Contents

[1 MediaWiki 2](#_Toc48593366)

[2 Proposed Solution architecture on AWS 2](#_Toc48593367)

[2.1 Proposed Deployment Architecture 2](#_Toc48593368)

[2.1.1. Architecture Overview 2](#_Toc48593369)

[2.1.2. Kubernetes Configurations 3](#_Toc48593370)

[3 Tools used Proposed approach 3](#_Toc48593371)

[3.1 AWS Services 3](#_Toc48593372)

[3.2 Third Party tools 3](#_Toc48593373)

[3.3 Operations Management 3](#_Toc48593374)

[3.3.1. Alerts and Monitoring 3](#_Toc48593375)

[3.4 Data protection 3](#_Toc48593376)

[3.3.2. Protection for data in transit 3](#_Toc48593377)

[4 Key Considerations 3](#_Toc48593378)

[5 Recommendations 3](#_Toc48593379)

[6 Project risks 4](#_Toc48593380)

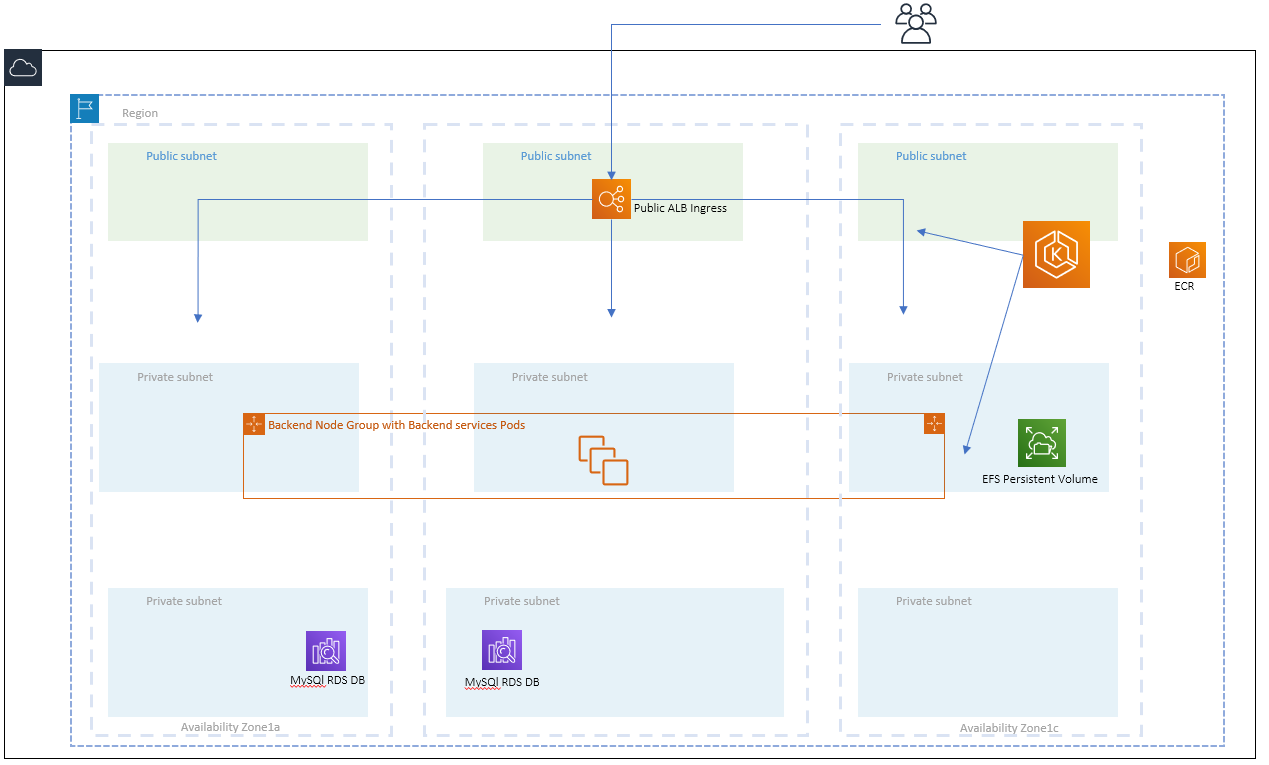
[7 Pre-requisites 4](#_Toc48593381)

# **MediaWiki**

MediaWiki is an open source collaboration and documentation platform widely used by Startups and Mid-Sized organizations.

# **Proposed Solution architecture on AWS**

## **Proposed Deployment Architecture**



# **Architecture Overview**

* A VPC will be created in Mumbai region across three availability zones.
* The network will be divided into three-tier architecture.
* Public Subnet Layer
* Private Subnet Layer
* Private DB Subnet Layer
* An EKS control plane version v1.17 will be deployed in three availability zones with accessibility across public and private subnet layers.
* AWS ALB will be used as in ingress resource to expose microservices.
* ALB ingress controller will annotate and configure SSL/TLS on ALB ingress resource.
* Metrics server will be deployed for Horizontal Pod Autoscaling.
* AWS EFS will be used as persistent volume claim for the pods require data to be persist.
* NAT Gateway will be used in VPC for outbound traffic on servers placed in private subnets.
* AWS CloudTrail, Config and IAM will be used to implement security and logging for audit purpose.
* Security groups and NACLS will be used to implement firewall policy.
* VPC flow logs will be enabled to capture the IP traffic going to and from network interfaces.

# **Kubernetes Configurations**

* + Role based access will be used to access EKS Cluster and to create, update, delete objects in Kubernetes.
  + Kubernetes ConfigMaps will be used for storing application properties and environment variables.
  + Kubernetes secrets will be used for storing confidential data like keys, database credentials in base64 encoded format.
  + Separate namespace will be used for dev/stage environment.
  + Metrics server will be deployed for enable horizontal pod autoscaler.
  + Helm Package Manager V3 will be used for deploying Kubernetes objects.

# **Tools used Proposed approach**

## **AWS Services**

* AWS VPC for network infrastructure.
* AWS CloudFormation for infrastructure deployment.
* AWS EC2 for Application and Database servers.
* AWS EKS for containers orchestration
* AWS ECR for container repository.
* Elastic Block Storage for volumes attached to the EC2 instances.
* AWS IAM for Identity and Access Management.

## **Third Party tools**

* Helm
* Docker

## **Operations Management**

# **Alerts and Monitoring**

* Detailed monitoring will be done for all servers using AWS CloudWatch.
* For Pods and Logs monitoring, AWS CloudWatch container insights will be deployed in EKS, which will send logs to CloudWatch log groups.

## **Data protection**

# **Protection for data in transit**

* SSL termination will be done on AWS Load Balancer, so the application communication is in encrypted and secure format.

# **Key Considerations**

# **Recommendations**

* Encryption of data at rest in EBS volumes and S3.
* AWS Managed KMS to be implemented for data encryption.
* Rotation of user passwords and SSH keys.
* MFA delete should be enabled on CloudTrail S3 bucket logs.
* We recommend using AWS Lambda for running backend jobs which does not require more than 15 mins to complete their job.
* We recommend deploying all critical components of the application in HA mode.
* We recommend using AWS WAF for protection against common web exploits like SQL injection, DDoS attacks.
* We recommend using Metrics server for Horizontal pod Autoscaler.
* We recommend the use of Helm Package manager for deploying objects and resources on EKS.
* We recommend opting for RI/Saving Plans, Spot Instances, after reviewing the utilization of EC2 Instances.

# **Project risks**

|  |  |  |
| --- | --- | --- |
| Sr No | Risk | Mitigation Strategy |
| 1 | Use of in-secure container images | Suggested to harden the container images and use as much as non-privileged user inside the container.  Suggested to enable scan-on-push flag in ECR for CVE suggestions. |
| 2 | Data Encryption compliance | Suggested to encrypt EBS volumes and use AWS KMS for managing encryption keys. |

# **Pre-requisites**

Below are the Pre-requisites required to test the deployment of MediaWiki on AWS

1. Cloud the Git repo

*https://github.com/kkumar117/mediawiki.git*

1. Deploy AWS RDS for backend database using *rds.json* CloudFormation template. Restore mediawiki.sql database to test.
2. Deploy AWS EFS for persistent volume claim
3. Docker for building docker image from Dockerfile and pushing it to ECR
4. Deploy EKS Cluster latest version v1.17
5. An ECR repo for storing docker image
6. Worker node group to schedule pod.
7. Install Helm Package manager V3+ to deploy mediawiki helm chart name *wiki*
8. You can use deploy.sh file that will build, push the docker image and install helm chart.

* You can make changes in values.yml for the parameters specific to you environment.

Like: dbname, dbhost, imagetag

Notes:

1. deploy.sh file will ask for user input regarding the image\_tag and env.
2. Above deployment is not in Highly Available environment as MediaWiki require some state to maintain user data and persistent files also.

We can scale the deployment using centralising the user data like session management, caching in a separate in-memory caching engine called Memcached This will also help us to achieve resiliency in the application and improve performance.