Microsoft

Azure Well-Architected Framework Review

Azure Well-Architected Framework (WAF) review recommendations for Customer's platform

8-Sep-21

Version 1.0 Draft

Prepared by

[list of architects]

Reviewed by

[list of architects]

Tech Executive Sponsor

[List of sponsors]

Table of Contents

1 Introduction

This is an architecture recommendations report from Microsoft for implementing [Their Solution]. This report is based on the information shared by Partner team.

1.1 Requirements & current state

Partner team has shared following requirements and current state with Microsoft team.

- Platform runs on 12 VM instances out of which 7 instances are public facing.
- Platform is mainly hosted using Nginx web server and runs on VMs, AKS and App Service in Azure. Webserver is hosted on Linux.
- Platform components - Account Manager, API, Authing tool, Config Manager, Delivery Platform and Report Service (this last one only internal interface) runs on VMs.
- Platform components - Assessment Builder, Assessment Builder API, and internal services Assessment Service and Itembank Service runs on AKS.
- Platform components - API runs on App Service and is accessed internally only.
- Platform is written using primarily C#, PHP and Angular Framework (a front-end JavaScript framework) programming languages.
- Platform uses nginx and PHP-FPM to front-end tier VMs. Nginx is also used as ingress controller in AKS.
- Platform stores logs on each VM and is recollected using New Relic agents, for the AKS cluster is still not done.
- Platform stores session data in two MySql databases - Account Manager and Delivery Service.
- Platform uses OAuth2/JWT for authentication by custom login provided by the Account Manager.
- Platform uses MySQL, Postgres and MongoDB for CRUD operations.
- Platform uses Redis Service for caching.
- Platform plans to use Azure Service Bus for brokered messaging.
- Platform functionality is compute and memory intensive. However, this characterization is not factored in capacity planning.
- Platform doesn't undergo regular benchmark and stress testing.
- Platform is being used by 100 unique visitor every hour on average.
- Platform serves 3M requests a month (i.e., 1 RPS). This is expected to grow next year.
- Platform has peak request time during the PST daytime.
- Platform has availability SLA of 99.9%.

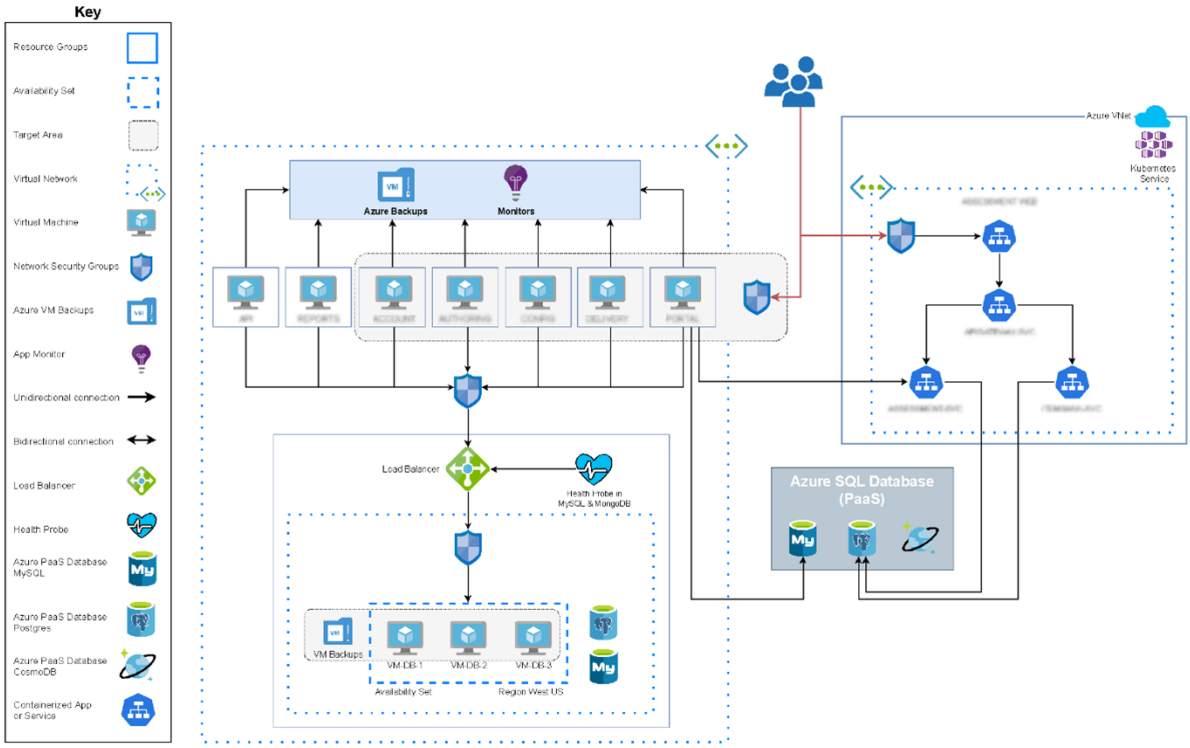
- Platform doesn't have documented Mean Time Between Failures (MTBF) and Mean Time Between Repair (MTBR) requirements.
- Platform doesn't have documented Recovery Point Objective (RPO) and Recovery Time Objective (RTO) requirements.
- Platform doesn't have documented end-to-end request latency requirements.
- Platform doesn't use any shared services e.g., firewall, DNS, etc.
- Platform doesn't use any Web Application Firewall. However, ACL is configured on Network Security Group for each VM.
- Platform doesn't use any up/down or in/out scaling.
- Platform uses Jenkins as a CI/CD system.
- Platform components aren't isolated for easier management, resourcing, or permissions.
- Platform uses Azure AD for RBAC.
- Platform undergoes 3rd party audit for security vulnerability scanning.
- Platform uses SSL for end-to-end TLS protection.
- Platform uses manual ACLs for network traffic restrictions.
- Platform uses Jenkins Credential Vault to secure *connectionstring* and other secure data.
- Platform doesn't assign identity to running web apps.
- Platform doesn't control/inspect egress traffic.
- Platform doesn't have process to patch underlying servers/platform.
- Platform doesn't use any security management solution.
- Platform uses New Relic to aggregate logs from infrastructure as well as application in some cases.
- Platform doesn't track or enforce organizational policies.
- Platform uses partially stateful applications.
- Platform doesn't assign any resource quota/resource limits.
- Platform doesn't have multiple instances across other Azure regions.
- Platform undergoes month backup and restore drill.
- Platform uses New Relic dashboard and Azure Insights to monitor web app uptime.
- Platform doesn't use any alternative routing in case an instance becomes unresponsive.
- Platform uses Azure Backup for backing up data and application.
- Platform uses New Relic and Azure Monitor for application monitoring.
- Platform uses New Relic Synthetic and Azure Insights for application health probing.
- Platform uses Azure Virtual Network to deploy VMs close to each other to improve network latency.
- Platform uses Azure Cost Management and CSP Portal to track and manage costs.
- Platform doesn't have documented process to manage internal costing/budgeting.
- Platform uses auto-shutdown for non-production workload VMs to control costs.
- Platform will use Azure Tags to for workload classification.

- Partner needs additional guidance on CosmosDB RUs model, VM autoscaling, Kubernetes migration, caching, etc.

2 Current architecture

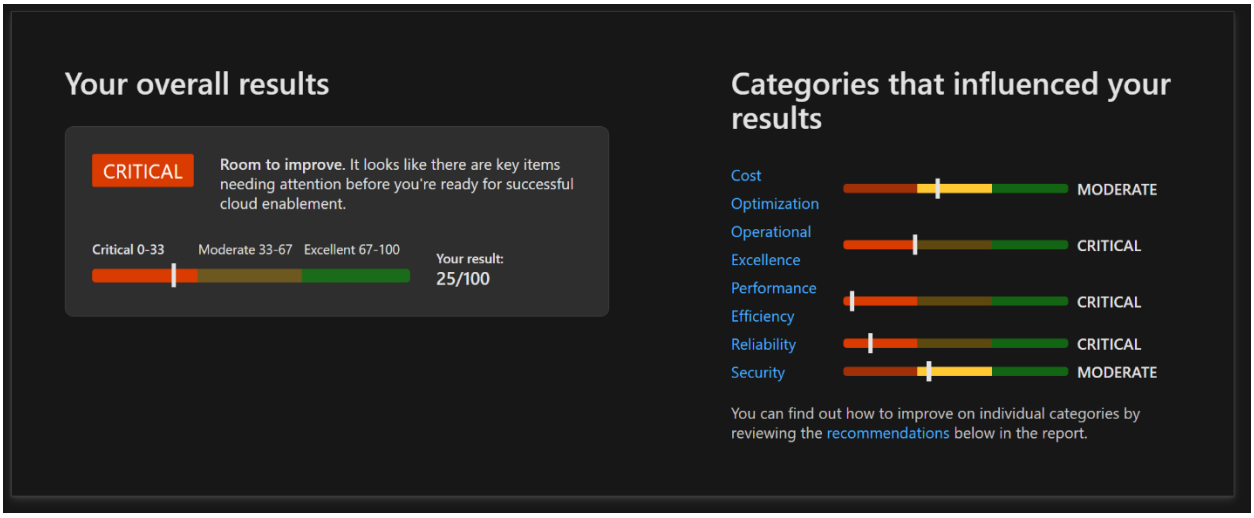
Current application architecture is discussed below.

2.1 Architecture



2.2 Azure Well-Architected assessment

Summary of Architecture assessment on 3-Nov-2020 using Azure Well-Architected framework is provided below.



2.3 Architecture improvement scope

Partner wants Microsoft to recommend improvements in current architecture w.r.t following scope.

1. Reliability
2. Security
3. Performance Efficiency
4. Operational excellence
5. Cost optimization

3 Recommendations

Microsoft recommends following changes to improve current architecture.

3.1 Reliability

Azure Well-Architected assessment provides 49 recommendations to improve reliability. Below is a prioritized list of changes to be implemented immediately.

1. Consider using Azure Application Gateway or Azure Front Door as entry point for user requests so that requests can be load balanced between active instances only.
2. Use multiple Azure regions for deploying application so that there is no single point of failure.
3. Session data store should be deployed on VMs in either Availability Zones or across multiple Azure regions to prevent outage due to issues with sessions data store.
4. Consider using Azure PaaS databases for storing sessions data for in-built high availability.
5. Deploy Jenkins VMs in either Availability Zones or across multiple Azure regions to ensure outage with Jenkins VMs does not block application deployment.
6. Consider using Azure DevOps with committed SLA of 99.9% for test, build and deploy services as well as infrastructure to avoid only VM level SLA associated with Jenkins VMs.
7. Consider using Redis Cache in front of database. This will ensure that Read queries will not hit database thereby reducing database traffic.
8. Consider using external Azure storage for AKS services so that AKS cluster outage doesn't cause data loss.
9. Consider using Premium SSDs or Ultra Disk for OS and data disks on VMs to ensure 99.9% VM connectivity SLA.
10. Consider implementing retry logic in application code that connects with database to handle planned maintenance with Azure managed database services.
11. Consider using geo-replication for Azure Database for MySQL, Azure Database for PostgresSQL and Azure CosmosDB.
12. Consider using Azure Monitor, Azure Log Analytics workspace and Application Insights for consolidate logs across multiple sources for easier visualization, insights and debugging.
13. Perform regular stress testing matching expected peak load and duration to identify and fix issues that may happen when application is under load.
14. Configure Azure Resource Health alerts to immediately identify and respond to availability of an Azure resource.

15. Identify long running operations/queries using Azure Monitor or by writing Log Analytics queries so that they can be tuned to prevent overconsumption of resources.

3.2 Security

Azure Well-Architected assessment provides 39 recommendations to improve security. Below is a prioritized list of changes to be implemented immediately.

1. Consider using Azure Update Management for VM updates to assess status of available updates, schedule installation of required updates, and review deployment results.
2. Configure MFA on all accounts + Azure PIM, that will eliminate 90% of identity attacks. Accounts that are protected only by password are easy targets for phishing and spoofing.
3. Use SSH keys in order to access VMs, avoid using username and password as this method is vulnerable to brute-force attacks.
4. Consider using Azure Web Application Firewall on Azure Front Door to inspect incoming requests at the Azure network edge and protect web app/services from common risks.
5. Use Azure Bastion for system management, it will allow to log into your VMs and avoid public Internet exposure using SSH with private IP addresses only.
6. Implement Azure DDoS protection for enhanced DDoS mitigation features to defend against DDoS attacks.
7. Identity : Disable legacy protocols in order to eliminate attack surface.
8. Implement conditional access to identify, remediate or block risky sign-ins.
9. Use managed Identity for VMs to access SQL database to avoid mixed authentication mode.
10. Configure Pod Identity for AKS in order to utilize managed identity in containers, that way it is possible to avoid using mixed authentication.
11. Extend CI/CD pipeline with automatic Dependency Check for 3rd party components vulnerabilities.
12. Extend CI/CD pipeline with code scanning tools (SATS tools) to identify security flaws.
13. Implement vulnerability management for container images to identify 3rd party components vulnerabilities.
14. Enable Pod Security policies (preview), which allows to define requirements to limit the use of privileged containers, access to certain types of storage, or the user container can run as.
15. Consider AKS network security policies to define rules that control the flow of traffic. For example, database components are only accessible to the application tiers that connect to them.

3.3 Performance Efficiency

Azure Well-Architected assessment provides 28 recommendations to improve performance efficiency. Below is a prioritized list of changes to be implemented immediately.

1. Optimize MySQL database queries by leveraging guidance on Performance Recommendations in Azure Database for MySQL.
2. Consider using a Content Delivery Network (CDN) because it reduces load times and improves speed responsiveness, by using edge servers that provide cached content next to your user's location.
3. Avoid sticky sessions and client affinity by storing state in external stores or distributed caching service such as Azure Cache for Redis.
4. Consider using scheduled scaling when you can predict high peaks before they occur for example with Azure Monitor Workbooks.
5. Use the monitoring tools to understand your performance bottlenecks. Start looking for the most common antipatterns: busy databases and chatty I/O applications.

3.4 Operational excellence

Azure Well-Architected assessment provides 24 recommendations to improve operation excellence. Below is a prioritized list of changes to be implemented immediately.

1. Consider using Azure Monitor for Containers to monitor the performance of container workloads deployed on Azure Kubernetes Service (AKS).
2. Consider using Azure Automation Update Management to assess and manage operating system updates, to create scheduled deployments within a defined maintenance window.
3. Consider performing business continuity drills using Azure Site Recovery to validate apps and data is available during planned and unplanned outages.
4. Consider using code analysis tools (e.g. SonarQube) to analyze the technical debt and identify areas of improvement.
5. Consider reviewing and understanding the existing limits and quotas to plan resource provisioning.
6. Consider using Azure Resource Manager templates to guarantee that the resources are deployed in the correct order, make deployment repeatable, allowing infrastructure auditing and change tracking.
7. Consider moving VMs running web servers to an AppService approach (like using AppService containers for Nginx) to allow scaling based on application needs.

3.5 Cost optimization

Azure Well-Architected assessment provides 13 recommendations to improve cost optimization. Below is a prioritized list of changes to be implemented immediately.

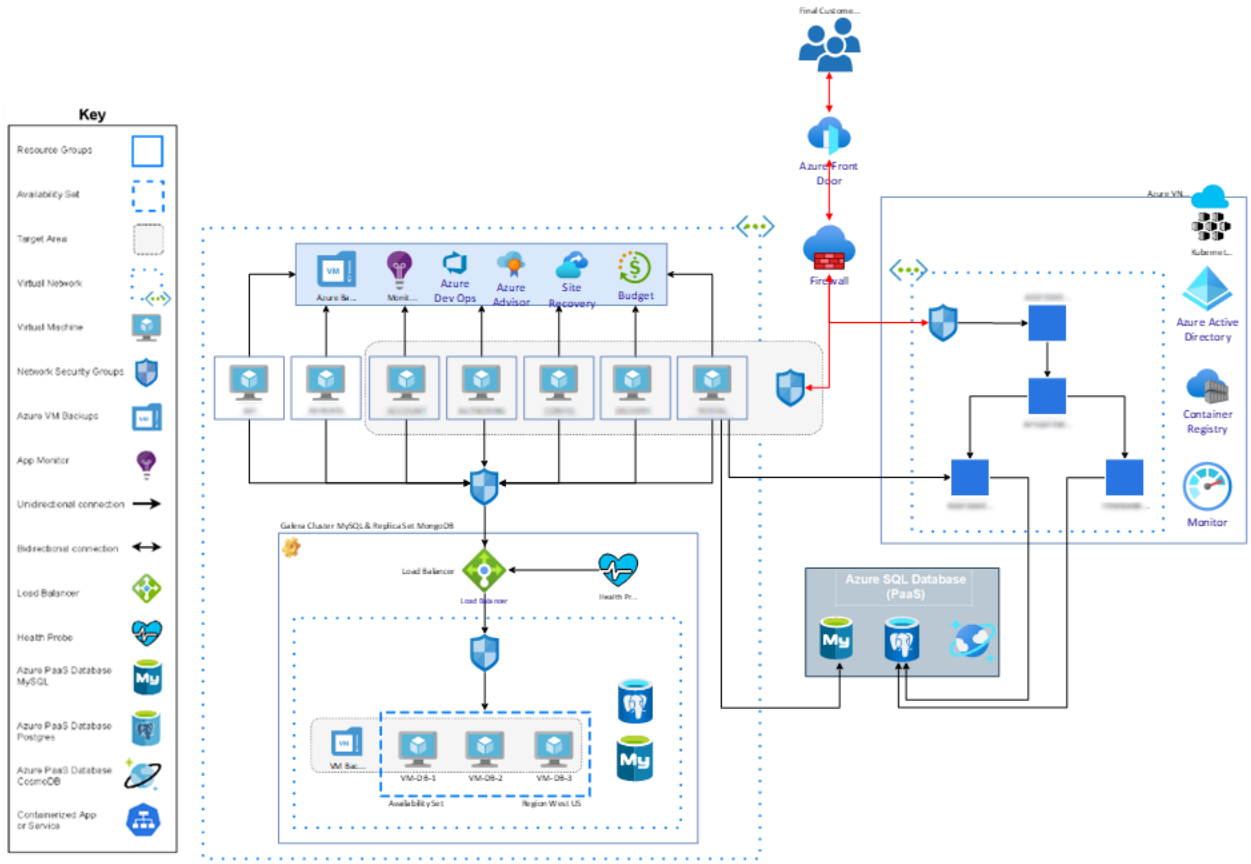
1. Consider creating tags in the Azure Portal to break down costs and get better cost awareness by workload, this will allow you to better design the cost optimization.
2. Create budgets and alerts to get notifications on possible anomalies or events that make costs suddenly fluctuate. Use these notifications for regular cost reviews.
3. Use the database recommendations from Azure Advisor to use the best size for usage patterns and performance needs of the platform.
4. Consider assessing your workloads to configure **auto-scaling policies**, so you can scale in and out for adjusting the cost to the actual demand, for example, in AKS you have the cluster autoscaler to automatically scale VM nodes.
5. Consider the traffic cost of data transfers between billing zones because it impacts in bandwidth costs.

3.6 Proposed Architecture

Diagram below depicts proposed architecture over short and long term.

3.6.1 Short term

Proposed short-term (2 to 4 weeks) architecture changes are depicted below.

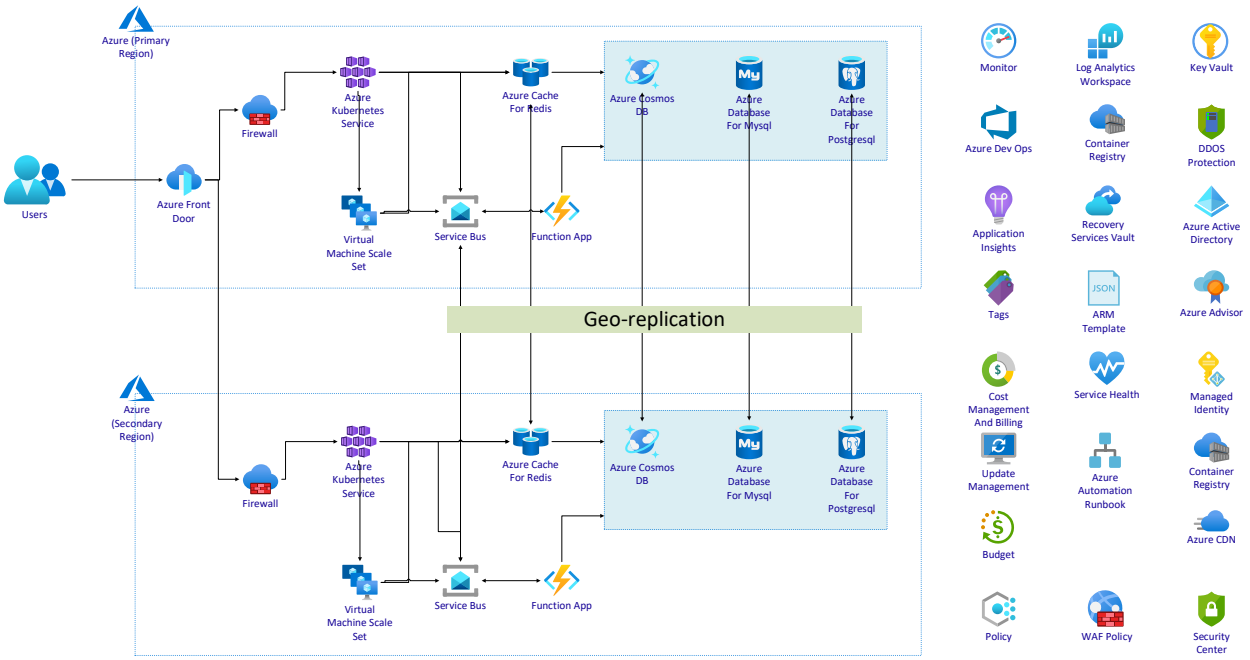


Key highlights of the proposed short-term architecture are as below.

1. Use of Azure Front Door for global load balancing and protection against common vulnerabilities for web applications.
2. Use of Azure Firewall to control all outbound traffic as well as protection against incoming traffic on non-HTTP routes.
3. Use of Azure Monitor for Containers for monitoring container performance and utilization.
4. Use of Azure Active Directory integration with AKS for RBAC and secured access to and from AKS cluster.
5. Use of Azure DevOps, Azure Advisor, Site Recovery and Cost Management to improve operation excellence capabilities.

3.6.2 Long term

Proposed long-term (4+ weeks) architecture changes are depicted below.



Key highlights of the proposed long-term architecture are as below.

1. Multiple Azure regions are used to avoid a single point of failure or contention.
2. Azure Front Door is used as a global load balancer to route requests securely between two Azure regional deployments.
3. Azure firewall is used to inspect and control all outbound traffic from application. It also inspects and control all inbound traffic over non-HTTP protocol.
4. All single instance VMs are replaced with VM Scaleset for high availability and scalability.
5. Azure Redis Cache is used in front of databases to offload database requests and enhance availability and performance.
6. All Azure Database services are set up with global geo-replication for reliability.

3.7 Next steps

Microsoft team recommends following next steps.

1. Implement prioritized recommendations provided in this report across Reliability, Security, Performance Efficiency, Operational excellence, and Cost optimization.
2. Implement recommendations in short-term proposed architecture.
3. Implement recommendations provided in WAF assessment.
4. Implement recommendations in long-term proposed architecture.

4 Resources

Use following resources for additional information.

- Microsoft Azure Well-Architected Framework
- Overview of the reliability pillar
- Overview of the security pillar
- Principles of cost optimization
- Overview of the operational excellence pillar
- Overview of the performance efficiency pillar
- Azure network round-trip latency statistics