Devops Assessment

New Social Theory, Devops Engineer Role.

UseCase

New Social Theory is developing an application which is going to provide following features to end customers:

1. The application will use OAuth authentication provided by Google or Facebook to allow users to sign up and sign in to the platform.

2. When users first log in, they can view profiles of the most popular celebrities on the internet.

3. Users can filter their feeds to only show content from the celebrities they are following.

4. In the feeds, users can view the social activities of celebrities on platforms like Facebook, Instagram, and Twitter.

5. The feeds will also display merchandise that celebrities are selling or endorsing on e-commerce platforms like Etsy or StockX.

6. For security reasons, the application will block access from certain countries like Iraq and Somalia.

7. To prevent brute force attacks, the application will implement rate limiting on certain API endpoints.

8. User settings and personal profiles will be persistently stored.

9. Data aggregated from various social platforms can be utilized for data analytics purposes.

10. The front-end will be developed using React Native technologies and deployed on the Google Play Store and Apple App Store.

11. The back-end will be built using Java Spring-Boot microservices.

12. All services will be stateless and multi-instance to support zero-downtime deployment.

13. API calls from outside the network must use HTTPS.

14. The application will be monitored, and alerts will be triggered in the event of service disruptions.

Devops Assessment

Assessment

Note:- You can download the file below and open it with (draw.io) to view the diagram.

Based on above functional requirements, please advise on following aspects from Devops

perspective:

1. Describe the CI/CD process for the project in terms of workflow diagrams.

You can provide a general, high-level workflow diagram that illustrates the typical CI/CD

process without going into the specific implementation details.

 # High-level workflow diagram

1) https://drive.google.com/file/d/1akps0aANHKWmEzInGVersw7JksuDmcrJ/view?usp=sharing

 # CICD Pipeline implementation Flow

2) https://drive.google.com/file/d/1w-zm-7i45NpIgCYwCEakcVIm7M-2JOJA/view?usp=sharing

 # Entity-diagram

3) https://drive.google.com/file/d/1YUjyDmZCR1Ge2eO-vVotSKdLN8MIWypV/view?usp=sharing

2. Draw the architecture diagram based on the functional requirements and what AWS services are you going to use and why?

You can sketch a high-level architecture diagram that showcases the key AWS services and components, without delving into the specifics of how they will be implemented.

Architecture diagram

1) https://drive.google.com/file/d/1knu4UaFRHCMIkt6sWr2j4eXPcpuG6IVH/view?usp=sharing

AWS Services Used:

1. Route 53: When users try to access the New Social Theory application, their requests are routed through Route 53, which manages DNS and directs them to the correct server.

2. AWS Shield: As users access the application, AWS Shield protects against DDoS attacks, ensuring the application remains available and responsive.

3. AWS WAF: The Web Application Firewall (WAF) protects the application from common web exploits and attacks, such as SQL injection and cross-site scripting (XSS).

4. Amazon CloudFront: CloudFront delivers content, such as celebrity profiles and merchandise images, to users with low latency and high performance.

5. Internet Gateway: The Internet Gateway serves as the entry point to the VPC, allowing users to access the application.

6. Amazon ELB: The Elastic Load Balancer (ELB) distributes traffic across multiple servers, ensuring no single server is overwhelmed and becomes a bottleneck.

7. Amazon EKS: EKS manages and orchestrates containers, allowing for efficient deployment and scaling of application components.

8. Autoscaling: Autoscaling ensures containers are scaled up or down based on demand, maintaining optimal performance and resource utilization.

9. Amazon RDS: The Relational Database Service (RDS) manages user data, celebrity profiles, and other relational data.

10. Amazon Elastic Cache: Elastic Cache provides in-memory caching, improving performance by reducing database query times.

11. S3: S3 stores static assets, such as images and videos, and serves them directly to users.

12. IAM: Identity and Access Management (IAM) controls user authentication and access to application resources.

13. SNS: The Simple Notification Service (SNS) sends notifications and alerts to users, such as updates on celebrity activity.

14. CloudWatch: CloudWatch monitors application performance, logging metrics and performance data.

15. AWS CloudTrail: CloudTrail provides audit and compliance capabilities, tracking API calls and resource changes.

16. API Gateway: API Gateway manages RESTful APIs, handling requests and responses between clients and the application.

17. Amazon Lambda: Lambda runs serverless code, such as data processing and analytics, without requiring server provisioning.

Reasons for choosing AWS services:

1. Scalability and high availability

2. Security and compliance
3. Integration with other AWS services
4. Cost-effectiveness
5. Support for containerized applications (ECS)
6. Robust analytics and monitoring capabilities

3. What tools are you going to use for monitoring and alerting purposes, and how they are going to be integrated with the infrastructure?
You can suggest a few popular monitoring and alerting tools, and provide a general overview of how they could be integrated with the infrastructure, without getting into the technical implementation.

Based on the requirements, I recommend using a combination of monitoring and alerting tools to ensure comprehensive coverage. Here are a few popular tools that can be used:

Monitoring Tools:

1. AWS CloudWatch: For infrastructure and application performance monitoring, log collection, and metric analysis.
2. Prometheus: For containerized application monitoring, metrics collection, and alerting.
3. Grafana: For visualization of metrics and logs from CloudWatch and Prometheus.
4. New Relic: For application performance monitoring, error tracking, and user experience insights.

Alerting Tools:

1. PagerDuty: For alerting, incident management, and notification workflows.
2. AWS CloudWatch Alarms: For triggering alerts based on defined thresholds and sending notifications to PagerDuty.

Integration Overview:

1. CloudWatch:
- Collects logs and metrics from AWS resources (EC2, RDS, ECS, etc.).
- Integrates with Prometheus for additional metrics collection.
- Triggers alerts based on defined thresholds and sends notifications to PagerDuty.

2. Prometheus:
- Collects metrics from containerized applications (Java Spring-Boot microservices).

4. Advise strategies or tools you are going to use to provide the desired security features.
You can recommend some general security strategies and tools that could be considered, without providing a comprehensive security plan.

Based on the requirements, here are some security strategies and tools that can be considered:

Authentication and Authorization:

Devops Assessment

1. OAuth 2.0 with Google and Facebook for secure authentication
2. JSON Web Tokens (JWT) for stateless authentication
3. Role-Based Access Control (RBAC) for authorization

Data Encryption:

1. Transport Layer Security (TLS) for encrypting API calls
2. HTTPS for secure communication between clients and servers
3. Encryption at rest using AWS Key Management Service (KMS)

Rate Limiting and IP Blocking:

1. NGINX or AWS API Gateway for rate limiting
2. IP blocking using AWS WAF or NGINX

Security Monitoring and Alerting:

1. AWS CloudWatch for monitoring and alerting
2. Prometheus and Grafana for monitoring and visualization
3. PagerDuty for incident management and notification

Secure Development Practices:

1. Code reviews and pair programming
2. Secure coding practices and guidelines
3. Regular security audits and penetration testing

Some recommended security tools include:

1. AWS IAM for identity and access management
2. AWS Cognito for user identity and access management
3. OWASP ZAP for security testing and vulnerability assessment
4. SonarQube for code analysis and security scanning
5. Burp Suite for web application security testing
6. Dependancy check
7. OPA Conftest
8. Trivy Scan

Note that this is not an exhaustive list, and a comprehensive security plan should be developed based on the specific requirements and risk assessment of the application.

5. If you can suggest some tools or strategies for the management and maintenance of the data and infrastructure from a longer-term perspective.
You can suggest some high-level approaches and tools for managing and maintaining the data and infrastructure over the long term, without going into the specific

implementation details.

Here are some high-level approaches and tools for managing and maintaining the data and infrastructure over the long term:

Data Management:

1. Data Warehousing: Amazon Redshift, Google BigQuery, or Snowflake for storing and analyzing large datasets.
2. Data Lakes: Amazon S3, Azure Data Lake Storage, or Google Cloud Storage for storing raw, unprocessed data.
3. Data Governance: Apache Atlas, AWS Glue
4. Backup and Recovery: Regular backups to Amazon S3, Azure Blob Storage, or Google Cloud Storage, with automated recovery processes.

Infrastructure Management:

1. Infrastructure as Code (IaC): AWS CloudFormation, Azure Resource Manager, or Terraform for automating infrastructure provisioning and management.
2. Container Orchestration: Kubernetes, Docker Swarm, or Apache Mesos for managing containerized applications.
3. Monitoring and Logging: Prometheus, Grafana, and ELK Stack (Elasticsearch, Logstash, Kibana) for monitoring and logging.
4. Automation and Scripting: Ansible, SaltStack, or PowerShell for automating infrastructure configuration and maintenance.

Scalability and Performance:

1. Auto Scaling: AWS Auto Scaling, Azure Autoscale, or Google Cloud Autoscaling for dynamically scaling resources.
2. Load Balancing: AWS ELB, Azure Load Balancer, or Google Cloud Load Balancing for distributing traffic.
3. Caching: Redis, Memcached, or Amazon ElastiCache for improving application performance.
4. Content Delivery Network (CDN): AWS CloudFront, Azure CDN, or Google Cloud CDN for distributing static content.

Security and Compliance:

1. Identity and Access Management (IAM): AWS IAM, Azure Active Directory, or Google Cloud IAM for managing access and identities.
2. Security Information and Event Management (SIEM): Splunk, ELK Stack, or Azure Sentinel for monitoring and analyzing security events.
3. Compliance Management: AWS Compliance, Azure Compliance, or Google Cloud Compliance for managing compliance requirements.

Devops Assessment

These are just a few examples of tools and strategies for managing and maintaining data and infrastructure over the long term. The specific tools and approaches will depend on the requirements and constraints of the project.