|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ROLLOUT RANGERS | CSE-D | 5TH SEM | | |  |  |
| OLX CLONE | | | | | | |
| Project Summary | | | | | | |
|  | | | | | | |
| Report Date | | Project Name | Project Manager | | |
| 11/09/2024 | | Continuous Integration, Delivery, and Deployment In AWS, Netlify, GitHub ,Dockers | Yogesh.B.G | | |
| EXECUTIVE SUMMARY | | | | | | |
|  | | | | | | |

Deploying an OLX clone on AWS involves leveraging various cloud services to ensure scalability, security, and reliability. Here’s a high-level summary of the deployment:

1. Architecture Overview

Frontend: Hosted on Amazon S3 and served through Amazon CloudFront for fast, global delivery and caching.

Backend: Hosted on Amazon EC2 or AWS Elastic Beanstalk to simplify deployment, scaling, and management.

Database: Amazon RDS (e.g., MySQL or PostgreSQL) or Amazon DynamoDB for scalable and managed database services.

Storage: Amazon S3 for image storage, allowing easy integration with CDN and scalable storage for large files like photos or videos of listings.

2. Core AWS Services Used

Compute: EC2 instances or AWS Elastic Beanstalk to manage backend services and APIs.

Storage: S3 for storing images and user-uploaded files, with high availability and durability.

Database: Amazon RDS or DynamoDB to manage structured data like user information, ad details, and transactions.

Content Delivery: Amazon CloudFront to reduce latency and enhance the speed of asset delivery globally.

Authentication & Authorization: AWS Cognito for user sign-up, login, and secure management of profiles.

3. Supporting Services for Enhanced Functionality

Amazon SQS: Message queue service to handle background tasks, such as sending notifications or processing ad approvals.

Amazon SNS: For notifications, alerting users about updates, message notifications, or admin alerts.

AWS Lambda: For running serverless functions, such as processing images or sending emails.

Amazon Recognition (optional): To automatically moderate images, detecting inappropriate content in listings.

4. Security Measures

AWS IAM: Manage user permissions and control access to AWS services.

SSL/TLS Certificates: Managed through AWS Certificate Manager for secure HTTPS access.

Amazon Guard Duty: Continuous security monitoring for suspicious activity and potential threats.

Data Encryption: S3 bucket encryption and RDS encryption for data at rest, SSL for data in transit.

5. Scalability and Availability

Auto Scaling: Set up auto-scaling for EC2 instances to handle traffic spikes.

Elastic Load Balancer (ELB): Distribute incoming traffic across multiple EC2 instances.

Multi-AZ Deployment: RDS with Multi-AZ configuration for high availability and failover support.

6. Monitoring and Logging

Amazon CloudWatch: Monitor application health, CPU usage, request latency, and other critical metrics.

AWS CloudTrail: Track API calls for security audits and compliance monitoring.

AWS Elastic Load Balancer Logs and Access Logs: Keep track of traffic patterns, user access logs, and server load.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Project Overview | | | | | | | | |
|  | | | | | | | | |
| task | % Done | Due date | | | DevOps Facilitator | | Milestones |
| Planning Stage | **10%** | **06.11.2024** | | | **Mr.Vishal.N** | | **Research and Development** |
| Development Stage | **20%** | **06.11.2024** | | | **Mr.Vishal.N** | | **Research and Development** |
| Testing Stag | **20%** | **07.11.2024** | | | **Mr.Vishal.N** | | **Research and Development** |
| Deployment Stage | **20%** | **08.11.2024** | | | **Mr.Vishal.N** | | **Research and Development** |
| Monitoring Stage | **20%** | **09.11.2024** | | | **Mr.Vishal.N** | | **Research and Development** |
| Feedback Stage | **10%** | **09.11.2024** | | | **Mr .Vishal.N** | | **Research and Development** |
|  |  |  | | |  | |  |
| man-hours | | | | | | | | |
|  | | | | | | | | |
| category | spent | | % of total | | | on track? | notes |
| **Planning and Assessment** | **5** | | **13.9** | | | **YES** | **Completed** |
| Requirements gathering: | **2** | | **5.5** | | | **YES** | **Completed** |
| Application assessment: | **1.5** | | **4.1** | | | **YES** | **Completed** |
| DevOps strategy planning | **1.5** | | **4.1** | | | **YES** | **Completed** |
| Tool selection and configuration | **2** | | **5.6** | | | **YES** | **Completed** |
| **Infrastructure Setup** | **9** | | **25** | | | **YES** | **Completed** |
| Cloud infrastructure setup (AWS/Azure/GCP) | **3** | | **8.3** | | | **YES** | **Completed** |
| Containerization (Docker): | **2** | | **5.5** | | | **YES** | **Completed** |
| Orchestration (Kubernetes) | **2** | | **5.5** | | | **YES** | **Completed** |
| Monitoring and logging setup | **2** | | **5.5** | | | **YES** | **Completed** |
| **Application Integration** | **7** | | **19.4** | | | **YES** | **Completed** |
| Code repository setup (Git) | **1** | | **2.7** | | | **YES** | **Completed** |
| Continuous Integration/Continuous Deployment (CI/CD) pipeline setup | **2** | | **5.5** | | | **YES** | **Completed** |
| Automated testing setup | **1.5** | | **4.1** | | | **YES** | **Completed** |
| Vulnerability management | **1** | | **2.7** | | | **YES** | **Completed** |
| **Security and Compliance** | **1.5** | | **4.1** | | | **YES** | **Completed** |
| Deployment automation | **4** | | **11.1** | | | **YES** | **Completed** |
| Security assessment | **1** | | **2.7** | | | **YES** | **Completed** |
| Compliance setup | **0.5** | | **1.3** | | | **YES** | **Completed** |
| Access control and identity management | **0.5** | | **1.3** | | | **YES** | **Completed** |
| **Testing and Quality Assurance** | **5** | | **13.9** | | | **YES** | **Completed** |
| Test planning | **1** | | **2.7** | | | **YES** | **Completed** |
| Test execution | **1.5** | | **4.1** | | | **YES** | **Completed** |
| Defect tracking and resolution: | **1** | | **2.7** | | | **YES** | **Completed** |
| Quality assurance | **1.5** | | **4.1** | | | **YES** | **Completed** |
| **Deployment and Maintenance** | **4** | | **11.1** | | | **YES** | **Completed** |
| Deployment planning | **1** | | **2.7** | | | **YES** | **Completed** |
| Deployment execution | **1** | | **2.7** | | | **YES** | **Completed** |
| Post-deployment monitoring | **1** | | **2.7** | | | **YES** | **Completed** |
| Maintenance and support | **1** | | **2.7** | | | **YES** | **Completed** |
|  |  | |  | | |  |  |
| STAKEHOLDERS | | | | | | | | |
|  | | | | | | | | |
| STAKEHOLDER | | | | USN | | | KEY RESPONSBILITY AREA |
| Yogesh B G | | | | 4NI22CS255 | | | **Developer and Team Lead** |
| Chirag R Gowda | | | | 4NI22CS263 | | | **UI/UX Designer** |
| Srujan U | | | | 4NI22CS217 | | | **Report and Conclusions** |
| Shravan H R | | | | 4NI22CS201 | | | **Test Cases** |
| Yogeshwar R | | | | 4NI22CS264 | | | **Assistant Developer** |
| Yadunandan K | | | | 4NI22CS251 | | | **Assistance in backend** |
| Thejesh | | | | 4NI23CS422 | | | **Helper** |
| Project Overview | | | | | | | | |
|  | | | | | | | | |

The DevOps project aimed to improve the efficiency, reliability, and scalability of OLX software development and deployment processes. The project focused on implementing DevOps practices, automating CI/CD pipelines, and ensuring continuous monitoring and feedback.

|  |
| --- |
| Key Objectives: |
| **To streamline the development, deployment, and maintenance processes for an OLX clone application, enabling rapid feature releases, efficient collaboration, and stable, high-quality service delivery.** |

|  |
| --- |
| Benefits: |
| **1.Faster Time to Market: Continuous integration and delivery (CI/CD) pipelines reduce the time required to push updates, allowing the team to launch new features quickly and respond to user feedback.**  **2. Improved Collaboration: Automated workflows and shared environments foster a collaborative culture between development and operations, improving efficiency and team cohesion.**  **3. Increased Reliability: Automated testing and monitoring ensure that the application remains stable and performs optimally, reducing downtime and providing a seamless experience for users.**  **4. Scalability: Using infrastructure-as-code (IaC) and containerization makes it easier to scale the application in response to increasing user demands.**  **5. Enhanced Security: By integrating security checks within CI/CD pipelines, DevOps practices improve vulnerability detection and compliance adherence.** |

|  |
| --- |
| Lessons Learned: |
| **1.Importance of Automation: Automating repetitive tasks (e.g., testing, deployment) saves time and reduces human error, which is crucial for handling frequent updates.**  **2. Value of Consistent Monitoring: Setting up comprehensive monitoring (using tools like Prometheus or Grafana) is key to identifying issues early, minimizing downtime, and understanding user interactions.**  **3. Need for Team Alignment: Success in DevOps depends on continuous collaboration between teams, and the adoption of a shared responsibility model is essential for smoother operations.**  **4. Efficient Rollbacks: Automated rollbacks in case of failed deployments are crucial for maintaining service stability and avoiding user disruption.**  **5. Gradual Implementation of DevOps: Migrating to DevOps gradually (starting with CI/CD, then IaC) is effective, as it allows teams to adapt and optimize processes without overwhelming the workflow.** |

|  |
| --- |
| Future Recommendations: |
| **1. Advanced Testing Automation: Expand the test suite to include end-to-end (E2E) testing and performance testing, ensuring the OLX clone maintains performance and functionality even under heavy loads.**  **2. Implement A/B Testing: Add A/B testing capabilities in the CI/CD pipeline to test new features on small user segments, enhancing decision-making based on user behavior.**  **3. Focus on Observability: Use more advanced observability tools to gain deeper insights into system performance, user behavior, and potential issues.**  **4. Expand Infrastructure as Code (IaC): Further automate infrastructure management by implementing IaC for environment consistency and streamlined scaling across regions.**  **5. Security Enhancements: Integrate security as code (SaC) practices, automating compliance checks and adding vulnerability scans to each CI/CD pipeline stage.** |

|  |
| --- |
| Conclusion: |
| **DevOps significantly enhances the development and delivery process of the OLX clone project, providing faster deployments, high reliability, and a scalable infrastructure. While implementation requires cultural shifts and initial effort, the benefits of improved efficiency, collaboration, and stability are well worth it.** |

|  |
| --- |
| Metrics: |
| **To measure the effectiveness of DevOps in the OLX clone, use the following metrics:**   1. **Deployment Frequency: Tracks the number of deployments per day/week, reflecting speed and efficiency.** 2. **Change Lead Time: Measures the time taken from code commit to production, indicating pipeline efficiency.** 3. **Mean Time to Recovery (MTTR): Time taken to recover from failures, showing reliability and the effectiveness of the rollback process.** 4. **Change Failure Rate: Percentage of changes that lead to failures, indicating the quality of automated testing.** 5. **System Uptime: Percentage of time the system remains operational, reflecting reliability and monitoring effectiveness.** 6. **User Engagement and Feedback: Tracks user satisfaction and engagement levels after each release, highlighting the impact of new features.** |