Your results

View how your workload aligns to best practices and recommendations to help you improve.

Sign in to save this assessment
Save, share, export, and access all assessments anytime with your Microsoft account.

Sign in

Recommendations for your workload

Actionable items to consider implementing to improve your workload across the five pillars of the Microsoft Azure Well-Architected Framework

Your overall results

CRITICAL **Room to improve.** It looks like there are key items needing attention before you're ready for successful cloud enablement.

Critical 0-33 | Moderate 33-67 | Excellent 67-100

Your result: 7/100

Categories that influenced your results

Reliability	CRITICAL
Security	CRITICAL
Cost Optimization	CRITICAL
Operational Excellence	CRITICAL
Performance Efficiency	CRITICAL

You can find out how to improve on individual categories by reviewing the recommendations below in the report.

Next Steps

Identified and classified business critical applications.

Ensure you have identified and classified the applications in your portfolio that are critical to business functions. Enterprise organizations typically have a large application portfolio, so prioritizing where to invest time and effort into manual and resource-intensive tasks like threat modeling can increase the effectiveness of your...

Review identify and classify business critical applications >

Identified how long the workload can be down for, and how much data it's acceptable to lose in a disaster.

Derive these values by conducting a risk assessment, and make sure you understand the cost and risk of downtime and data loss. These are nonfunctional requirements of a system and should be dictated by business requirements.

Define RPO and RTO for your workload >

Be aware of your resource limits in Azure.

Azure Resource Manager (ARM) enforces limits and quotas on how many resources of each type you can provision per Azure Subscription, and even per Azure Region. Some limits are a hard maximum, while others are a soft limit that can be increases upon request through a support case at no charge. When working with Virtu...

Review limits >

Improve your results

Recommended actions priority: All

Our recommendations for improving your results are organized by category below.

Reliability CRITICAL

41 recommended actions

Show less

Results breakdown

Critical 0-33 | Moderate 33-67 | Excellent 67-100

Your result: 12/100

41 recommended actions

- Automate key rotation
- Automate your tests
- Identify recovery targets for your workload
- Create application specific health probes
- Implement resiliency strategies in your workload
- Develop a plan for region/zone/network outages
- Architect storage for resiliency
- Distribute data geographically
- Segregate read operations from update operations
- Load balance traffic across availability zones
- Collect and store logs and key metrics of critical components
- Simulate a failure path for cross premise connectivity
- Put operational procedures into place for if data size exceeds limits
- Manage load balancer connections to avoid port exhaustion
- Backup keys and secrets in a geo-redudant way
- Decouple the lifecycle of the application from its dependencies
- Create health probes that validate data consistency
- Store session state in an external data store
- Monitor long-running workflows for failures
- Plan for expected usage patterns
- Have redundant network connections to on-prem data sources
- Identify distinct workloads
- Automatically test your failover and failback process
- Create a disaster recovery plan
- Document regional failure plan
- Operate your workload in multiple regions
- Validate Availability Zones are in required regions
- Measure and monitor key availability targets
- Plan for dependent service outages
- Avoid session state
- Perform a failure mode analysis
- Deploy to multiple availability zones
- Have clearly defined availability targets
- Decouple your application services
- Compute a composite SLA for your workload
- Test under expected peak load
- Archive application configuration and installation information
- Detect and remediate faults through chaos engineering

- Implement application throttling
- Perform chaos testing by injecting faults
- Use high availability offerings for platform services

Security CRITICAL

92 recommended actions

Show less

Results breakdown

Critical 0-33 | Moderate 33-67 | Excellent 67-100

Your result: 6/100

92 recommended actions

- Implement threat protection for the workload
- Configure emergency access accounts
- Adopt threat modeling processes
- Restrict access to backend services to a minimal set of public IP addresses, only those who really need it
- Establish a security operations center (SOC)
- Scan container workloads for vulnerabilities
- Establish a detection and response strategy for identity risks
- Classify your data at rest and use encryption
- Adopt a zero trust approach
- Protect all public endpoints with appropriate controls
- Adopt a formal DevSecOps approach to building and maintaining software
- Implement security strategy to contain attacker access
- Implement a branch policy strategy to enhance DevOps security
- Implement Conditional Access Policies
- Implement established processes and timelines to deploy mitigations for identified threats
- Ensure all Azure environments that connect to your production environment/network apply your organization's policy and IT governance controls for security
- Establish lifecycle management policy for critical accounts
- Discover and remediate common risks to improve Secure Score in Azure Security Center
- Data in transit should be encrypted at all points to ensure data integrity
- Establish security benchmarking using Azure Security Benchmark to align with industry standards
- Use only secure hash algorithms (SHA-2 family)
- Conduct periodic access reviews for the workload
- Deprecate legacy network security controls
- Define security requirements for the workload
- Use penetration testing and red team exercises to validate security defenses for this workload
- Build a security containment strategy
- Establish an incident response plan and perform periodically a simulated execution
- Integrate network logs into a Security Information and Event Management (SIEM)
- Use service endpoints and private links where appropriate
- Use Azure Firewall or a 3rd party next-generation firewall to protect against data exfiltration
- Establish a designated group responsible for central network management
- Develop and implement a process to track, triage, and address threats into the application development lifecycle
- Establish a unified enterprise segmentation strategy
- Review, prioritize, and proactively apply security best practices to cloud resources
- Use tools like Azure Disk Encryption, BitLocker or DM-Crypt to encrypt virtual disks
- Implement a landing zone concept with Azure Blueprints and Azure Policies
- Define an access model for keys and secrets
- Develop a security plan
- Designate the parties responsible for specific functions in Azure
- Use Managed Identities for authentication to other Azure platform services
- Implement a solution to configure unique local admin credentials
- Implement just-in-time privileged access management
- Automatically remove/obfuscate personally identifiable information (PII) for this workload
- Implement lifecycle management process for SSL/TLS certificates
- Review and consider elevated security capabilities for Azure workloads
- Store keys and secrets outside of application code in Azure Key Vault
- Standardize on modern authentication protocols
- Configure quality gate approvals in DevOps release process
- Involve the security team in the development process
- Establish a process for key management and automatic key rotation
- Integrate code scanning tools within CI/CD pipeline
- Define a set of Azure Policies which enforce organizational standards and are aligned with the governance team
- Apply security controls to self-hosted build agents in the same manner as with other Azure IaaS VMs
- Clearly define CI/CD roles and permissions
- Remove platform-specific information from HTTP headers, error messages, and web site content
- Follow DevOps security guidance and automation for securing applications
- Mitigate DDoS attacks
- Use standard and recommended encryption algorithms
- Design virtual networks for growth
- Assign permissions based on management or resource groups
- Synchronize on-premises directory with Azure AD
- Develop a security training program
- Configure and collect network traffic logs
- Maintain a list of frameworks and libraries as part of the application inventory
- Implement automated deployment process with rollback/roll-forward capabilities
- Implement identity-based storage access controls
- Add planning, testing, and validation rigor to the use of the root management group
- Configure web apps to reuse authentication tokens securely and handle them like other credentials
- Leverage a cloud application security broker (CASB)
- Identify and classify business critical applications
- Regularly simulate attacks against critical accounts
- Implement security playbooks for incident response
- Ensure security team has Security Reader or equivalent to support all cloud resources in their purview
- Use managed identity providers to authenticate to this workload
- Restrict application infrastructure access to CI/CD only
- Use identity services instead of cryptographic keys when available
- Continuously assess and monitor compliance
- Establish process and tools to manage privileged access with just-in-time capabilities
- Implement role-based access control for application infrastructure
- Limit long-standing write access to production environments only to service principals
- Identify technologies and frameworks used by the application
- Make sure you understand the security features/capabilities available for each service and how they can be used in the solution
- Provide guidance for either platform managed keys (PMK) or customer managed keys (CMK), based on security or compliance requirements of this workload
- Establish a SecOps team and monitor security related events
- Enforce password-less or Multi-factor Authentication (MFA)
- Update frameworks and libraries as part of the application lifecycle
- Establish a designated point of contact to receive Azure incident notifications from Microsoft
- Implement resource locks to protect critical infrastructure
- Enforce naming conventions and resource tagging for all Azure resources
- Implement defenses that detect and prevent commodity attacks
- Use CDN to optimize delivery performance to users and obfuscate hosting platform from users/clients
- Define a process for aligning communication, investigation and hunting activities with the application team

Cost Optimization CRITICAL

50 recommended actions

Show less

Results breakdown

Critical 0-33 | Moderate 33-67 | Excellent 67-100

Your result: 9/100

50 recommended actions

- Set up a disaster recovery strategy that splits the application components and data into defined groups
- Organize data into access tiers
- Mitigate DDoS attacks
- Consider reserved instances
- Shut down VM instances not in use
- Consider VM Zone to Zone DR
- Use cost modeling to identify opportunities for cost reduction
- Configure auto-scale policies for your workload (both in and out)
- Use Azure Advisor
- Review Azure Advisor recommendations periodically
- Use developer SKUs for dev/test purposes
- Use App Service Premium (v3) plan where possible
- Consider using Service Endpoints and Private Link
- Be aware of cross-region data transfer costs
- Prefer Microsoft backbone for networking
- Be aware of cost implications of Web Application Firewall
- Consider reserved capacity for Storage
- Use data lifecycle policy
- Consider using shared disks for suitable workloads
- Consider using reserved Premium disks
- Consider utilizing disk bursting
- Consider the cost of data transfers and make sure cross-region peering is used efficiently
- Understand the cost implications of Availability Zones
- Understand cloud-native features and implement where possible
- Define a clear price model for individual services
- Define critical system flows
- Map application dependencies
- Understand the Azure services used and cost implications
- Understand the operational capabilities of Azure services
- Learn if there are any discounts available for the services already in use
- Leverage the hybrid use benefit
- Assign a budget and spend limit to the workload
- Establish a cost owner for each service used by the workload
- Use cost forecasting for budget alignment
- Consider multi-tenant or microservices scenarios when running multiple applications
- Separate data and log disks
- Utilize the PaaS pay-as-you-go consumption model where relevant
- Understand how the budget is defined
- Have ongoing conversation between app owner and business
- Develop a plan to modernize the workload
- Use RBAC to contol access to dashboards and data
- Set up alerts for cost limits and thresholds
- Collect logs and metrics from Azure resources
- Define end-date for each environment
- Consider spot VMs
- Consider B-series VMs

- Look for Public IPs and orphaned NICs
- Enforce naming conventions and resource tagging for all Azure resources
- Define a naming convention
- Pause AKS clusters

Operational Excellence CRITICAL

72 recommended actions

Show less

Results breakdown

Critical 0-33 | Moderate 33-67 | Excellent 67-100

Your result: 4/100

72 recommended actions

- Implement just-in-time privileged access management
- Store keys and secrets outside of application code in Azure Key Vault
- Implement procedures for key/secret rotation
- Use Managed Identities for authentication to other Azure platform services
- Monitor the expiry of SSL certificates
- Test your failover and failback process
- Make sure that failed tests at least temporarily block deployments
- Implement automated deployment process with rollback/roll-forward capabilities
- Deploy your workload in an active-passive configuration
- Implement release gates
- Use deployment strategies to deploy your workloads
- Define a hotfix process in case normal deployment procedures needs to be bypassed
- Use shared application and data services where appropriate
- Enable Key Vault Soft-Delete
- Use a systematic approach in your development and release process
- Reduce the need for manual operations
- Define all infrastructure components as code
- Track and address configuration drift
- Automate infrastructure deployment process
- Make sure that critical test environments have 1:1 parity with productions
- Deploy all infrastructure through an infrastructure-as-code process
- Test for performance, scalability, and resiliency
- Perform some tests in production
- Perform smoke tests
- Perform integration testing
- Make sure that all tests are automated and carried out periodically
- Perform business continuity drills
- Perform security and penetration testing regularly
- Make sure that specific methodologies are used to structure the deployment and operations process
- Implement a process between dev and ops to resolve production issues
- Make sure that operational shortcomings and failures are analyzed and used to improve and refine operational procedures
- Use tools to govern services and configurations
- Consider storing application configuration in a dedicated management system like Azure App Configuration or Azure Key Vault
- Make sure that configuration settings can be changed or modified without rebuilding or redeploying the application
- Understand the impact of changes in application health and capacity
- Configure appropriate log levels for environments
- Instrument the workload to monitor customer experience
- Monitor critical external dependencies
- Enforce resource level monitoring
- Collect Azure activity logs in your aggregation tool
- Use a log aggregation technology
- Correlate resource-level logs
- Setup black-box monitoring to monitor the platform and customer experience
- Analyze health data for your workload
- Gather logs in a structured format
- Instrument your workload
- Collect application level logs
- Understand the impact of dependencies
- Implement strategies for resiliency and self-healing
- Correlate application log events
- Compare regional capacity requirements to availability
- Implement a health model
- Create Azure Resource Health alerts
- Codify the process to provision and de-provision capacity
- Test and validate manual operation runbooks
- Document critical manual processes
- Automate recovery procedures
- Use the health model to classify failover situations
- Analyze long-term trends to predict operational issues before they occur
- Define standards, policies and best practices as code
- Enable Service Health alerts on your workload
- Send reliable alert notifications
- Prioritize operational events
- Define a process for alert reaction
- Tailor dashboards to your needs
- Implement tools to visualize application health
- Take advantage of multiple subscriptions where appropriate
- Use Azure Resource Tags to enrich resources with operational meta-data
- Use Platform as a Service offerings where appropriate
- Use feature flags
- Monitor for new features and updates that can improve your workload
- Identify if there are components with more relaxed performance requirements

Performance Efficiency CRITICAL

47 recommended actions

Show less

Results breakdown

Critical 0-33 | Moderate 33-67 | Excellent 67-100

Your result: 4/100

47 recommended actions

- Determine and document what acceptable performance is
- The health model can determine if a fault is transient
- Plan your growth, then choose regions that will support those plans
- Define a testing strategy
- Identify baseline performance targets and goals
- Aggregate application and resource logs
- Use critical system flows in the health model
- Configure retention times for logs and metrics
- Analyze long-term trends to predict performance issues
- Track how your resources scale
- Have an overall monitoring strategy for scalability
- Monitor the components required to serve a single request
- Determine the acceptable operational margin between peak utilization and maximum load
- Determine appropriate metrics for your workload
- Collect application logs from all environments with a tool like Azure Application Insights
- Capture logs in a structured format
- Correlate events across all tiers of your workload
- Develop a troubleshooting guide for database performance problems
- Develop a troubleshooting guide for high CPU or memory issues
- Determine how to isolate increased response times
- Use appropriate performance testing tools
- Use application profiling tools
- Identify human and environmental resources needed to create performance tests
- Evaluate service limits and quotas to ensure they can support future growth
- Deploy to paired regions
- Choose the right database to match usage
- Use microservices when possible
- Identify sensible non-functional requirements
- Monitor how long it takes to scale against your targets
- Choose metrics appropriately for your scaling policies
- Preemptively scale based on trends
- Know how long it takes to respond to scaling events
- Build a capacity model for your workload
- Monitor capacity utilization to forecast future growth
- Learn how to use network capturing tools
- Optimize your resource choices
- Offload SSL traffic by using the gateway offloading pattern
- Understand your performance bottlenecks around latency and throughput
- Test and validate your defined latency and throughput targets
- Consider using proximity placement groups for components that are very sensitive to network latency
- Acquire dedicated networking resources as required
- Design for eventual consistency
- Plan for the growth of your data over time
- Have a large scale event management strategy in place
- Use a Content Delivery Networks (CDN)
- Establish targets for database performance
- Implement database partitioning