Implementation for OWL270

ONLINE WORKS FOR LIFE

About Customer: Online Works For Life (OWL270)

OWL270 given a dual responsibility to help school students by being their best teaching platform for their curriculum and also for their skill development.
I want to be unique like how all students are different in their approach, likewise amongst many other Edutech companies all over India, OWL270 is an OTT platform like Netflix and Amazon, the only difference is our exclusivity is for education only. I will be a bridge for easy learning to my students.

Customer Challenge

OWL270 is an OTT based application to run on the AWS cloud. OWL270 has designed all the backend services as a microservice applications. Initially there were not that much change in the API development/enhancement however as soon as business grow developer need to cope up with the frequent functional changes and customer had no CICD pipeline initially, they had faced lots of trouble in terms of code management, test, build and deployment manually. Some more requirements that the customer came to us where:

- Automated Build, Deployment and Rollback was required to reduce manual intervention by developers in production. The changes to both the application and infrastructure was desired to be automated.
- To have high availability and fault-tolerant architecture on the cloud within a region.
- Applications to be managed and monitored 24/7 with a complete dashboard view of AWS Cloud.

To address these requirements and Challenges, OWL270 engaged with Workmates to implement the application infrastructure on the AWS Cloud and also make it DevOps compliant with secure, high-performing, resilient, and efficient infrastructure for their applications.

Our DevOps Solution Approach

Workmates DevOps team helped adopt the DevOps methodology to their software development and release processes. The application architecture for the AWS cloud was designed with the right balance of AWS services to reduce operational overhead while keeping the costs low. Most of the undifferentiated activities on the cloud was automated, thus reducing the overall time and risk in deployment of new services. The key aspects of the solution design includes:

- Micro services based architecture for the web application and API's using) Amazon ECS
- Automated provisioning of the infrastructure Including Network, ECS , ASG and Launch Configurations on the AWS cloud using CloudFormation
- A fully featured CI CD pipeline using GitHub Action is been setup. Automatic Rollback if the deployment fails has been enabled on the ECS Service deployment configuration.
- ECR used as Docker container registry and the Authentication to the registry done via IAM role.

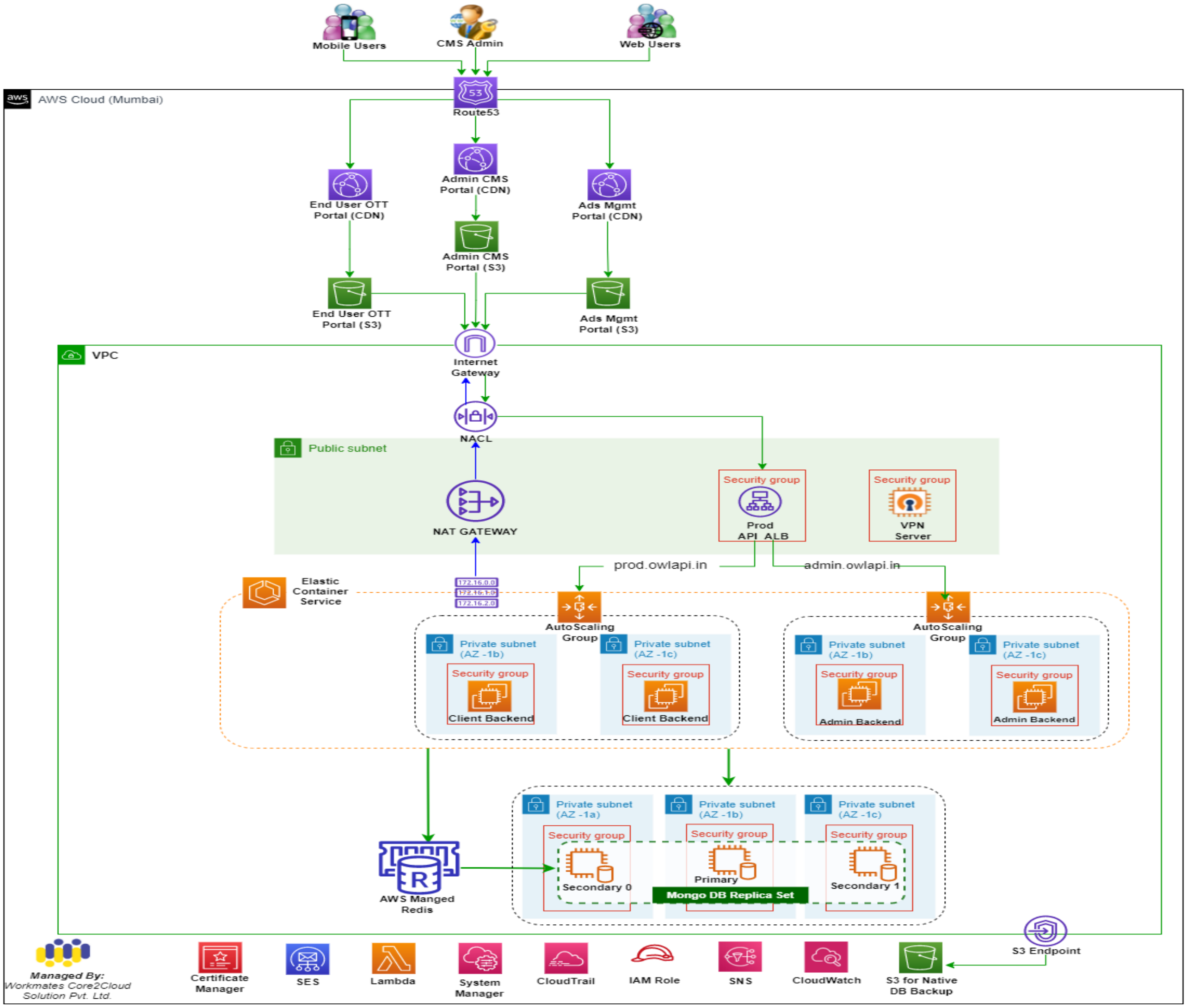
- Infrastructure and applications monitoring enabled using AWS CloudWatch Container Insights.
- The Standard output and error logs for the micro service running on the containers are managed on the centralized Cloud Watch Logs.
- To handle extreme spikes in traffic Service Autoscaling has been enabled with Target Tracking Scaling Policies with ALBRequestCountPerTarget
- Applications Load Balancer with SSL termination has been setup to distribute the applications traffic.
- All the Infra and DB backup are stored on S3 storage service with 15 days retention and can be accessible at any point of time.
- Highly available Mongo DB cluster setup on the EC2 with 3 Node RelicaSet with 1 Primary and 2 Seconday(on Different AZs)
- AWS Config setup for Continuous Monitoring, assessment and change management for the AWS resource's configurations.
- SSM Patch Manager configured to scan EC2 instances and report compliance on a schedule, install available patches on a schedule, and patch or scan instances on demand.

Key Services Used

AWS Services Used	Use Case
AWS ECS	Managed Container Orchestration Service
AWS EC2	Container Workload Management, and MongoDB Instances.
AWS Application Load balancer	Load Balancing and Traffic Management across the containers
S3	Object Storage for Media related Contents
Amazon EBS GP2	Persistent Storage for Stateful apps such as DB
AWS KMS	For EBS and S3 encryption
AWS ECR	Container Registry for docker images
Amazon CloudFront	Content Delivery Network
AWS CloudFormation	Creation of VPC, EKS cluster and ASG
Amazon CloudWatch	Infrastructure and Workload monitoring and also for Logging Solution for all microservice applications
AWS Config	Conduct assessment and audit of the AWS resources
AWS Systems Manager	For On demand Patching EC2 Servers.

Third-Party Tools	UseCase
GitHub Actions	Build, test, and deploy the microservices using ci.yaml file
GitHub	Software Version Control
OpenVpn	Secure access to AWS Infrastructure

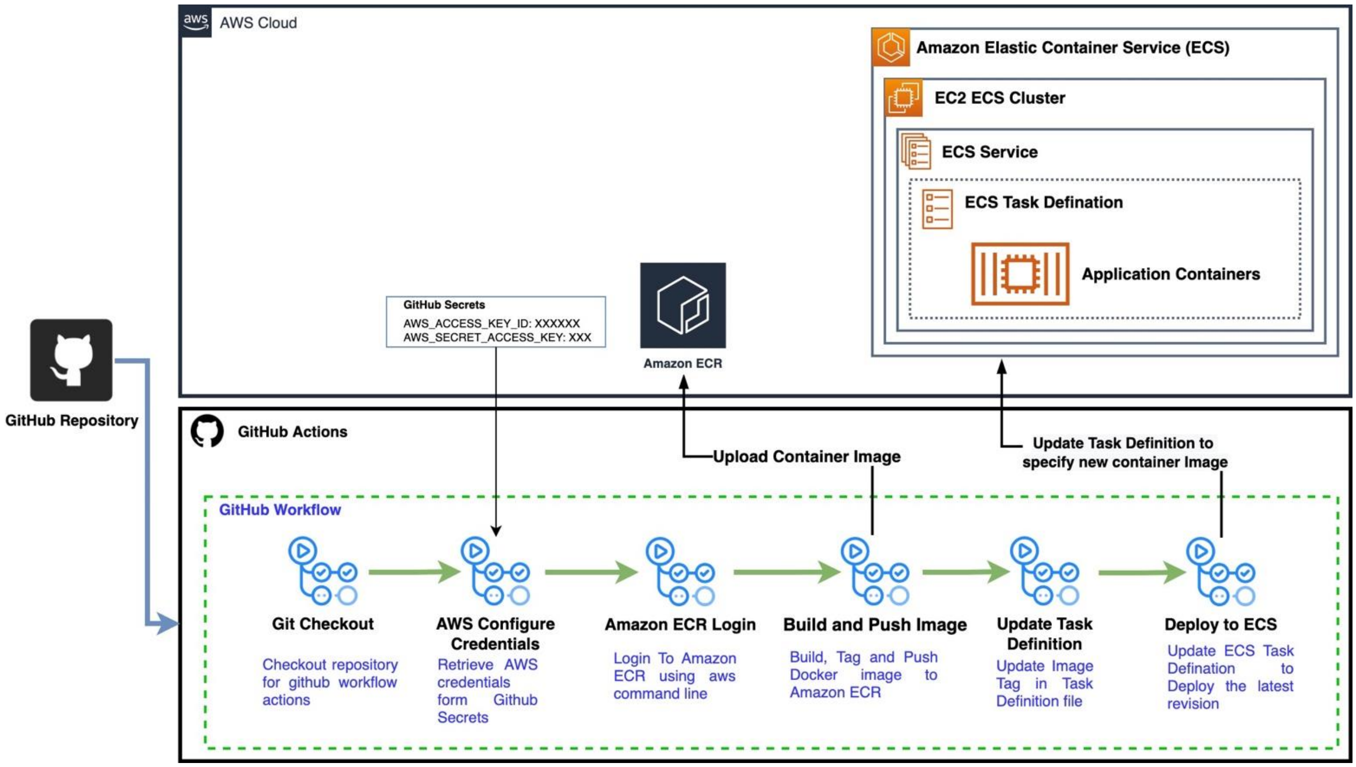
Solution Architecture



Security Considerations

- AWS IAM role based access control to restrict users to only the required resources.
- Deep visibility into API calls are maintained through AWS Cloud Trail, including who, what, and from where calls were made. All user related activities are tracked and logged.
- For any Administrative task Remote user have need to connect to VPN client for accessing the servers. All the RDP/SSH port are bound with OpenVPN server, also default ports will be changed to the custom port.
- The DB ports are accessible only from the Application containers and are restricted using Security Group.
- All the container workloads are under the private subnets, the microservices are exposed using the Application Load Balancer. SSL listeners has been setup for ALB and certificate has been issued using AWS ACM.
- Encryption enabled both in rest and transit using KMS and ACM
- The AWS credentials are made part of GitHub Environmental Variables and are not exposed during CICD pipeline.

Continuous Integration and Continuous Deployment


AWS Cloud
Amazon Elastic Container Service (ECS)
EC2 ECS Cluster
ECS Service
ECS Task Defination
Application Containers
GitHub Secrets
AWS_ACCESS_KEY_ID: XXXXXX
AWS_SECRET_ACCESS_KEY: XXX
Amazon ECR
GitHub Repository
GitHub Actions
Upload Container Image
Update Task Definition to specify new container Image
GitHub Workflow
Git Checkout
Checkout repository for github workflow actions
AWS Configure Credentials
Retrieve AWS credentials form Github Secrets
Amazon ECR Login
Login To Amazon ECR using aws command line
Build and Push Image
Build, Tag and Push Docker image to Amazon ECR
Update Task Definition
Update Image Tag in Task Definition file
Deploy to ECS
Update ECS Task Defination to Deploy the latest revision

The build of any microservices is triggered with commit on GitHub. The CICD workflow is written in yaml (.github/workflows/aws.yaml) and is a job containing series of steps. And GitHub provides a visualization on which it is very easy to visualize the current step of the pipeline. The CICD steps of pipeline includes the following:

1. The Reviewer merges the code into the GitHub repository’s master branch, this triggers the initial stage of the pipeline i.e Git checkout, and here source code is pushed into the $GITHUB_WORKSPACE
2. Since all the apps are NodeJS apps , there is no need to compile the code. So the next step is running the Build.
3. Now we tag the docker image obtained from the previous stage and push it to the Amazon ECR. For ECR login the AWS credentials are configured by setting Access and Secret Key directly from the GitHub Variables (restrict passing of AWS credentials directly from the pipeline) and We have also enabled ECR scan on push for scanning any security vulnerabilities on the image.
4. In this step we will have to update the task definition file with the latest image tag that we built and pushed in the previous step.
5. One the Task Definition file is updated the final step is to create a revision of the task definition and update the service with the latest revision of the task definition.
6. GitHub Actions has a notification enabled , that if the workflow fails the email notifications is received. Choose how you want to receive workflow run updates for repositories that you are watching that are set up with GitHub Actions.

Results and Benefits

OWL270 OTT and backend application was successfully deployed on AWS environment while meeting all security & high availability guidelines as per the stated compliance directives. The following are some of the key benefits for the customer

1. Micro services based application architecture allowed for modular development and smaller frequent releases.
2. Enhanced monitoring and alerting capability from the Amazon CloudWatch will notify the DevOps team on any production issues so they can mitigate it immediately.
3. The automated deployment of the AWS stack and code has freed developers of infrastructure administration and scaling tasks
4. The overall security posture on the cloud is improved using cloud native security features like encryption and private networks and continuous compliance using AWS Config.
5. AWS native security features are highly secured and data/secrets were encrypted using Asynchronous customer managed key (CMK) and remediating the noncompliant AWS resources using AWS Config service.
6. Lead Time for Changes is very fast and efficient. Also reduces the time, cost, human effort, Maintenance time
7. Faster MTTR (Mean Time to Repair) using automated rollback.

Workmates Core2Cloud

About Workmates

Workmates core2cloud Solution Pvt. Ltd is an AWS Advance consulting partner and Leading Cloud Management Company in Eastern India. Workmates Core2cloud is a cloud managed services company focused on AWS services, the fastest growing AWS Advanced Consulting Partner in India. We focus on Managed services, Cloud Migration and Implementation of various value-added services on the cloud including but not limited to Cyber Security and Analytics. Our skills cut across various workloads like Microsoft, SAP, Media Solutions, DevOps ,E-commerce, Analytics, IOT, Machine Learning, VR, AR etc. Our VR services are transforming many businesses.