**1. Application Overview**

The Group36 application is a cloud-based microservices solution designed to handle video processing and provide API services to users. It is structured to process video data efficiently while offering a web-based API that enables seamless interaction for end-users. Leveraging Amazon Web Services (AWS) infrastructure, this application ensures scalability, reliability, and cost-effectiveness, addressing both CPU-intensive tasks and frequent API requests. Built with a microservices architecture, the application consists of two primary components: a web client API and a video processing service, both hosted on Amazon Elastic Container Service (ECS) with Fargate for streamlined container management. This design empowers the application to dynamically adjust its resource usage based on workload demands, making it a robust solution for enterprise-grade deployments.

**2. Application Architecture**

**Architecture Diagram and Overview**

The architecture of the Group36 application is a microservices-based model hosted entirely on AWS, leveraging cloud-native services to ensure reliability and scalability. Below is a breakdown of each major component used in the architecture:

* **Elastic Container Service (ECS) with Fargate**: ECS manages and runs the Docker containers for both the web client API and video processing microservices. ECS Fargate abstracts the need to manage underlying server infrastructure, allowing for seamless scaling and reducing the operational overhead associated with container orchestration.
* **Application Load Balancer (ALB)**: The ALB distributes incoming HTTP requests across ECS tasks, ensuring that both the web client API and video processing services can handle multiple concurrent requests efficiently. This load balancing capability enables fault tolerance, ensuring continuous service even if individual instances fail.
* **Elastic Container Registry (ECR)**: ECR serves as a secure Docker image repository for both services, simplifying deployment updates and version control of container images. With ECR, deploying updated services is straightforward, allowing for quick iteration without complex repository management.
* **Amazon Route 53**: Route 53 provides domain name services, routing user requests through the ALB for enhanced accessibility and reliability. This allows users to interact with the application via a user-friendly domain name, masking the complexities of the underlying infrastructure.
* **Amazon Secrets Manager**: Secrets Manager securely stores and manages credentials for accessing private repositories on ECR. This integration ensures that credentials are stored securely and are easily accessible to authorized services, enhancing the application's security.
* **Amazon CloudWatch**: CloudWatch monitors and logs performance metrics for ECS services, offering insights into CPU, memory usage, and error rates in real-time. This allows for proactive management and troubleshooting of service issues.

The architecture is designed to be modular, with each service performing a specific role within the application ecosystem, promoting reliability, scalability, and efficient resource utilization.

(*Include your architecture diagram here.*)

**3. Justification of Architecture**

The architecture of the Group36 application was designed with scalability, security, and performance in mind, aligning with the requirements of a cloud-based microservices solution.

**Division into Microservices**

The decision to split the application into two primary microservices—the web client API and the video processing service—was driven by the unique requirements of each service. The web client API is optimized for handling HTTP requests from end-users, while the video processing service is CPU-intensive, focusing on resource-demanding tasks. This separation enables each service to be managed, scaled, and optimized independently, promoting modularity and maintainability.

**Choice of Compute: ECS with Fargate**

Amazon ECS with Fargate was chosen to manage the Docker containers due to its serverless infrastructure approach, which eliminates the need to maintain the underlying servers. Fargate allows for automatic scaling based on the defined CPU and memory thresholds, which is essential for the CPU-heavy video processing tasks. The serverless model of Fargate also reduces operational complexity, allowing the team to focus on application development rather than infrastructure management.

**Load Distribution and Communication**

The Application Load Balancer (ALB) plays a crucial role in distributing incoming requests among ECS tasks. The ALB supports HTTP/HTTPS protocols and offers automatic routing based on health checks, ensuring that user requests are handled by healthy instances. This approach enhances fault tolerance and improves the resilience of the application.

**Service Abstractions**

Service abstractions like ALB, ECR, and Secrets Manager streamline the application’s infrastructure by simplifying common tasks. The ALB manages routing and load distribution, ECR provides a managed Docker registry for storing container images, and Secrets Manager securely stores credentials for accessing ECR. These abstractions reduce complexity, improving the application's performance and security.

These design choices reflect a balance of performance, cost, and ease of management, ensuring that the application remains flexible, scalable, and adaptable to changing requirements.

**4. Response to Project Criteria**

This section addresses each project criterion in detail, ensuring that the application fulfills all requirements set forth for the deployment on AWS.

(*Expand based on specific project criteria that need to be addressed.*)

**5. Cost Estimate**

Using the AWS Pricing Calculator, the estimated monthly cost for running the Group36 application with 50 concurrent users is as follows:

* **ECS with Fargate**: Approximately $200
* **Application Load Balancer**: $50
* **Elastic Container Registry (ECR)**: $5
* **Secrets Manager**: $0.40
* **CloudWatch Monitoring**: $10
* **Route 53 Domain Costs**: $1

This cost estimate reflects the typical usage and operational costs for handling moderate traffic. For a full breakdown, please refer to the AWS Pricing Calculator link: (*Insert shareable AWS Pricing Calculator link here*).

**6. Scaling Up the Application**

To scale the Group36 application to support 10,000 concurrent users, several architectural enhancements are required:

**Microservices**

The current microservices design may benefit from further subdivision, especially within the video processing service. Additional microservices can be created to handle specific video processing tasks, distributing the load more effectively and minimizing latency.

**Compute**

ECS with Fargate can remain viable, but enhanced auto-scaling configurations would be necessary to accommodate large spikes in demand. Alternatively, high-performance EC2 instances could be used to optimize costs and improve resource utilization.

**Load Distribution**

At higher scales, a multi-tiered load balancing approach may be required. Introducing regional load balancers and using Amazon CloudFront as a content delivery network (CDN) for static content can alleviate pressure on the ALB, improve latency, and deliver faster response times globally.

**7. Securing the Application**

Security is integral to the design of the Group36 application. Key security measures implemented include:

* **IAM Roles with Least Privilege**: Each ECS task and service operates with dedicated IAM roles and minimal permissions, adhering to the principle of least privilege to prevent unauthorized access.
* **Encrypted Communication**: The ALB enforces HTTPS, ensuring encrypted data transmission between the client and the application.
* **Controlled Access to ECR**: Secrets Manager securely stores ECR credentials, restricting access to authorized services only.
* **Monitoring and Logging**: CloudWatch provides real-time monitoring and logging, enabling rapid response to security incidents and identifying potential vulnerabilities.

These measures reinforce a strong security posture, protecting both the application and its users from cyber threats.

**8. Sustainability**

Sustainability is considered at multiple levels within the Group36 application:

* **Software**: Utilizing ECS and Fargate’s serverless model minimizes idle resource consumption, as resources are dynamically allocated based on demand.
* **Hardware**: AWS-managed infrastructure allows us to leverage their efficient, energy-optimized data centers.
* **Data Center**: AWS’s commitment to renewable energy aligns with sustainability goals, as our application benefits from AWS’s sustainable data center practices.
* **Resources**: By optimizing container images to reduce memory and CPU usage, we reduce operational costs and environmental impact.

These practices contribute to a sustainable, efficient application, reducing the ecological footprint of the deployment.

**Conclusion**

The Group36 application is designed for scalability, security, and efficiency within AWS’s cloud ecosystem. The architecture leverages AWS’s managed services to reduce complexity, ensuring that the application can scale dynamically while maintaining cost efficiency and sustainability. The design considerations, security measures, and sustainability practices reflect an application that is ready for enterprise-scale deployment with minimal operational overhead.