ensuring fast metadata retrieval even under high traffic conditions.

- **Reliability:**

- i. *High Availability* → Amazon Aurora uses multi-AZ deployments, providing automatic failover for high availability. Elastic Load Balancer (ALB) distributes traffic evenly to ECS containers and Lambda functions, ensuring the system remains responsive during traffic spikes.
- ii. *Fault Tolerance* → Amazon SQS decouples tasks, ensuring that message queues continue to function during failure events and retries occur automatically. EventBridge provides reliable event routing, preventing workflow disruptions.
- iii. *Recovery* → Aurora provides automatic backups and supports point-in-time recovery. S3 versioning ensures that image data is preserved and can be restored if needed.

- **Cost Optimisation:**

- i. *Pay-as-You-Go* → Amazon Aurora uses multi-AZ deployments, providing automatic failover for high availability. ALB distributes traffic evenly to ECS containers and Lambda functions, ensuring the system remains responsive during traffic ups and downs.
- ii. *Efficient Resource Allocation* → Amazon Aurora's auto-scaling and Elastic Load Balancer (ALB) ensure that resources are allocated efficiently based on demand, preventing over-provisioning, and reducing unnecessary costs.
- iii. *Serverless Architecture* → AWS Lambda and ECS with Fargate are both serverless options that scale automatically, reducing the need for idle resources and lower operational costs.

- **Sustainability:**

- i. *Efficient Storage* → Amazon S3 uses highly durable, scalable storage, which is optimized for long-term retention of image data without requiring extensive hardware or energy resources. Data is only replicated and stored where necessary, preventing over-consumption of resources.
- ii. *Serverless Compute* → AWS Lambda and ECS with Fargate only use compute resources when needed, minimizing idle time, and reducing energy consumption.
- iii. *Sustainable Cloud Services* → AWS's data centers are designed to be energy-efficient and powered by renewable energy sources, contributing to the environmental sustainability of the Pixplore system. By relying on AWS's infrastructure, Pixplore benefits from AWS's promise for sustainability.

Load Testing

To ensure the reliability and scalability of Pixplore, we conducted load testing using **K6**, an open-source load testing tool designed for modern web applications. The purpose of this testing was to simulate real-world user traffic, identify system bottlenecks, and validate the performance of our AWS-backed infrastructure.

Here are the results of the two runs we were able to perform:

First run:

```
Generate Pre-Signed URL
X Pre-Signed URL received
  99% - ✓ 2169 / X 3

Upload File to S3
X File uploaded successfully
  37% - ✓ 886 / X 1363

Search Images by Label
X Search successful
  99% - ✓ 2169 / X 3

checks.....: 78.98% 5104 out of 6513
data_received.....: 16 MB 36 kB/s
data_sent.....: 8.6 MB 20 kB/s
group_duration.....: avg=2.3s min=206.46ms med=641.46ms max=1m0s p(90)=8.23s p(95)=13.71s
http_req_blocked.....: avg=795.11ms min=0s med=0s max=16.19s p(90)=677.23ms p(95)=7.45s
http_req_connecting.....: avg=736.03ms min=0s med=0s max=15.28s p(90)=233.9ms p(95)=7.23s
http_req_duration.....: avg=388.70ms min=0s med=288.14ms max=40.04s p(90)=653.49ms p(95)=961.02ms
{ expected_response:true }...: avg=385.6ms min=217.29ms med=290.67ms max=7.03s p(90)=780.61ms p(95)=972.49ms
http_req_failed.....: 21.01% 1369 out of 6513
http_req_receiving.....: avg=17.09ms min=0s med=116.9µs max=2.72s p(90)=1.06ms p(95)=2.33ms
http_req_sending.....: avg=164.83µs min=0s med=0s max=3.33ms p(90)=546.44µs p(95)=616.33µs
http_req_tls_handshaking.....: avg=58.97ms min=0s med=0s max=3.72s p(90)=224.76ms p(95)=232.95ms
http_req_waiting.....: avg=371.52ms min=0s med=283.33ms max=48.04s p(90)=538.97ms p(95)=879.25ms
http_reqs.....: 6513 14.81457/s
iteration_duration.....: avg=4.97s min=1.7s med=2.4s max=1m1s p(90)=14.97s p(95)=16.41s
iterations.....: 2172 4.98046/s
vus.....: 1 min=1 max=50
vus_max.....: 50 min=50 max=50

running (7m19.6s), 00/50 VUs, 2172 complete and 0 interrupted iterations
default ✓ [=====] 00/50 VUs 7m0s
PS C:\Users\Abdul_Ahad\Documents\GitHub\Pixplore\testing\LoadTesting> |
```

Second run:

```
running (4m50.4s), 40/50 VUs, 2362 complete and 0 interrupted iterations

Generate Pre-Signed URL
X Pre-Signed URL received
  97% - ✓ 4231 / X 93

Upload File to S3
X File uploaded successfully
  36% - ✓ 1549 / X 2682

Search Images by Label
✓ Search successful

checks.....: 78.45% 10104 out of 12879
data_received.....: 31 MB 73 kB/s
data_sent.....: 17 MB 40 kB/s
group_duration.....: avg=719.62ms min=210.86ms med=618.17ms max=21.45s p(90)=1.27s p(95)=1.69s
http_req_blocked.....: avg=128.81ms min=0s med=0s max=9.27s p(90)=466.59ms p(95)=484.66ms
http_req_connecting.....: avg=69.98ms min=0s med=0s max=8.74s p(90)=226.57ms p(95)=233.03ms
http_req_duration.....: avg=361.58ms min=0s med=283.99ms max=4.3s p(90)=566.01ms p(95)=947.19ms
{ expected_response:true }...: avg=389.62ms min=228.86ms med=307.77ms max=4.3s p(90)=645.33ms p(95)=986.36ms
http_req_failed.....: 21.54% 2775 out of 12879
http_req_receiving.....: avg=17.83ms min=0s med=110.4µs max=3.77s p(90)=999.4µs p(95)=2.05ms
http_req_sending.....: avg=155.2µs min=0s med=0s max=4.51ms p(90)=536.8µs p(95)=601.5µs
http_req_tls_handshaking.....: avg=58.72ms min=0s med=0s max=2.3s p(90)=236.72ms p(95)=244.35ms
http_req_waiting.....: avg=343.59ms min=0s med=281.16ms max=4.3s p(90)=485.63ms p(95)=809.43ms
http_reqs.....: 12879 30.512412/s
iteration_duration.....: avg=2.48s min=1.46s med=2.31s max=22.7s p(90)=3.26s p(95)=3.58s
iterations.....: 4324 10.244248/s
vus.....: 1 min=1 max=50
vus_max.....: 50 min=50 max=50

running (7m02.1s), 00/50 VUs, 4324 complete and 0 interrupted iterations
default ✓ [=====] 00/50 VUs 7m0s
```

Disaster Recovery

We employ a set of strategies for disaster recovery and are as follows:

1. Pilot Light

The pilot light strategy ensures that critical components of Pixplore remain operational even during significant disruptions. By leveraging AWS Lambda's serverless compute capabilities, we maintain essential backend logic without the need for constantly running infrastructure. Amazon RDS (Aurora) is configured as serverless with automatic failover, ensuring that our database is always available, even in the event of a failure. ECS (Elastic Container Service) with Fargate only requires minimal task execution during normal operations, further supporting a lightweight, always-ready environment. This setup minimizes operational costs while maintaining readiness for full-scale recovery when needed.

2. Warm Standby

For more rapid recovery scenarios, Pixplore implements a warm standby strategy. This approach involves maintaining pre-configured resources that can be quickly activated to handle higher loads. Our use of ECS with Fargate for containerized services allows containers to spin up swiftly, ensuring immediate availability during traffic spikes. Similarly, AWS Lambda scales automatically to cater for increased requests, providing a resilient and responsive environment that minimizes disruption to end users.

3. Backup and Restore

Pixplore relies on robust backup and restore mechanisms to ensure data integrity and swift recovery from data loss. All user uploaded images are securely stored in Amazon S3, which supports versioning and redundancy for high durability. For application code and infrastructure, we use Git for version control and Terraform with an S3 backend to maintain and recover our deployment configurations. This ensures that both user data and application infrastructure are protected and recoverable in any failure scenario.

4. Multi-Site or Active-Active

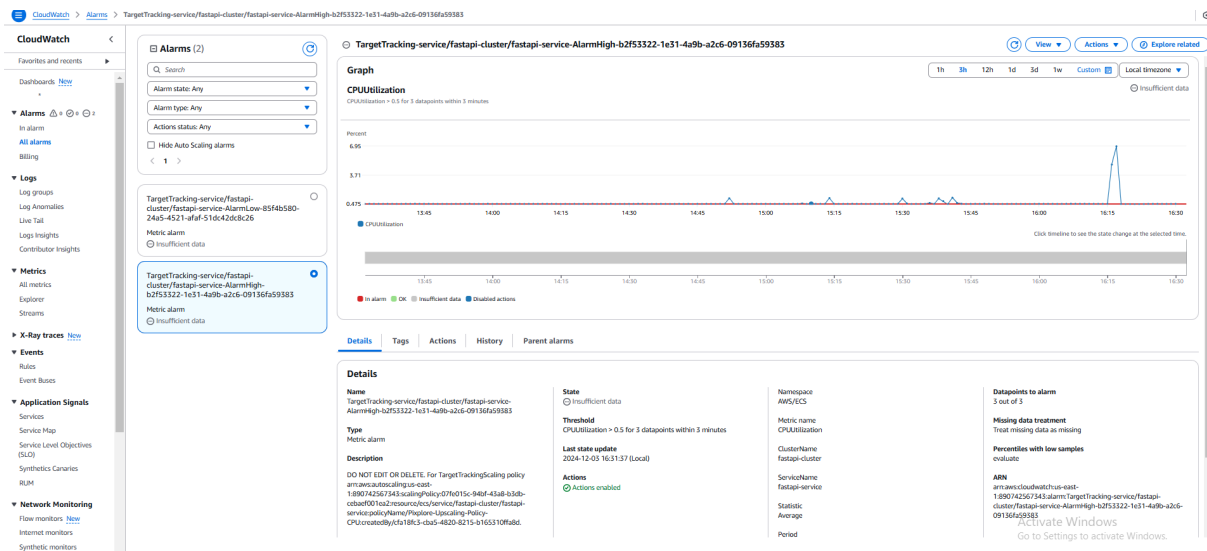
Currently, Pixplore does not implement a multi-region or active-active disaster recovery strategy. Instead, we focus on a single-region architecture optimized for cost efficiency and scalability. While this approach meets our current operational requirements, we can expand to multi-region disaster recovery in the future as the need arises, however we are confident that AWS's commitment to the provision of quality services deem region unavailability, unlikely.

5. Elastic Disaster Recovery

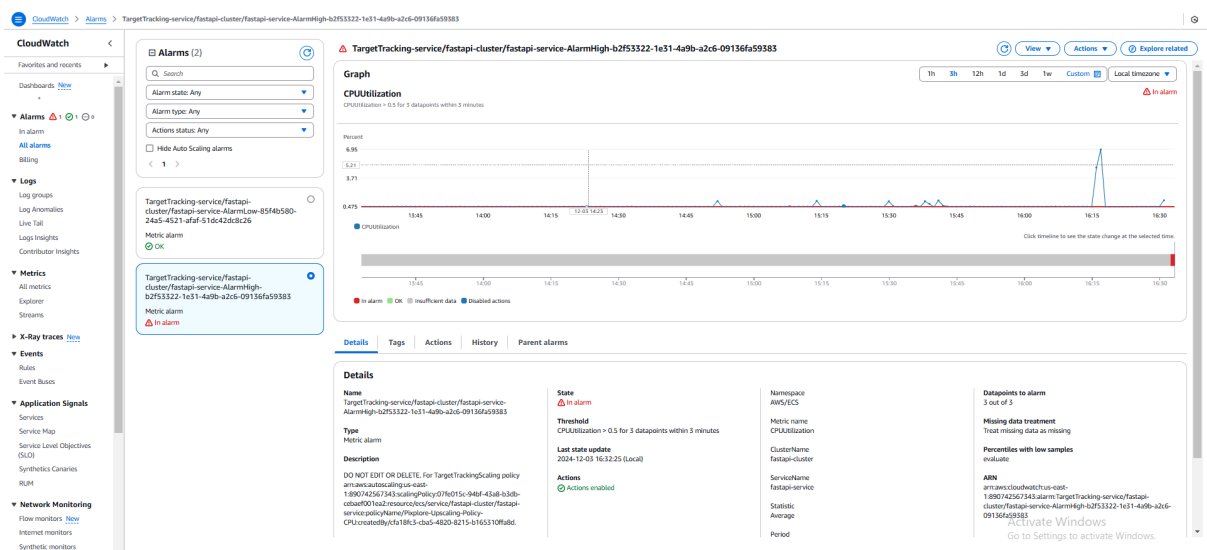
Pixplore employs elastic disaster recovery to dynamically adapt to varying traffic demands during outages or spikes. Through AWS's auto-scaling capabilities, both Lambda functions and ECS tasks adjust seamlessly to handle increased loads. This ensures that our system can quickly recover from disruptions by provisioning additional compute resources as needed, without manual intervention. The serverless nature of these services enhances cost efficiency and reduces downtime, providing users with uninterrupted access to essential functionalities.

A demonstration of this is described below, it also shows our Load Balancer working:

Currently the Alarm is set to watch minimal change in CPU to trigger scaling.



Tracker went in alarm state.



Scaling Policy applied, setting the count to 2 as can be seen in the picture below.

The screenshot shows the AWS Management Console for the 'ecs-tg' target group. The 'Details' section displays the following configuration: Target type (IP), Protocol (Port), Port (8000), IP address type (IPv4), Load balancer (elastic-lb), Protocol version (HTTP1), and VPC (vpc-0c38bca680795a135). The 'Targets' section shows 2 total targets, all healthy. The 'Distribution of targets by Availability Zone (AZ)' section shows 2 targets in us-east-1a and 0 in us-east-1b. The 'Registered targets' section shows 2 targets: 10.0.1.169 and 10.0.2.131, both healthy.

We have two tasks running currently.

The screenshot shows the AWS Management Console for the 'fastapi-service'. The 'Service overview' section shows the status as 'Active' and 'Tasks (2 Desired)' with 0 Pending and 2 Running. The 'Tasks' section shows 1/4 tasks, with 2 running and 2 stopped. The 'Containers' section shows 1 container running.

These tasks have been added to loadbalancer's target group, where we have two healthy instances.

EC2 > Target groups > ecs-tg

Dashboard

EC2 Global View

Events

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

Load Balancing

Load Balancers

Target Groups

Trust Stores

Auto Scaling

Auto Scaling Groups

Settings

ecs-tg

Actions

Details

Target type

IP

IP address type

IPv4

Protocol

Port

HTTP:8000

Protocol version

HTTP1

VPC

vpc-0d38d468b07795a139

Load balancer

ecs-alb

2

Total targets

2

Healthy

0

Unhealthy

0

Unused

0

Initial

0

Draining

0

Anomalous

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

Targets

Monitoring

Health checks

Attributes

Tags

Registered targets (2)

Anomaly mitigation: Not applicable

De-register

Register targets

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Filter targets

| <input type="checkbox"/> | IP address | Port | Zone | Health status | Health status details | Administrative override | Override... | Anomaly detection... |
|--------------------------|------------|------|-----------------|---------------|-----------------------|-------------------------|-------------|----------------------|
| <input type="checkbox"/> | 10.0.1.169 | 8000 | us-east-1a L... | Healthy | - | No override | No event... | Normal |
| <input type="checkbox"/> | 10.0.2.131 | 8000 | us-east-1b L... | Healthy | - | No override | No event... | Normal |

Activate Windows
Go to Settings to activate Windows.

Search working without any issue.

Upload

Search



Logout

Search Labels

Label to search:

food

Search



Activate Windows
Go to Settings to activate Windows.

After adding the down scaling policy, the tasks have scaled in and now reduced to minimum desired capacity of 1.

In Alarm.

Alarms (2)

Search

Alarm state: Any

Alarm type: Any

Actions status: Any

Hide Auto Scaling alarms

Clear selection

Create composite alarm

Actions

Create alarm

| <input type="checkbox"/> | Name | State | Last state update (Local) | Conditions | Actions |
|--------------------------|---|----------|---------------------------|---|-----------------|
| <input type="checkbox"/> | TargetTracking-service/fastapi-cluster/fastapi-service-AlarmHigh-30630ab7-ba2f-472a-801c-0b599617c519 | OK | 2024-12-03 16:49:43 | CPUUtilization > 75 for 3 datapoints within 3 minutes | Actions enabled |
| <input type="checkbox"/> | TargetTracking-service/fastapi-cluster/fastapi-service-AlarmLow-34ff2bc1-806d-4188-8195-540c060539fc | In alarm | 2024-12-03 16:49:03 | CPUUtilization < 67.5 for 15 datapoints within 15 minutes | Actions enabled |

Scaling Activity Updating.

Amazon Elastic Container Service

Clusters updated

Namespaces

Task definitions

Account settings updated

Install AWS Copilot

Amazon ECR

Repositories

AWS Batch

Documentation

Discover products

Subscriptions

Tell us what you think

Amazon Elastic Container Service

Clusters > fastapi-cluster > Services > fastapi-service > Service auto scaling > View

Task definition fastapi-task-2

Deployment status Success

Health and metrics Tasks Logs Deployments Service auto scaling Tags

How to use service auto scaling

Task count

Number of tasks to run for fastapi-service 1-3

Scaling policies Scheduled actions Recommendations

Scaling policies (1)

Find a scaling policy

Table with 6 columns: Name, Type, Status, Scaling mechanism, Target metric

Table with 1 row: Playfare-UpScaling-Policy-CPU, Target tracking, Active, Maintain target, ECSServiceAverageCPUUtilization (75%)

Scaling activities

Find scaling activity

Table with 7 columns: Resource ID, Scalable dimension, Status, Status message, Description, Not scaled reason, Cause

Table with 10 rows of scaling activities

Number of tasks are reducing.

Amazon Elastic Container Service

Clusters updated

Namespaces

Task definitions

Account settings updated

Install AWS Copilot

Amazon ECR

Repositories

AWS Batch

Documentation

Discover products

Subscriptions

Tell us what you think

Amazon Elastic Container Service

Clusters > fastapi-cluster > Services > fastapi-service > Tasks

Task definition fastapi-task-2

Deployment status Success

Health and metrics Tasks Logs Deployments Service auto scaling Tags

Tasks (1/5)

Filter tasks by property or value

Filter desired status Any desired status

Filter launch type Any launch type

Table with 11 columns: Task, Last status, Desired status, Task definition, Health status, Started at, Container instance, Launch type, Platform, CPU, Memory

Table with 6 rows of task details

Containers for task 2315b353cf6e4f01bd667c524cc01f51

Containers (1)

Filter containers

Table with 8 columns: Container name, Container runtime ID, Image URI, Image Digest, Status, Health status, CPU, Memory hard/soft limit

Table with 1 row of container details

And are being unregistered by the ALB.

Dashboard

EC2 Global View

Events

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Capacity Reservations

Images

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups

Elastic IP

Placement Groups

Key Pairs

Network Interfaces

Load Balancing

Load Balancers

Target Groups

Trust Stores

Auto Scaling

Auto Scaling Groups

Settings

Target groups > ecs-tg

Actions

Details

Target type IP

IP address type IPv4

Protocol : Port HTTP: 8000

Load balancer ecs-alb

Protocol version HTTP1

VPC vpc-0d58bda8b0799e119

3 Total targets

2 Healthy

0 Unhealthy

0 Unused

0 Initial

1 Draining

Distribution of targets by Availability Zone (AZ)

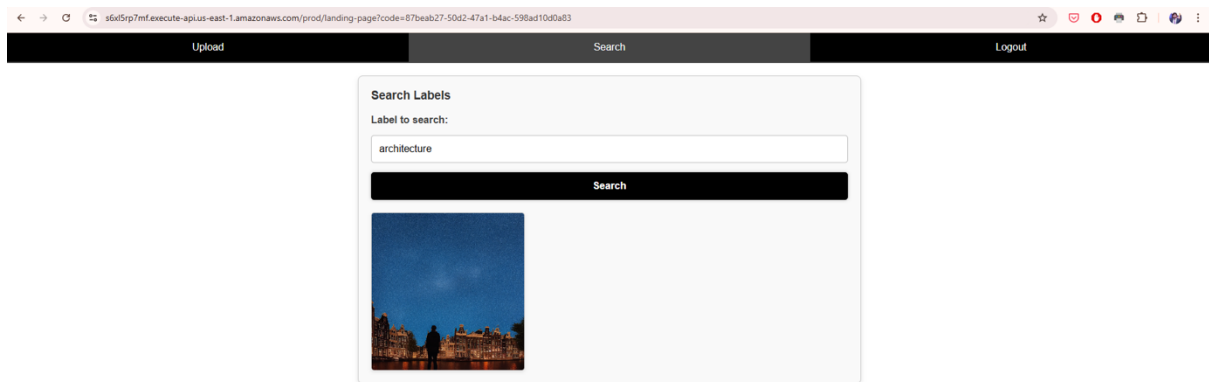
Registered targets (3)

Filter targets

Table with 8 columns: IP address, Port, Zone, Health status details, Administrative override, Overriden, Anomaly detection

Table with 3 rows of target details

The search is working as expected.



Cost Calculator

Our system is currently implemented and operates exclusively in the production (PROD) environment. We do not maintain separate DEV, QA, or UAT environments, as our focus has been on optimizing the live system for scalability, reliability, and performance. However, we have included what we think would be the expected costs for these environments.

Below is a summary of the environments' estimated costs:

| Environment | Estimated Monthly Cost |
|-------------------------------|------------------------|
| Development (DEV) | ~ 20\$ |
| Quality Assurance (QA) | ~ 40\$ |
| User Acceptance Testing (UAT) | ~ 80\$ |
| Production (PROD) | ~ 28\$ |

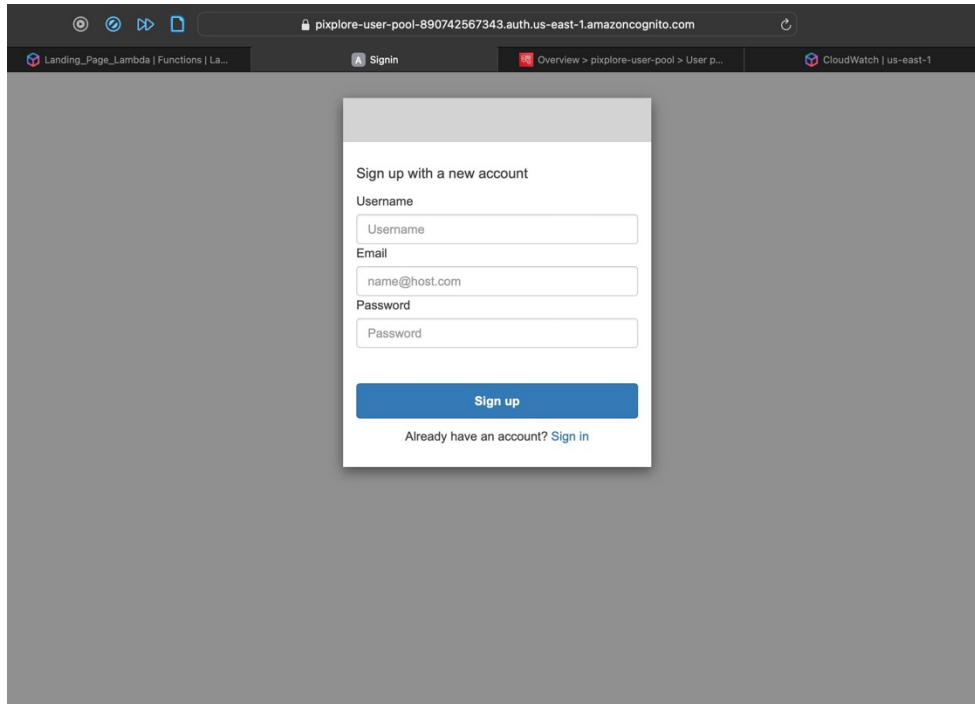
Link to our current estimates for production (PROD) retrieved using AWS Pricing Calculator:

<https://calculator.aws/#/estimate?id=b68b10a9b600ae103f24733dc8832132f168ad89>

User Guide

A demonstration of the user experience; once logged in there are buttons in the header to navigate to the upload and search image pages.

User first visits the signup page:



Sign up with a new account

Username

Email

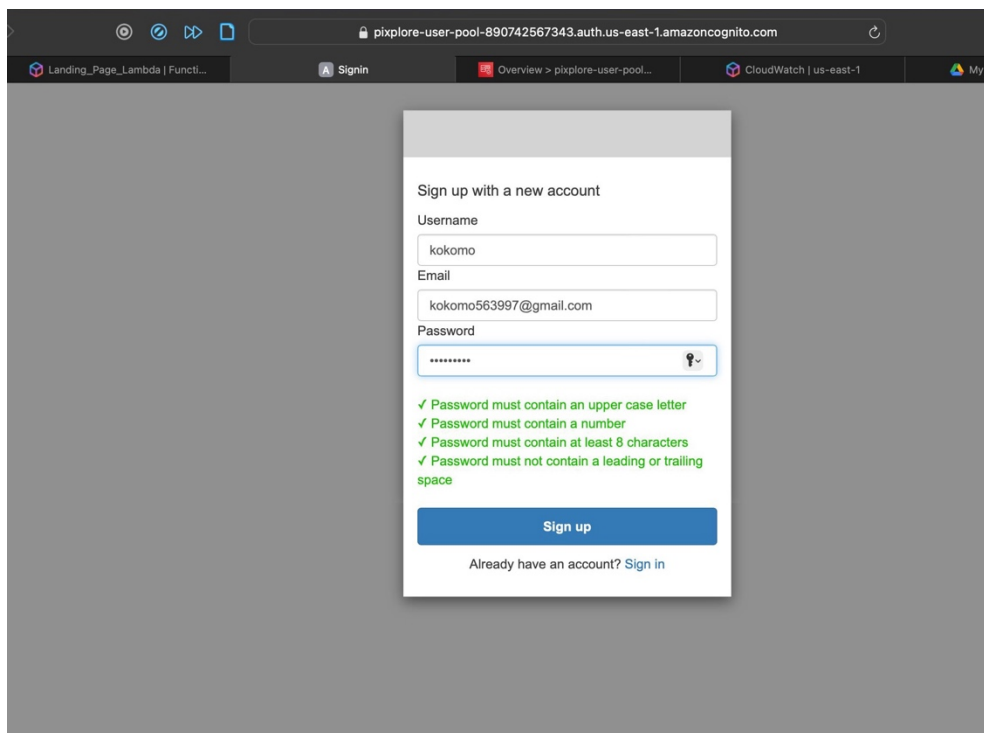
name@host.com

Password

Sign up

Already have an account? [Sign in](#)

User signs up with details compliant with the password checks.



Sign up with a new account

Username

kokomo

Email

kokomo563997@gmail.com

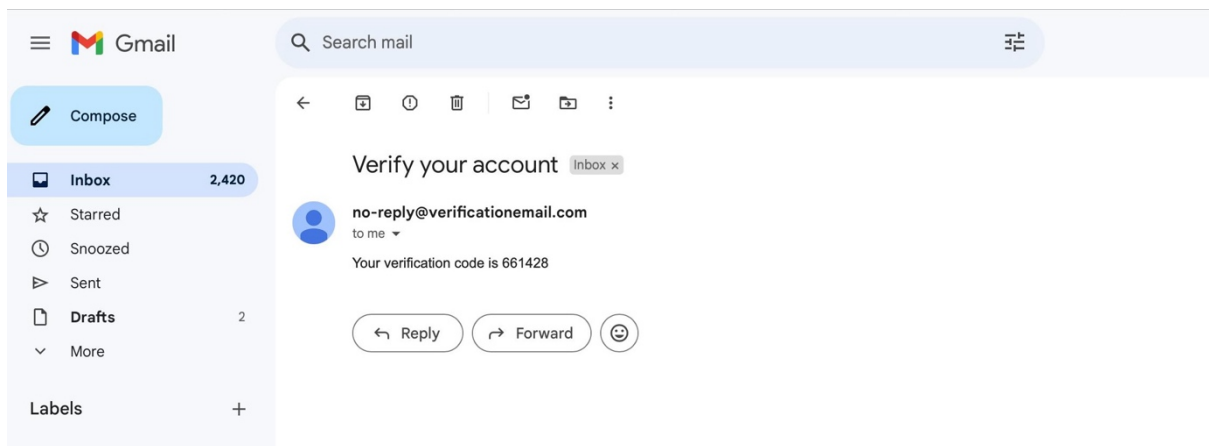
Password

- ✓ Password must contain an upper case letter
- ✓ Password must contain a number
- ✓ Password must contain at least 8 characters
- ✓ Password must not contain a leading or trailing space

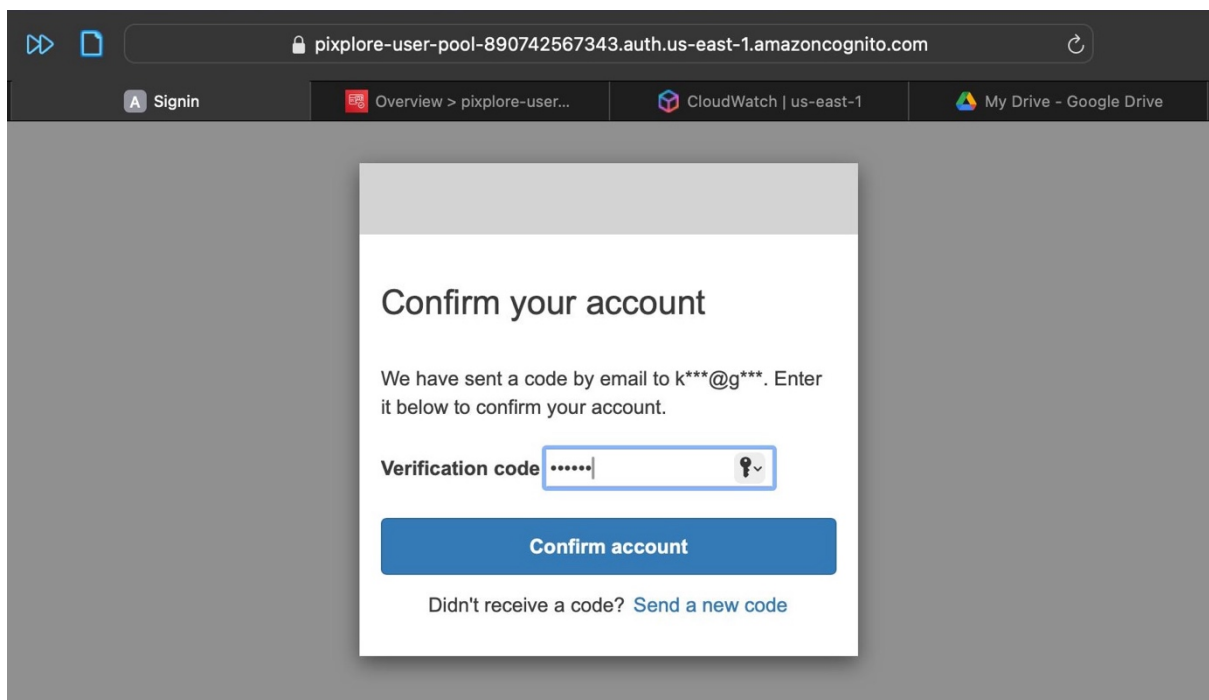
Sign up

Already have an account? [Sign in](#)

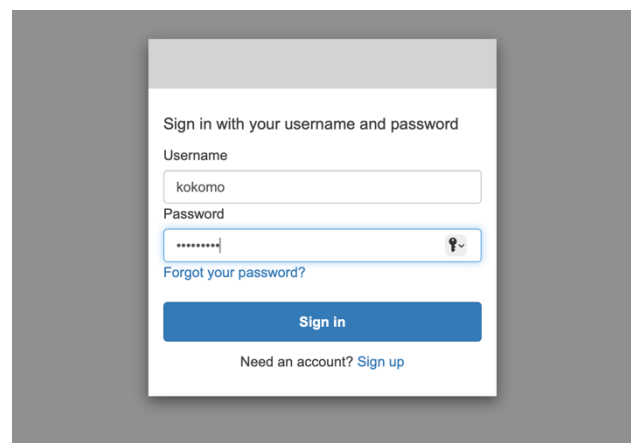
User is sent a verification code to their registered email.



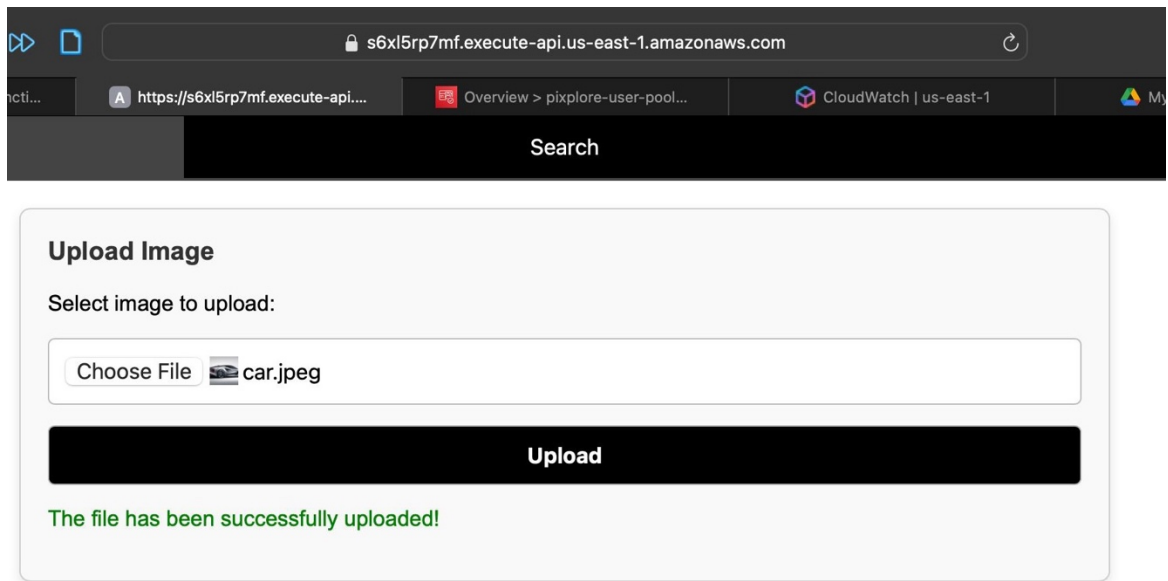
User enters the code to complete the signup process.



User then logs in.



User then uploads an image.



Upload Image

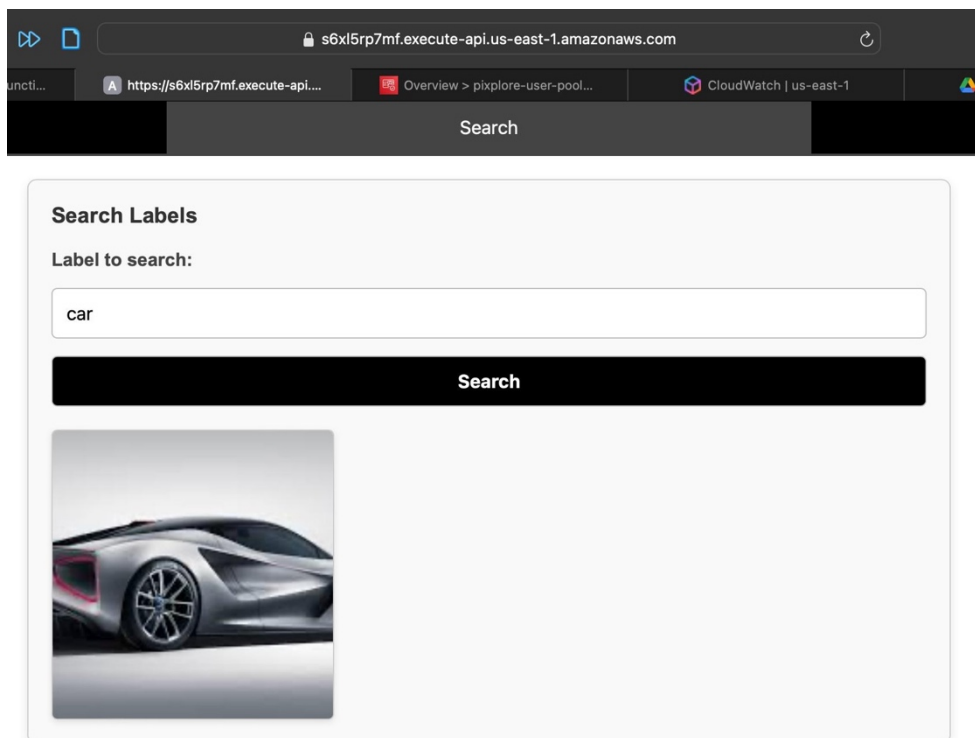
Select image to upload:

Choose File car.jpeg

Upload

The file has been successfully uploaded!

User then searches for the image using a keyword 'car'.




Search Labels

Label to search:

car

Search



User can then log out.