Microsoft

Microsoft Azure Virtual Training Day: Well-Architected

Microsoft

Well-Architected Overview

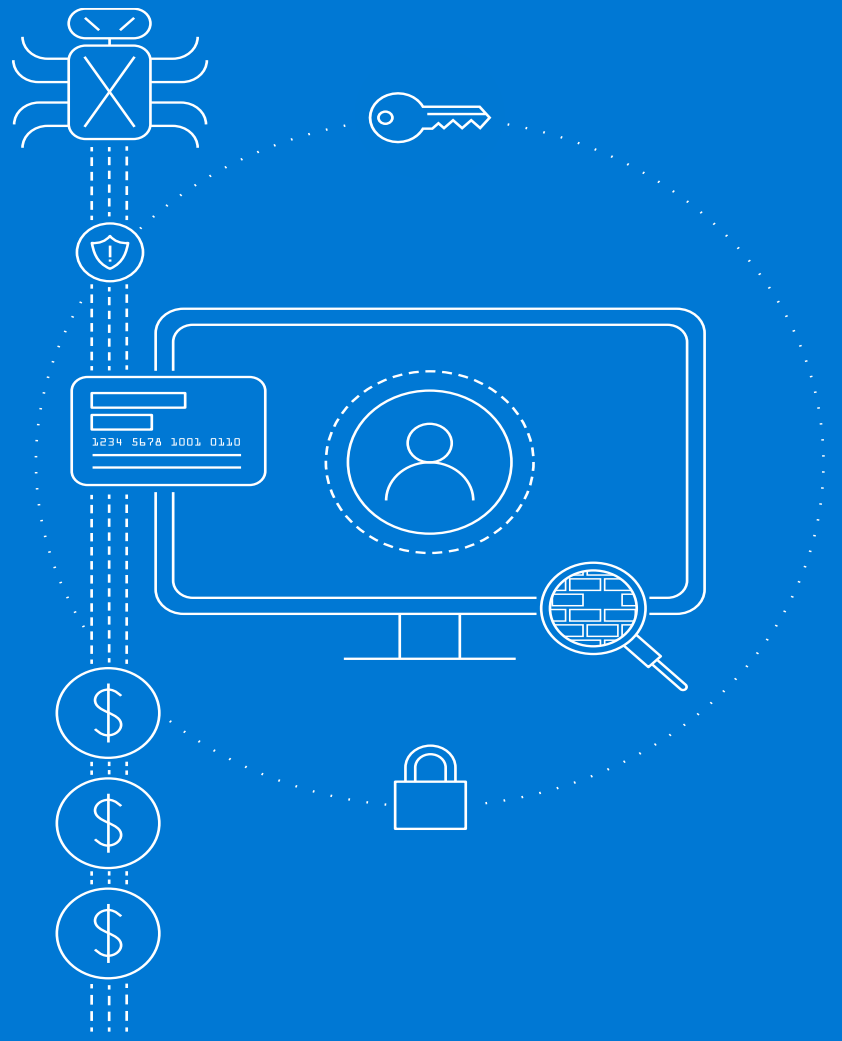
Agenda

- Why is being well-architected important?
- Overview: Microsoft Azure Well-Architected
- Overcoming workload quality inhibitors
- How to get started? – Well-Architected Review & Azure Advisor Demo
- Resources

Data breaches cost you —and your customers

Customer PII was the most frequently, and costliest compromised type of record per latest data breach study*

$3.86M	Average total cost of a data breach
80%	Number of breaches carried out with customer PII
$150	Customer PII average cost per record
$175	Increased cost per record of customer PII in breaches caused by a malicious attack
$137,000+	Remote workforce impact on average total cost of data breaches



*Cost of a Data Breach Report 2020, IBM Security, Ponemon Institute.

Run Well-Architected cloud workloads—to create value

Invest in these actions:

- Manage budget
- Improve workloads security
- Increase incident response
- Streamline internal processes
- Find costly mistakes
- Enhance workload performance

To avoid these consequences:

Expenses, losses

Trust

Damages

Well-architect—
optimize workloads for performance

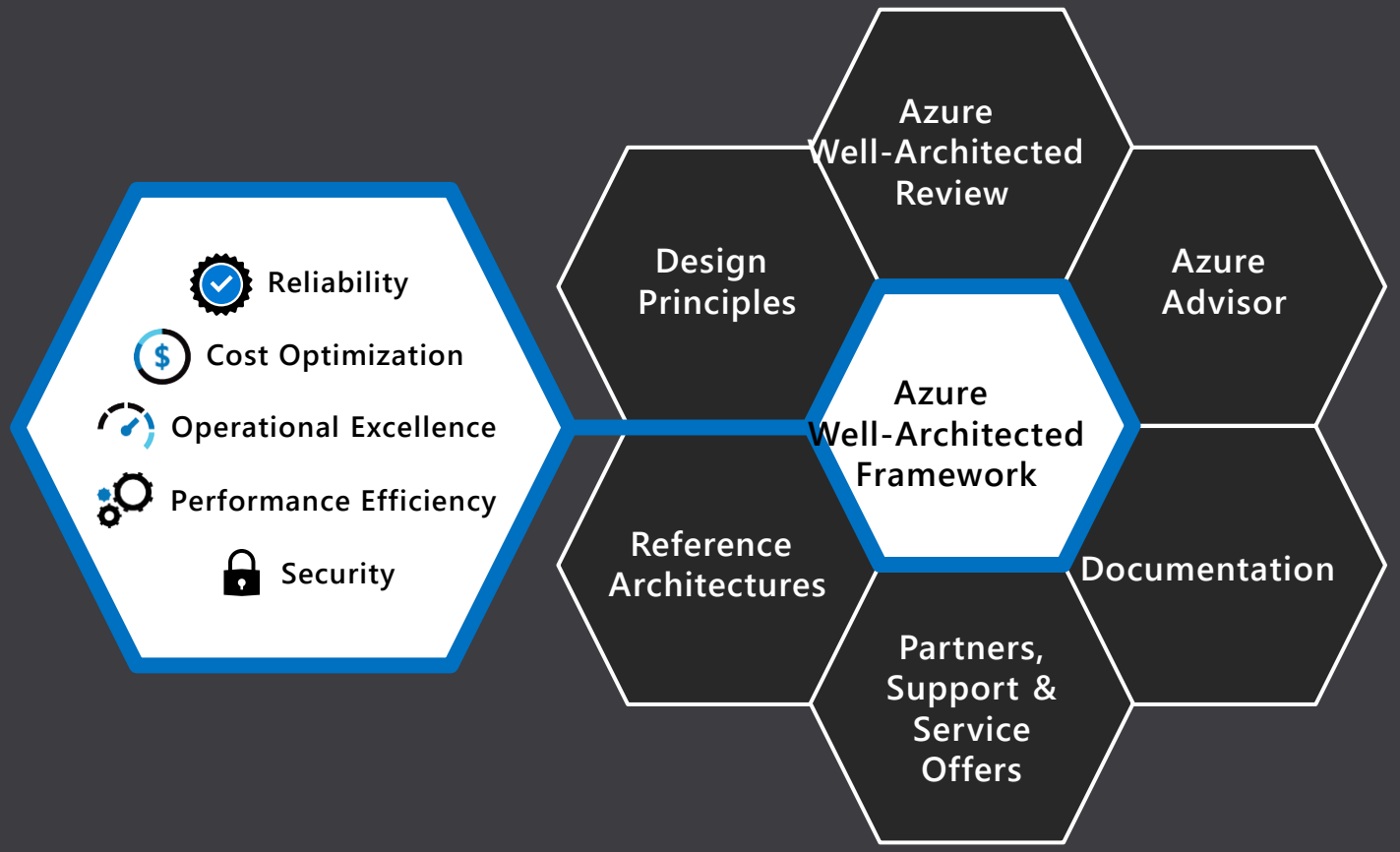
Build workloads with confidence with proven best practices

Design performant workloads using deep technical guidance

Optimize workloads with actionable focus areas

Microsoft Well-Architected—

Build and manage high-performing workloads



Building well-architected workloads— is a **shared responsibility**

Scope of **Well-Architected Assessments**

Customer application

Customer **app** or **workload**, built on the Azure platform

Platform features

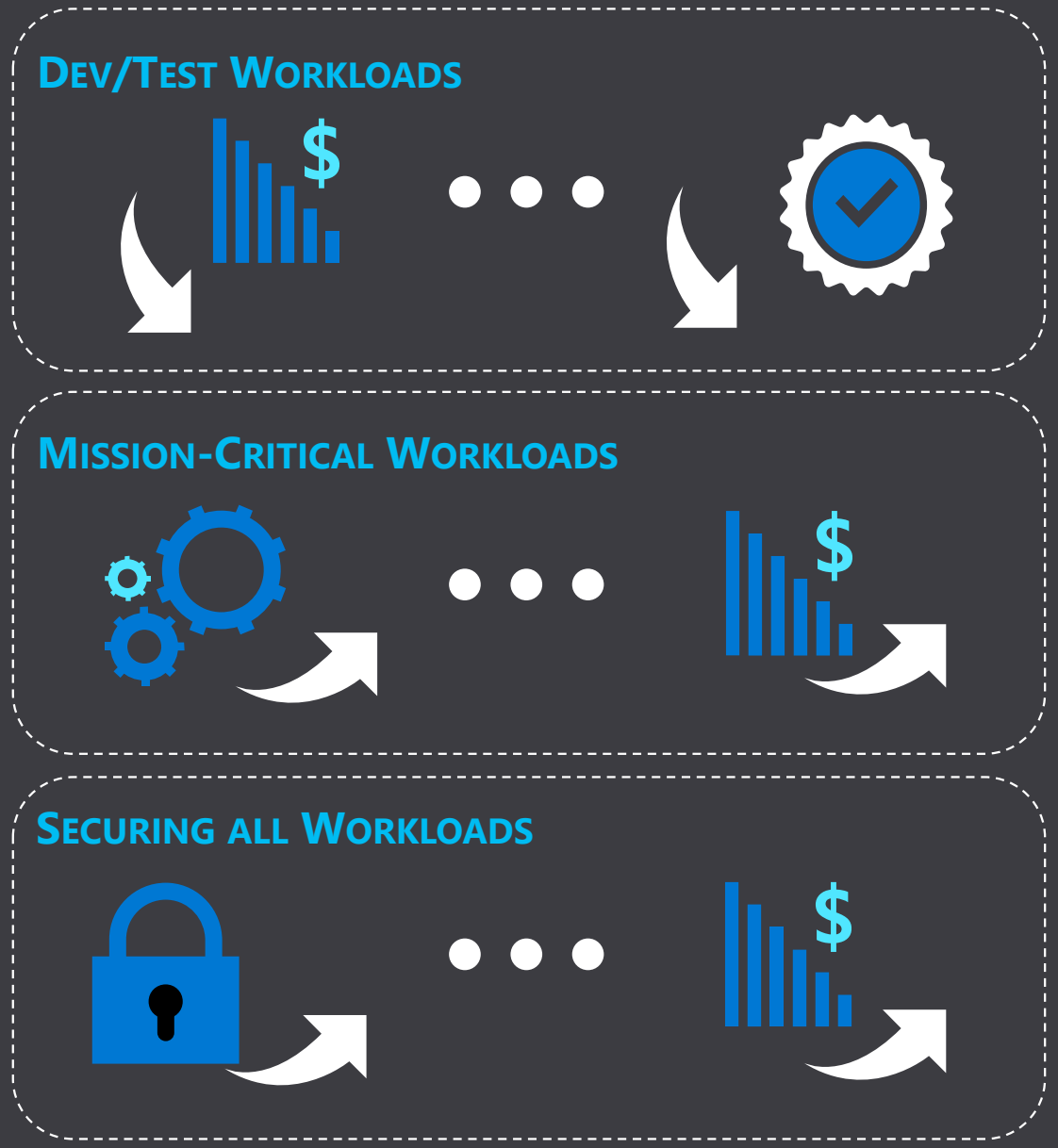
Optional Azure capabilities **a customer enables** – to ensure security, reliability, operability, performance

Platform foundation

Core capabilities **built into the Azure platform** – how the foundation is designed, operated, and monitored

Business requirements influence decisions about workload architectures

What tradeoff decisions must you make in a business context?



Overcoming workload quality inhibitors

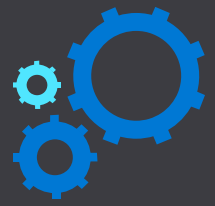
Cost Optimization

- No cost and usage monitoring
- Unclear on underused/ orphaned resources
- Lack of structured billing management
- Budget reductions from lack of support for cloud adoption by leadership

Operational Excellence

- No rapid issue identification
- No deployment automation
- No communication mechanisms & dashboards
- Unclear expectations and business outcomes
- No visibility on root cause for events

Performance Efficiency



- No monitoring new services
- No monitoring current workloads health
- No design for scaling
- Lack of rigor and guidance for technology and architecture selection

Reliability

- Unclear on resiliency capabilities for improved architecture design
- Lack of data back up practices
- No monitoring of current workload health
- No resiliency testing
- No support for disaster recovery

Security

- No access control mechanism (authentication)
- No security threat detection mechanism
- Lack of security thread response plan
- No encryption process

Best practices to drive workload quality

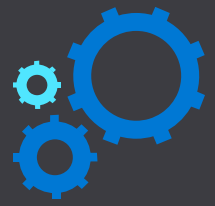
Cost Optimization

- Azure Hybrid Benefit
- Reserve Instances
- Shutdown
- Resize
- Move to PaaS

Operational Excellence

- DevOps
- Deployment
- Monitor
- Processes & cadence

Performance Efficiency



- Design for scaling
- Monitor performance

Reliability

- Define requirements
- Test with simulations & forced failovers
- Deploy consistently
- Monitor workload health
- Respond to failure & disaster

Security

- Identity & access management
- Infra protection
- App security
- Data encryption & sovereignty
- Security operations

How do you get started?

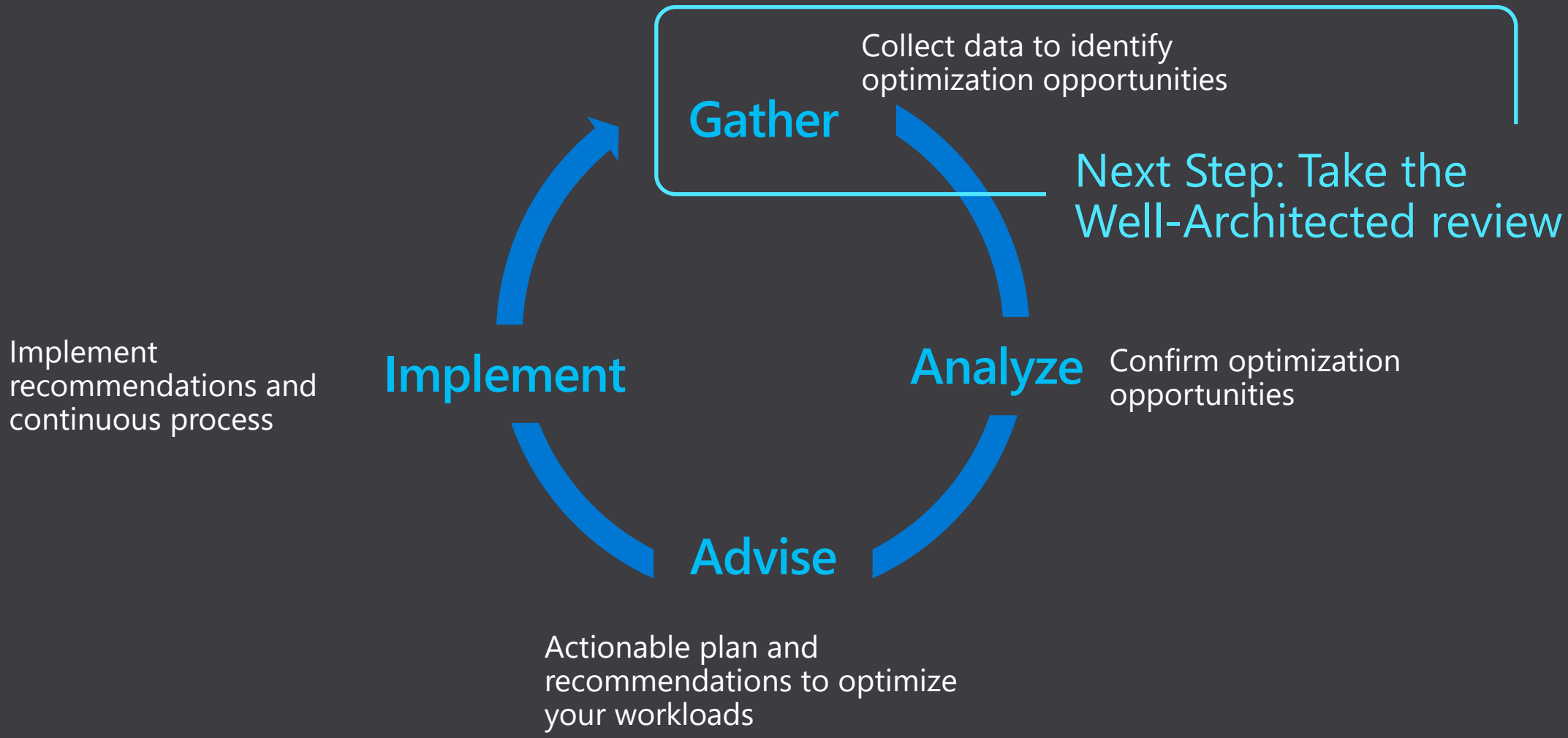
Optimize existing workloads

- Identify optimization opportunities with the Azure Advisor Score
- Understand necessary changes or past incident occurrences
- Review technical guidance of Well-Architected Framework
- Consider architecture design tradeoffs to achieve business goals
- Define & implement technical recommendations
- Implement workload optimizations on a regular cadence

Design & deploy new workloads

- Map workload architectures across business priorities
- Review technical guidance of Well-Architected Framework
- Assess workload architecture design with the Well-Architected Review
- Consider architecture design tradeoffs to achieve business goals
- Build, deploy and manage Well-Architected, optimized workloads on Azure

Optimize existing workloads - Process



Using the Azure Well-Architected Review

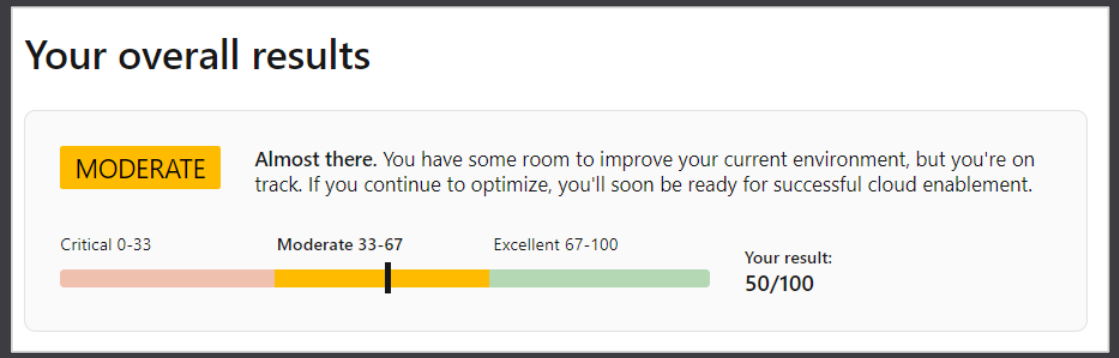
- This web-based assessment helps improve the quality of a workload by
- Examining the workload across the 5 pillars of the Azure Well Architected Framework (Reliability, Cost Optimization, Security, Operations Excellence, and Performance Efficiency)
- Providing specific guidance to improve architecture and overcome detected hurdles effectively
- Proactively focusing on the pillar where most attention is needed
- Driving consistency into workload discussions throughout the team

https://aka.ms/wellarchitected/review

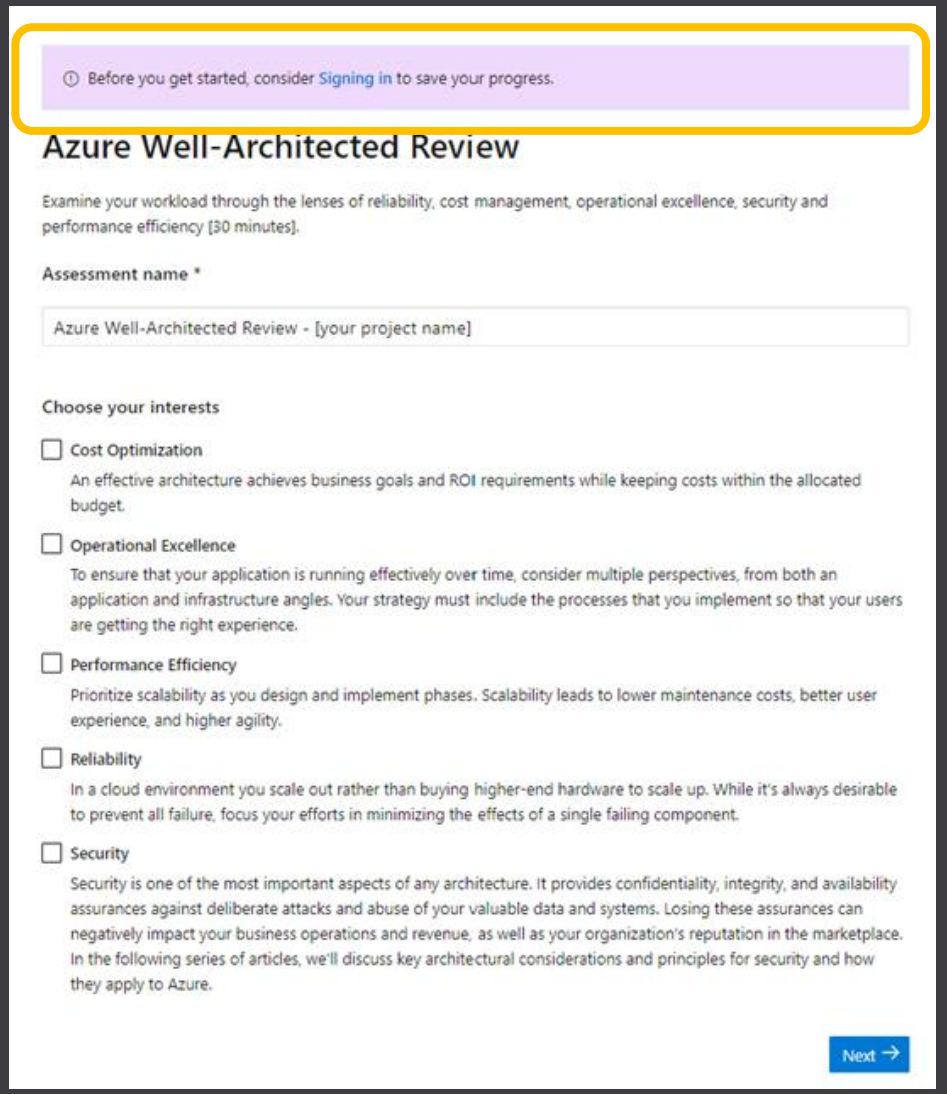
Azure Well-Architected Review

Assess workloads with the pillars of the Microsoft Azure Well-Architected Framework:

—Understand the Well-Architected level of your workload environment.



—Follow technical guidance for next steps of how to improve the quality of your workloads.



Microsoft

Demo

https://aka.ms/wellarchitected/review

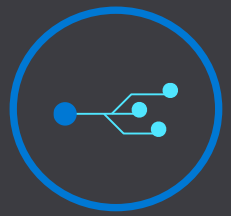
Architect and optimize workloads for success

Leverage assessment recommendations
Azure Well-Architected Review

Advance your training
Well-Architected Learning Path

Browse reference architectures
Azure Architectures

Azure Enablement Show
Channel 9 Show



Review design principles
Well-Architected Design Principles

Review the documentation
Azure Well-Architected Framework

Engage a partner
Partner Offers

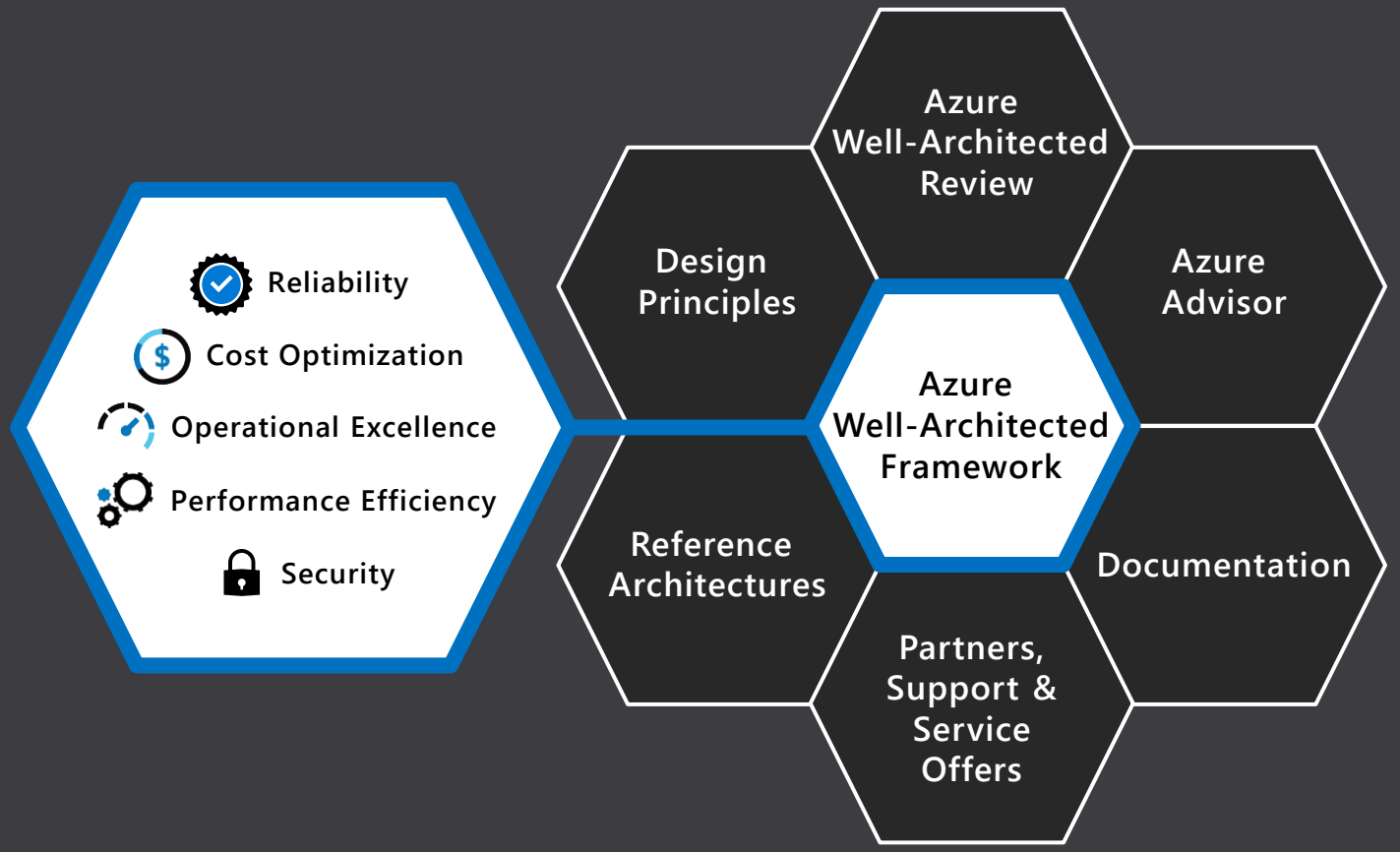
Find Service Offers
MS Consulting Services

Microsoft

Security Pillar

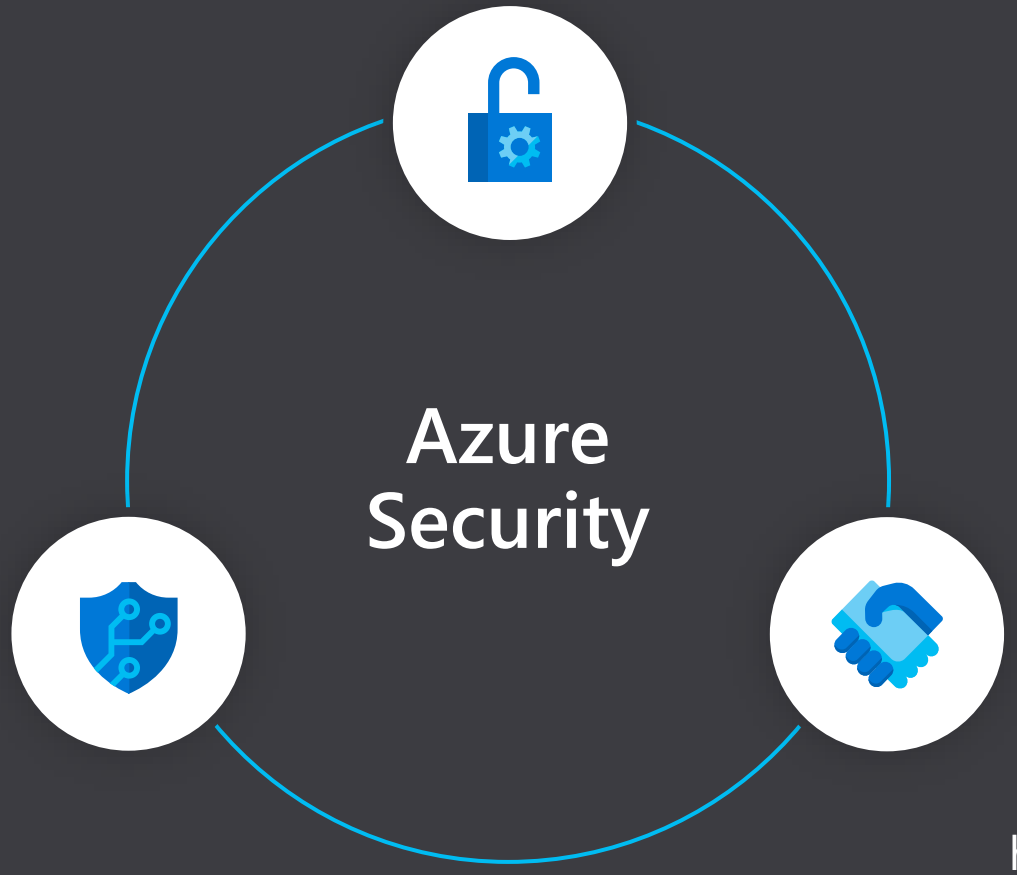
Microsoft Well-Architected—

Build and manage high-performing workloads



Operations

Security operations that work for you



Technology

Enterprise-class technology

Partnerships

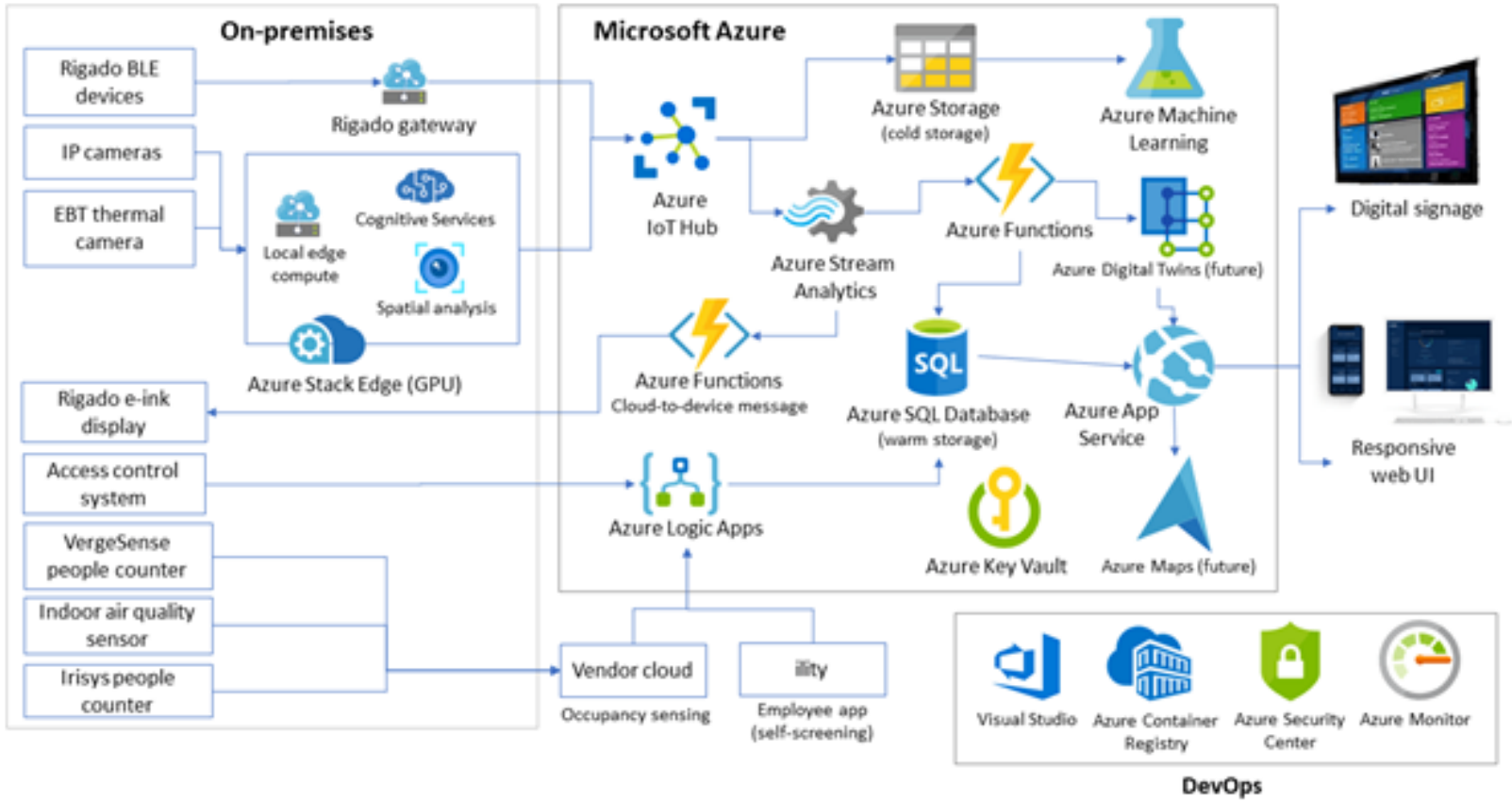
Partnerships for a heterogeneous world

RXR

RxWell

Situation

At a high level, the RxWell solution's architecture has three main zones: on-premises, the cloud, and the end-user apps. RXR and its partners committed to strict adherence to the principles of Responsible AI. All data is anonymized, and only aggregated data gets stored. Even that aggregated data is treated as sensitive, and none of it is shared externally. Meanwhile, the multilayered security controls of Azure help keep it all safe.



Solution

"When it came to developing RxWell, there was simply no other company that had the capability and the infrastructure to meet our comprehensive data, analytics, and security needs than Microsoft. With our partnership, the RxWell program provides our customers the tools they need to safely navigate the 'new abnormal' of COVID-19 and beyond."

—Scott Rechler, Chairman and CEO, RXR Realty

Impact

"The well-architected Azure framework, fundamentally addresses things like scalability, reliability, security, and operational excellence. Because we started building our solution from day one on those pillars, that helped us to absorb all these nuances from the integration perspective."

—Saurav K. Chandra, Principal Architect for Internet of Things, Infosys

Build and manage proactively secured workloads

Security provides principles to protect, detect, and respond to threats across your Azure environment.

Build upon a secure foundation

- Design assuming workload failure with multi-layer protection controls.
- Build workloads using zero-trust principles in both IaaS and PaaS
- Embrace Azure's security investments, resources, and compliance certifications

Proactively stay secure with native controls

- Continuously manage your workload security from a single pane of glass with Azure Security Center.
- Protect your workloads from malicious attacks with cloud-native Azure Web Application Firewall
- Manage identity and access for your workload with Azure Active Directory

Detect and respond to threats

- Leverage large-scale intelligence from decades of Microsoft security experience to work with the Microsoft Intelligent Security Graph, collected from 8 trillion threat signals analyzed daily
- Embrace automation with Azure Defender to get threat protection for your workload
- Establish procedures to identify and mitigate threats for your workloads with Azure Sentinel

Build on a secure foundation

Principle: Build a comprehensive strategy

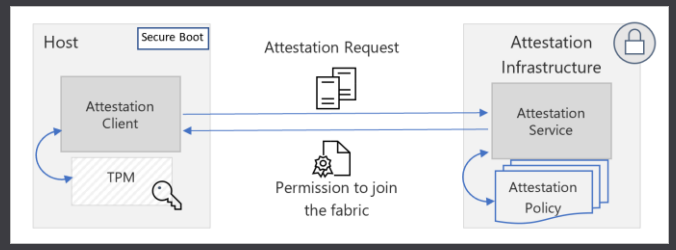
A security strategy should consider investments in culture, processes, and security controls across all system components. The strategy should also consider security for the full lifecycle of system components including the supply chain of software, hardware, and services.

Protect customer data

- Use Azure Active Directory to manage access to Azure resources.
- Use Azure Key Vault to store sensitive data such as certificates, connection strings, and tokens.
- The Azure Security Benchmark provides recommendations to improve the security of your workloads, data, and services.

Secure hardware

- Azure is hosted on custom-built hardware with integrated security.
- Host Attestation Service ensures that host machines are trust-worthy before they're allowed to interact with customer data.



high-level architecture of the host attestation service

Test and monitor

- Run simulated penetration attacks to detect system vulnerabilities and validate defenses.
- Classify, protect, and monitor sensitive data assets using access control, encryption, and logging.

Build upon a secure foundation

Principle: Assume Zero Trust

DDoS protection

DDOS protection tuned to your application traffic patterns

Web Application Firewall

Centralized inbound web application protection from common exploits and vulnerabilities

Azure Firewall

Data exfiltration protection using centralized outbound and inbound (non-HTTP/S) network and application (L3-L7) filtering

Network Security Groups

Distributed inbound & outbound network (L3-L4) traffic filtering on VM, Container or subnet

VNET Integration

Restrict access to Azure service resources (PaaS) to only your Virtual Network

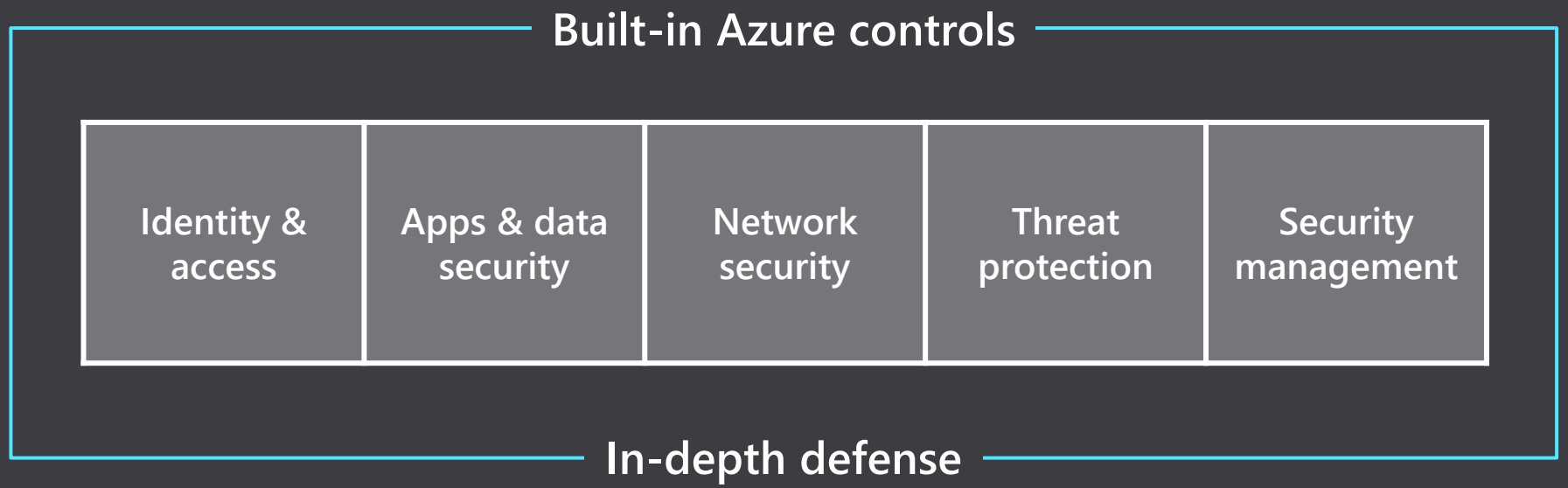
Application protection

Segmentation

Proactively stay secure with native controls

Principle: Leverage native controls

Native security controls are maintained and supported by the service provider, eliminating or reducing effort required to integrate external security tooling and update those integrations over time.



Proactively stay secure with native controls

Principle: Leverage native controls

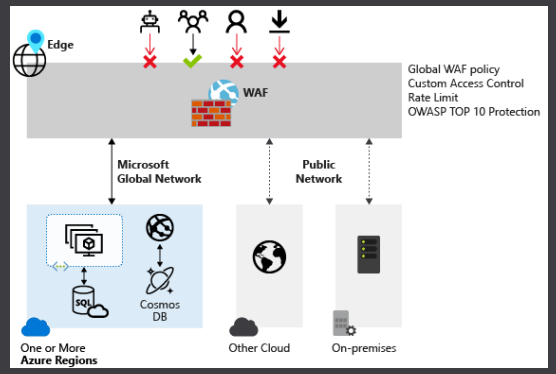
Azure Security Center

A unified infrastructure security management system that:

- Strengthens the security posture of your data center
- Provides advanced threat protection across your hybrid workloads in the cloud—on Azure, or on premises

Web Application Firewall

- Centralized protection and inspection of HTTP requests to prevent attacks such as SQL Injection or Cross-Site Scripting.



Azure Active Directory

- Microsoft's cloud-based identity and access management service, which helps your employees securely access resources.
- Managed Identities eliminates the need to store credentials that could be leaked.
- Use Azure AD Connect for synchronizing Azure AD with your existing on-premises directory.

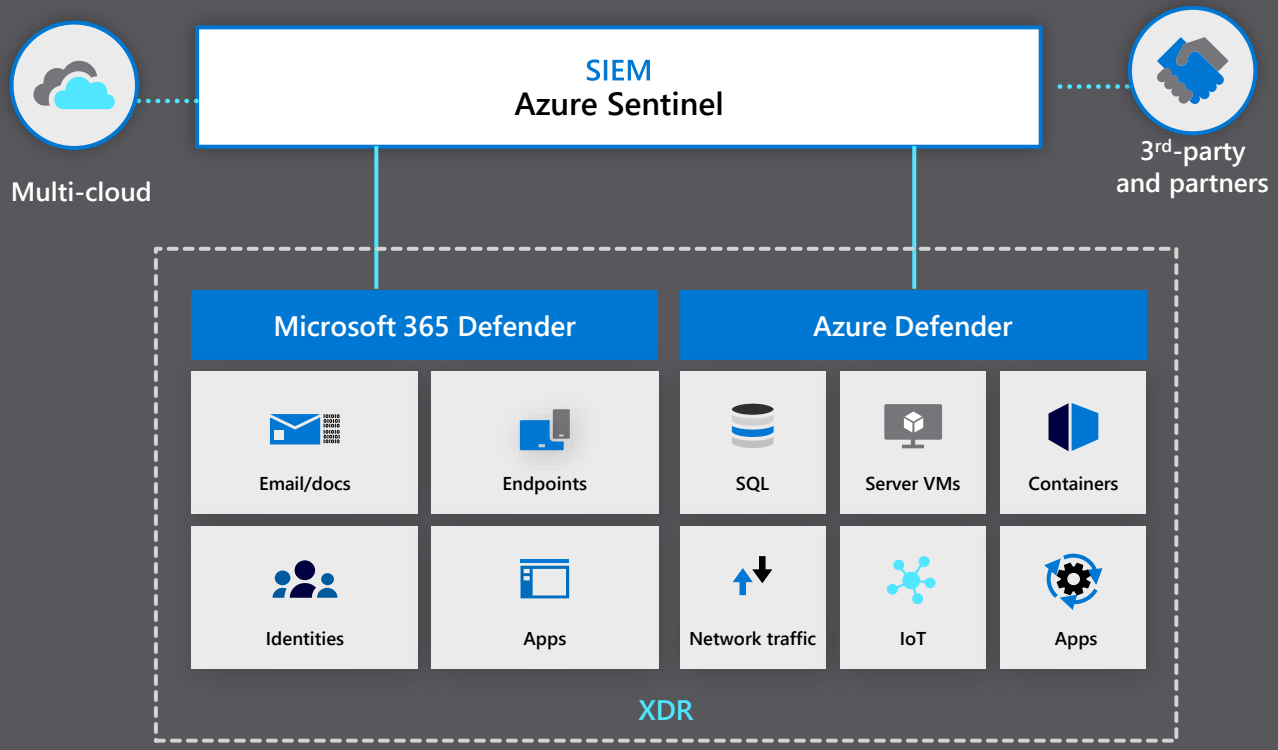
Detect and respond to threats

Principle: Design for resilience

Native security and governance

- ASC/Secure Score
- Firewall
- Web App Firewall
- SQL Protection
- API Protection
- ...

Native threat detection



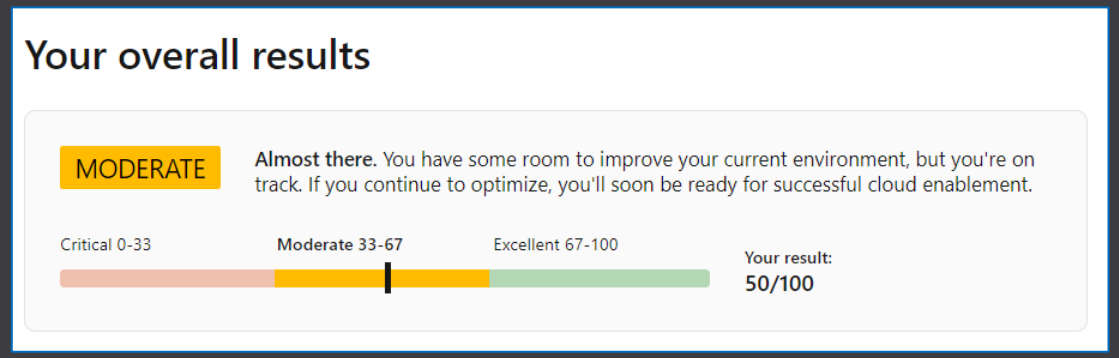
Security checklist

- ❑ Use Identity as Primary Access Control
- ❑ Assign permissions to users, groups, and applications at a certain scope through Azure RBAC. Use built-in roles when possible.
- ❑ Restrict control plane access based on a need-to-know basis and least privilege security principles.
- ❑ Enforce multi-factor verification for users
- ❑ Protect all public endpoints with Azure Front Door, Application Gateway, Azure Firewall, Azure DDoS Protection
- ❑ Prevent direct internet access of virtual machines

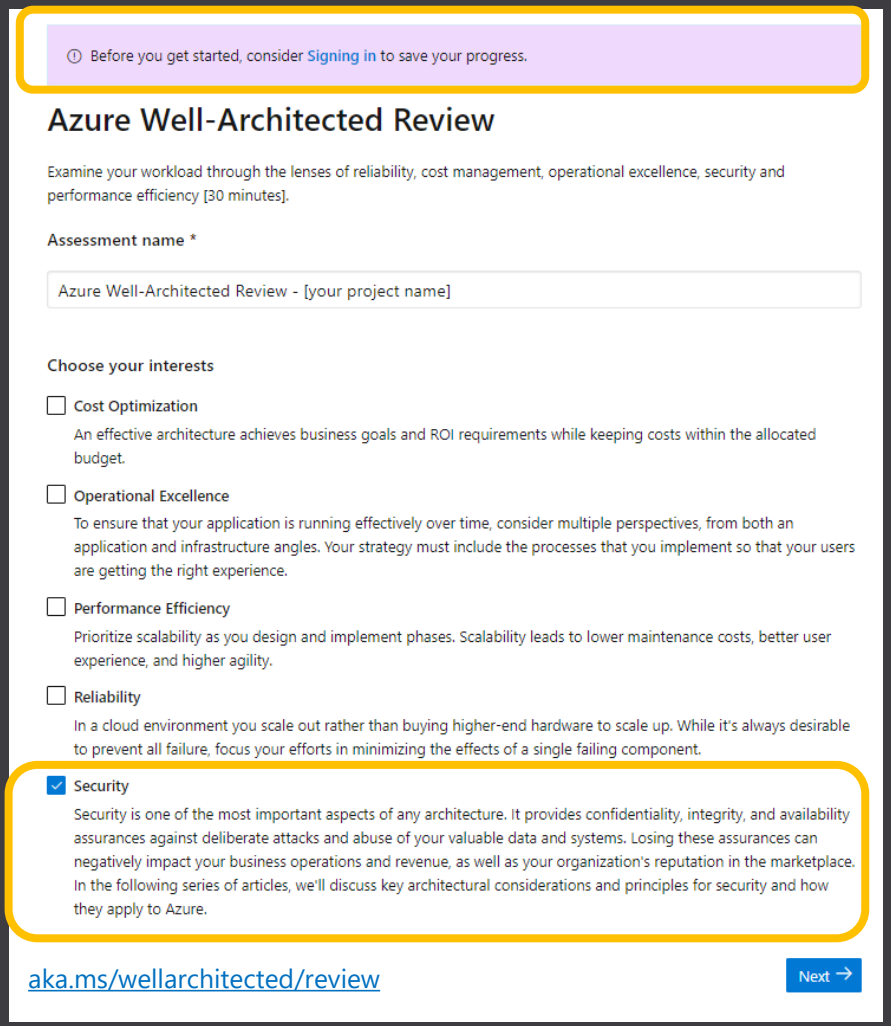
Azure Well-Architected Review

Assess workloads with the pillars of the Microsoft Azure Well-Architected Framework:

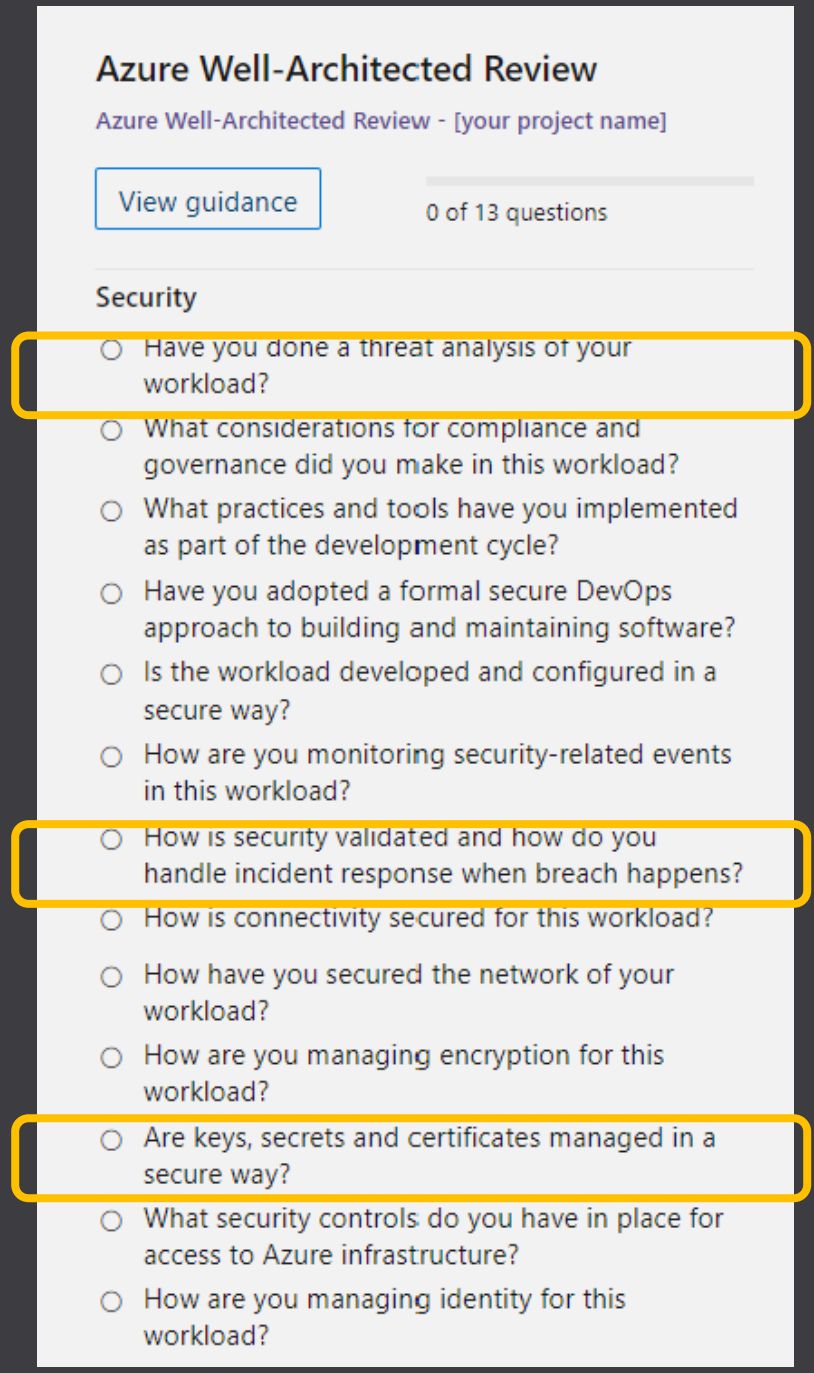
—Understand the Well-Architected level of your workload environment.



—Follow technical guidance for next steps of how to create and optimize your workloads.



Let's walk through some questions for Security in the Well-Architected Review



Have you done a threat analysis of your workload?

Threat modeling is an engineering technique which can be used to help identify threats, attacks, vulnerabilities and countermeasures that could affect an application. Threat analysis consists of defining security requirements, identifying threats, mitigating threats, validating threat mitigation. All of those are needed to ensure proper security of a workload on both the prevention and reaction fronts.

- ❑ Threat modeling processes are adopted, identified threats are ranked based on organizational impact, mapped to mitigations and communicated to stakeholders.
- ❑ Timelines and processes are established to deploy mitigations (security fixes) for identified threats.
- ❑ Security requirements are defined for this workload.
- ❑ Threat protection was addressed for this workload.
- ❑ Security posture was evaluated with standard benchmarks (CIS Control Framework, MITRE framework etc.).
- ❑ Business critical workloads, which may adversely affect operations if they are compromised or become unavailable, were identified and classified.
- ❑ None of the above.

How is security validated and how do you handle incident response when breach happens?

If prevention fails and security of the application is breached, proper response and mitigation can minimize damage and contain the attacker within minimal boundaries.

- ❑ For containerized workloads, Azure Defender (Azure Security Center) or other third-party solution is used to scan for vulnerabilities.
- ❑ Penetration testing is performed in-house, or a third-party entity performs penetration testing of this workload to validate the current security defenses.
- ❑ Simulated attacks on users of this workload, such as phishing campaigns, are carried out regularly.
- ❑ Operational processes for incident response are defined and tested for this workload.
- ❑ Playbooks are built to help incident responders quickly understand the workload and components, to mitigate an attack and do an investigation.
- ❑ There's a security operations center (SOC) that leverages a modern security approach.
- ❑ A security training program is developed and maintained to ensure security staff of this workload are well-informed and equipped with the appropriate skills.
- ❑ None of the above.

Are keys, secrets, and certificates managed in a secure way?

Secrets like API keys and certificates are sensitive pieces of information that need to be managed in a secure way - that includes proper storage, encryption and access control.

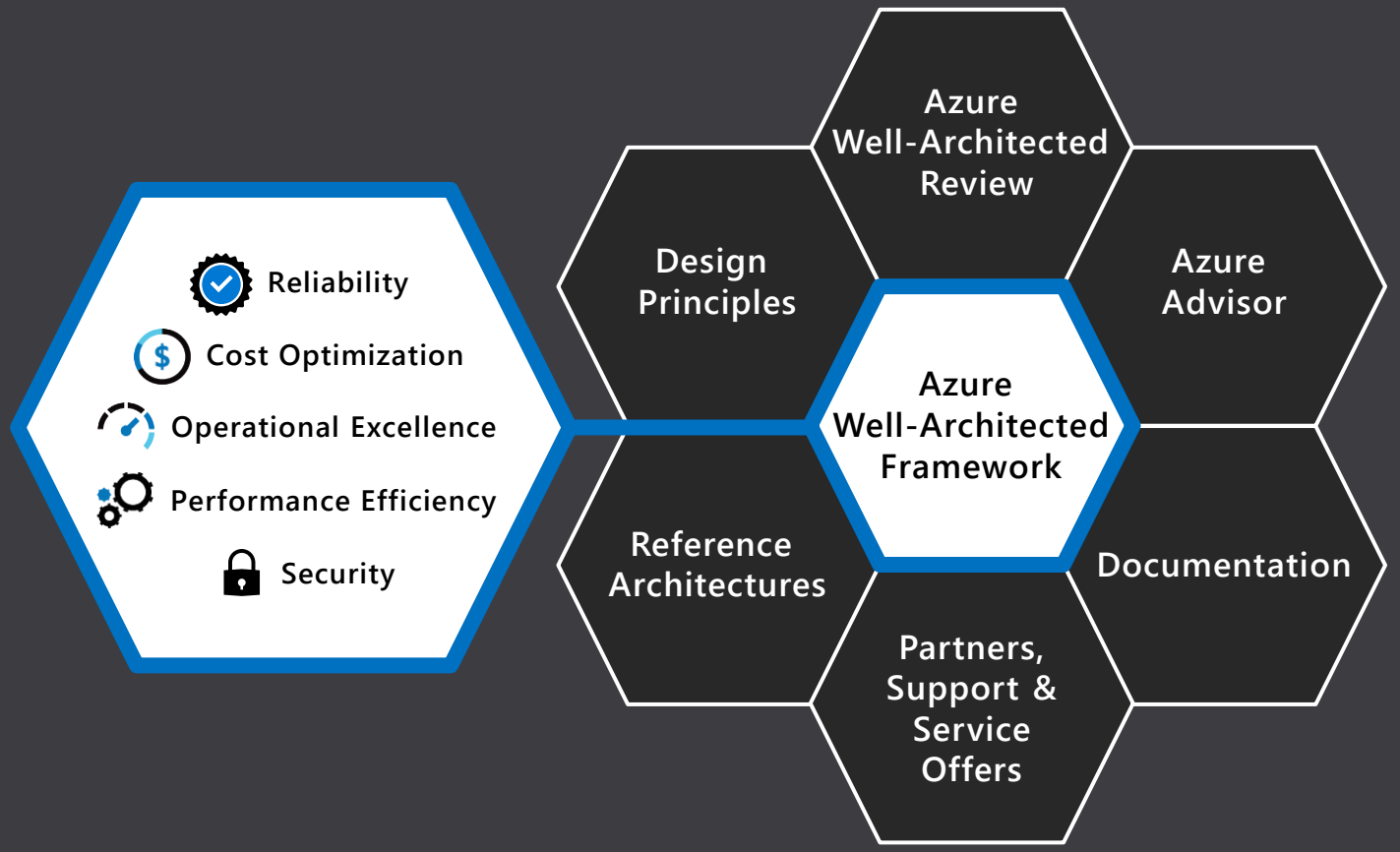
- ❑ There's a clear guidance or requirement on what type of keys (PMK - Platform Managed Keys vs. CMK - Customer Managed Keys) should be used for this workload.
- ❑ Passwords and secrets are managed outside of application artifacts, using tools like Azure Key Vault.
- ❑ Access model for keys and secrets is defined for this workload.
- ❑ A clear responsibility / role concept for managing keys and secrets is defined for this workload.
- ❑ Secret/key rotation procedures are in place.
- ❑ Expiry dates of SSL/TLS certificates are monitored and there are renewal processes in place.
- ❑ None of the above.

Microsoft

Performance Efficiency Pillar

Microsoft Well-Architected—

Build and manage high-performing workloads



Build and manage—scalable, efficient workloads

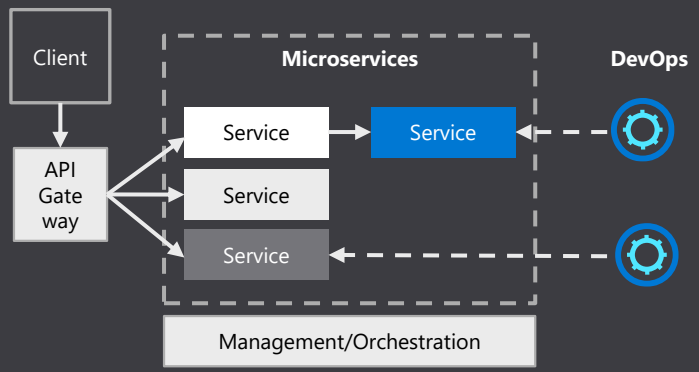
Design and manage workloads that scale according to load changes, and efficient systems, processes, and resources

Tools to provide scalability

- Manage resource scaling with Azure SQL Database and Azure App Services—or scale dynamically with demand with Azure Autoscale
- Optimize your network and storage with Azure Cosmos DB, Azure Traffic Manager, and Azure Cache for Redis

Efficient architecture tradeoffs

- Design parts of the process to be discrete and decomposable to maximize compute resources, and take microservices architecture into account



Active response to performance issues

- Evaluate health levels of workloads with Azure Monitor and Log Analytics to provision resources dynamically, and scale to match demand
- Assess and remediate deep application performance issues and trends with Azure Application Insights
- Embrace a data-driven culture to deliver timely insights across data to your entire organization

Optimal service execution

Principle: Invest in capacity planning

Test continuously

- Establish baselines for your application and its supporting infrastructure
- Always test the effect on performance when code or infrastructure changes are made
- Monitor typical and peak system loads to provide visibility on operational peaks outside designed limits

Anticipate load fluctuations

- Test for expected loads because of planned events, for example, sales promotions, or holidays
- Plan for unexpected political, economic, and weather events
- Choose paired regions, and ensure that all regions can adequately scale to maximize uptime

Carefully evaluate services and costs

- Review service-level agreements (SLAs) of similar services to calculate the best fit for your application
- Consider the effects of business requirements when making trade-offs between cost and performance
- Use cost calculators to estimate initial and operational costs

Efficient architecture tradeoffs

Principle: Run performance testing in the scope of development

Distributed systems require more effort

- Evaluate the systemic effect of each application—its supporting services, and the latency between application layers
- Ensure that all services can scale to support loads, and that one service will not be a bottleneck
- Services may need to scale differently under loads

Test & tune performance

- Establish an SLA that defines performance targets for latency, number of requests, and exception rate for each workload
- Use proven best practices such as properly instrumenting code, monitoring multiple load percentages, and systemic troubleshooting

Avoid performance antipatterns

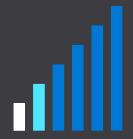
- Performance antipatterns are common, defective processes and implementations within organizations—likely to cause scalability problems when an application is under pressure
- Antipatterns may be obvious, for example, the inability to scale from on-premises to the cloud

Active response to performance issues

Principle: Continuously monitor the application and supporting infrastructure

Azure Monitor

- Comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments.
- Helps to maximize the availability and performance of applications and services



Log analytics

- Edit and run log queries from data collected by Azure Monitor Logs, and interactively analyze the results
- Retrieve records matching precise criteria, identify trends, analyze patterns, and provide a variety of data insights

Application insights

- Provides visibility into app performance and utilization patterns
- Monitors various data sources, including request, response, and failure rates, exceptions, page views, and load performance

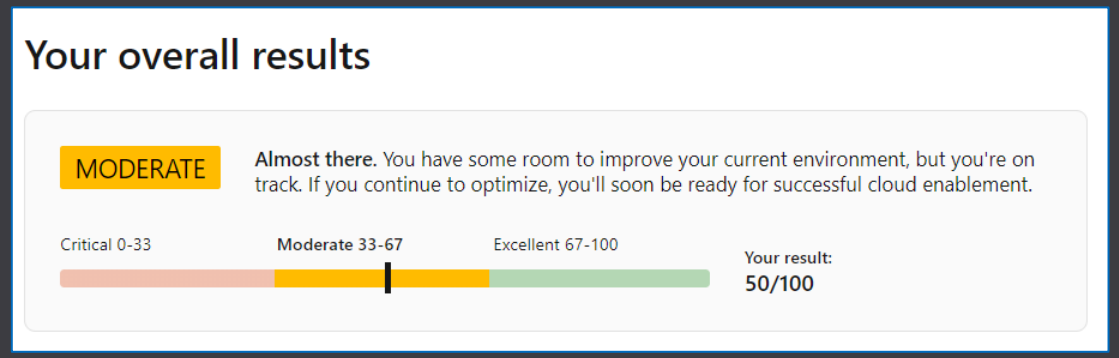
Scaling design checklist

- ❑ Stateless operations
- ❑ Autodetection and load balancing
  - ❑ Reactive scaling
  - ❑ Scheduled scaling
- ❑ Scale as a unit
- ❑ Partitioning
  - ❑ Workload
  - ❑ Data
- ❑ Offload CPU and I/O intensive
- ❑ Consider moving towards *shared-nothing* architecture
- ❑ Eventual consistency where possible
- ❑ Aggressive caching

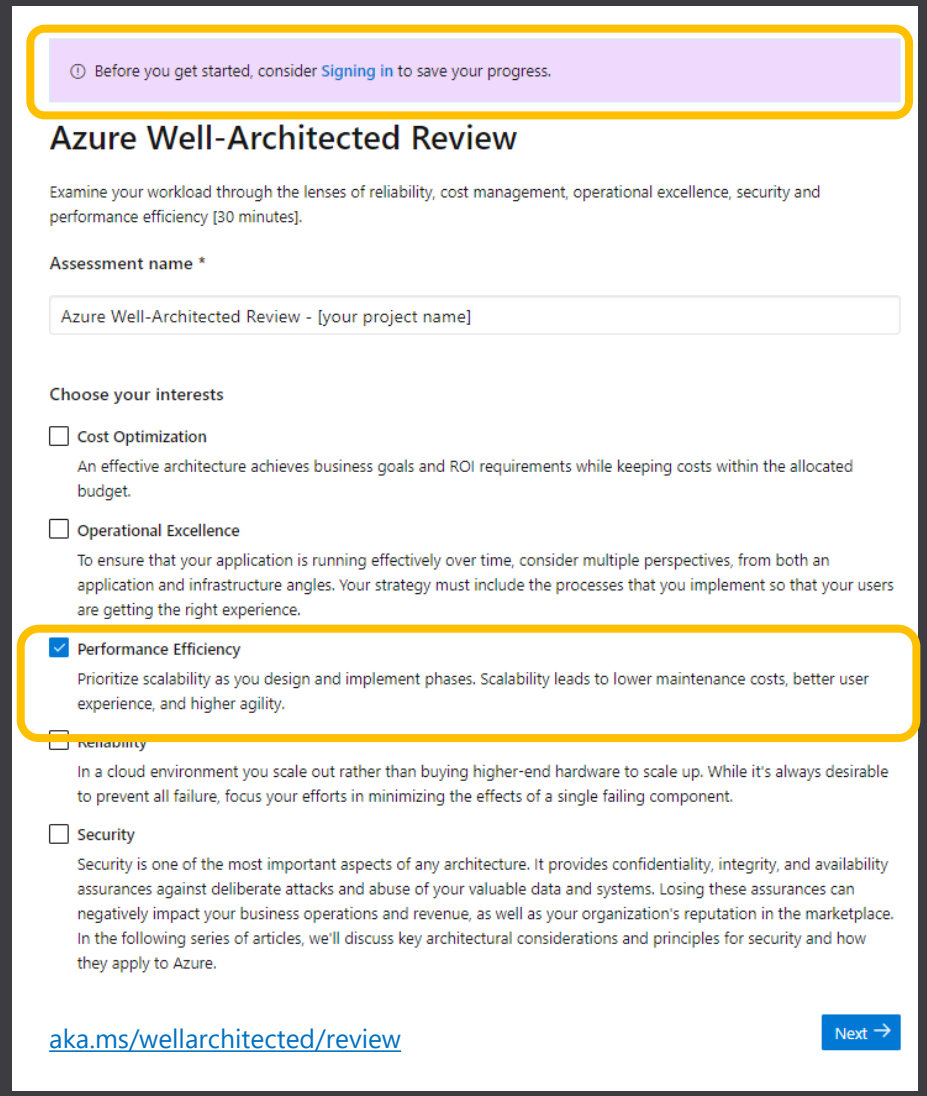
Azure Well-Architected Review

Assess workloads with the pillars of the Microsoft Azure Well-Architected Framework:

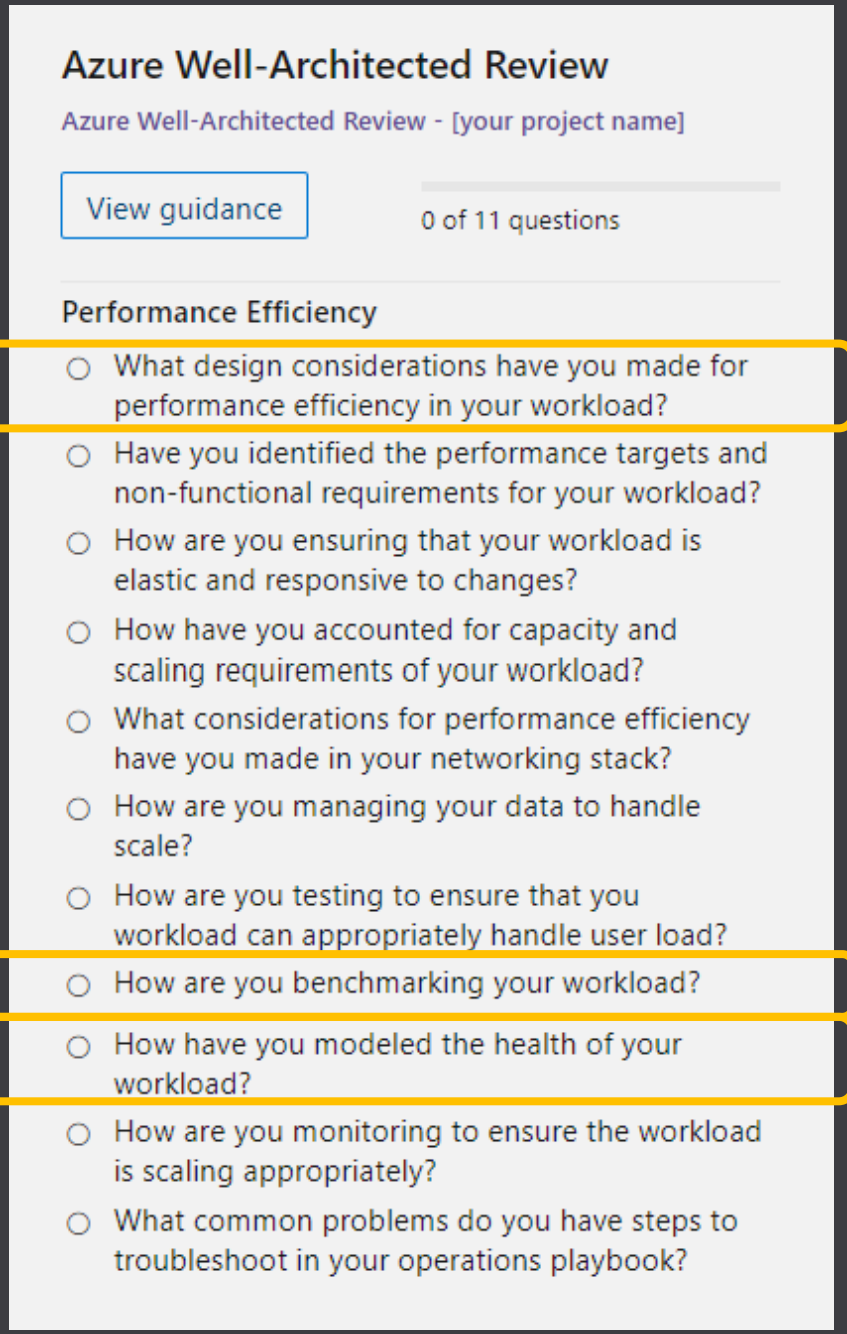
—Understand the Well-Architected level of your workload environment.



—Follow technical guidance for next steps of how to create and optimize your workloads.



Let's walkthrough some questions for Performance Efficiency in the Well-Architected Review



What design considerations have you made for performance efficiency in your workload?

As traffic fluctuates into your application the number of underlying resources that you need will vary over time.

- ❑ The workload is deployed across multiple regions.
- ❑ Regions were chosen based on location, proximity to users, and resource type availability.
- ❑ Paired regions are used appropriately.
- ❑ You have ensured that both (all) regions in use have the same performance and scale SKUs that are currently leveraged in the primary region.
- ❑ Within a region the application architecture is designed to use Availability Zones.
- ❑ The application is implemented with strategies for resiliency and self-healing.
- ❑ Component proximity is considered for application performance reasons.
- ❑ The application can operation with reduced functionality or degraded performance in the case of an outage.
- ❑ You choose appropriate datastores for the workload during the application design.
- ❑ Your application is using a micro-service architecture.
- ❑ You understand where state will be stored for the workload.
- ❑ None of the above.

How have you modeled the health of your workload?

- ❑ Application and resource level logs are aggregated in a single data sink or able to be cross-queried.
- ❑ A health model is used to qualify what 'healthy' and 'unhealthy' states represent for the application.
- ❑ Critical system flows are used to inform the health model.
- ❑ The health model can distinguish between transient and non-transient faults.

- ❑ The health model can determine if the workload is performing at the expected targets.
- ❑ Retention times for logs and metrics been defined and housekeeping mechanisms are configured.
- ❑ Long-term trends are analyzed to predict performance issues before they occur.
- ❑ None of the above.

How are you benchmarking your workload?

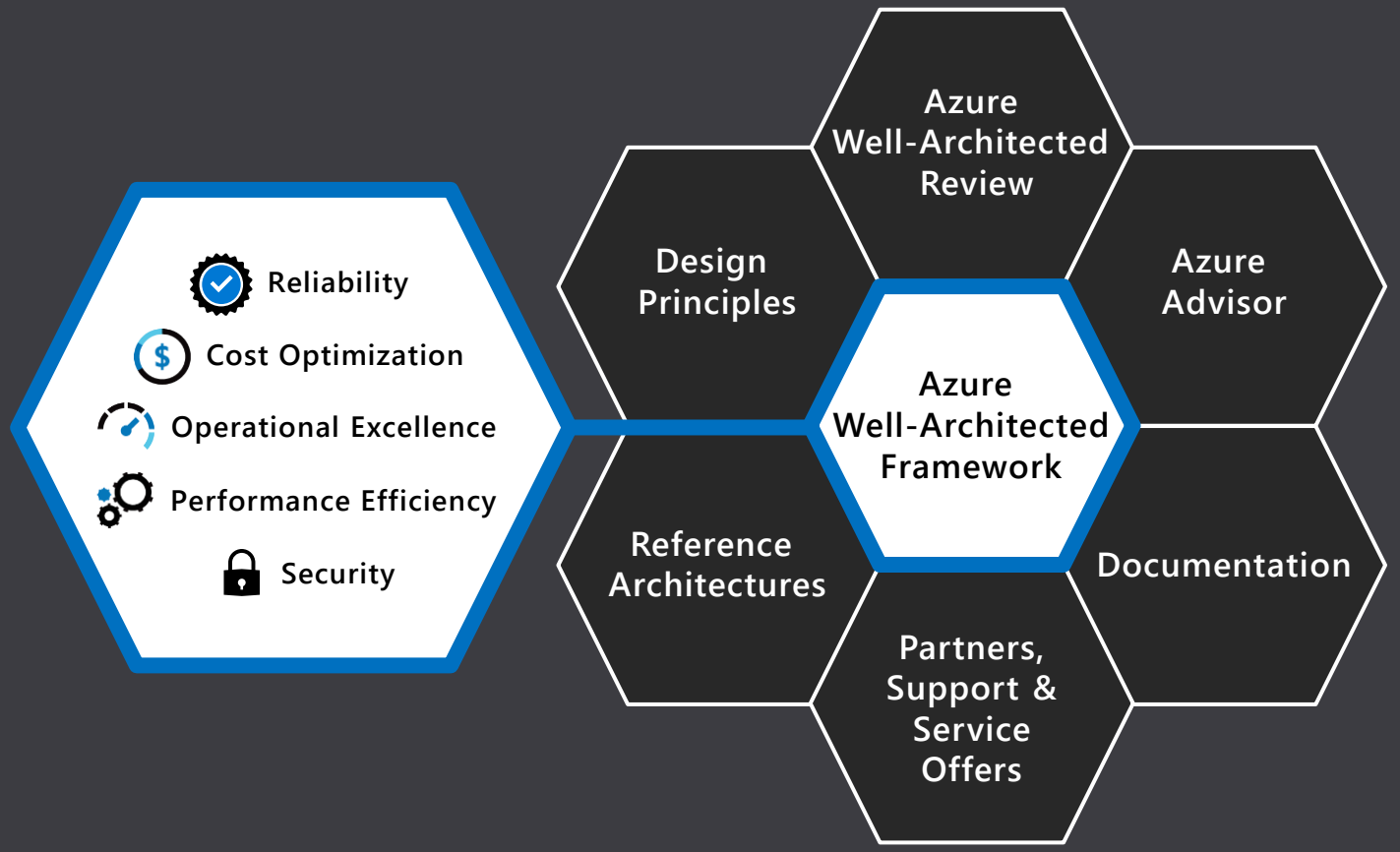
- ❑ You have identified goals or a baseline for workload performance.
- ❑ Performance goals are based on device and/or connectivity type as appropriate.
- ❑ You have defined an initial connection goal for your workload.
- ❑ There is a goal defined for complete page load times.
- ❑ You have defined goals for an API (service) endpoint complete response.
- ❑ There are goals defined for server response time.
- ❑ You have goals for latency between the systems & microservices of your workload.
- ❑ There are goals on database query efficiency.
- ❑ You have a methodology to determine what acceptable performance is.
- ❑ None of the above.

Microsoft

Operational Excellence Pillar

Microsoft Well-Architected—

Build and manage high-performing workloads



Build, deploy, and manage workloads—with trustworthy processes

Agile and accurate processes

- Apply DevOps to break down silos between development and operations across your organization
- Build and test workloads with Continuous Integration and Continuous Delivery (CI/CD) both in development and production stages
- Perform extensive automated testing with Azure Pipelines or manual testing with Azure Testing Plans

Focused and assertive application monitoring

- Dive deep into your workload's information with Log Analytics for infrastructure and with Azure Application Insights for application trends
- Manage the health of your system and activity logging by consuming core monitoring insights provided by Azure Monitor

Continuous improvement

- Enjoy the flexibility of creating agile and independent workloads with microservices and loosely coupled architectures
- Use Chaos Engineering practices and run regular tests to reach higher levels of maturity and operational effectiveness.
- Reduce process risks by automating operational tasks and deployments with Azure Automation, Azure CLI and Azure PowerShell

Agile and accurate processes

Principle: Optimize build and release processes

Infrastructure as Code

- Define the entire Infrastructure as Code just as you define your application
- Increase accuracy and reduce process risks preventing configuration drift
- Enable easy recreation of new environments, e.g., for developing new features

Continuous Integration & Continuous Delivery

- Build and test workloads with Continuous Integration and Continuous Delivery (CI/CD) both in development and production stages, to achieve a single and consistent way of building and deploying.
- Eliminate error-prone manual interventions
- Versioning of CI/CD pipelines for traceability of changes

Automated testing

- Perform extensive automated testing to ensure a stable code base and resource composition before deploying to critical systems
- Achieve a faster time-to-ship with fewer errors

Focused and assertive application monitoring

Principle: Monitor system and understand operational health

Monitor build and release processes

- Give developers early feedback on pushed code changes
- Avoid outages caused by the rollout of new features

Monitor infrastructure and application health

- Build confidence in the overall health of your workload
- Dive deep into instrumentation with Log Analytics for infrastructure monitoring
- Instrument your code to collect all relevant events and metrics
- Use comprehensive dashboards that are tailored to your audiences
- Leverage Azure Application Insights for observing application trends

Understand workload health to meet business goals

- Understand the business impact of reduced workload health
- Correlate events and metrics across different parts of your solution
- Respond to issues with self-healing capabilities

Continuous improvement

Principle: Use loosely coupled architecture

Strive for a true DevOps model

- Apply DevOps to break down silos between development and operations across your organization..
- Run agile and independent teams that are in charge of developing and running their parts of the workload
- Limit impact of issues by having clear boundaries between services

Microservices design

- Enjoy the flexibility of creating agile and independent workloads with microservices.

Continuous improvement

Principle: Rehearse recovery and practice failure

Rehearse recovery

- Only tested recovery procedures will work in times of emergency
- Validate operation runbooks
- Run regular tests and conduct dry runs of failover scenarios

Practice failure

- Test your workload with injected faults in a safe environment
- Use Chaos Engineering practices to reach higher levels of maturity
- Employ a Red Team to find issues and weak points

Continuous improvement

Principle: Embrace operational improvement

Evolve processes

- Establish well-defined owners and playbooks for procedures and tasks to optimize operational effectiveness.
- Establish regular cadences for testing operational procedures and tasks.
- Review operational incidents to improve operational effectiveness.
- Establish Root Cause Analysis processes.

Optimizing inefficiencies through automation

- Save time, reduce risks and avoid errors by automating operational tasks or any deployments that may occur on a schedule, response to events/monitoring alert, or ad-hoc based on external factors.
- Automate deployments with Infrastructure as Code to define the infrastructure that needs to be deployed.
- Optimize workload configurations by automating software installs, adding data to a database, updating networking and other actions.

Automation checklist

Reduce toil, improve efficiency, and ensure consistency

Infrastructure deployment checklist

Infrastructure configuration checklist

Operational task checklist

Infrastructure deployment checklist

- ❑ Choose a declarative over imperative approach
  - ❑ Azure Resources Manager (ARM) templates
  - ❑ Terraform
  - ❑ Azure Control Plane (via Azure REST API)
- ❑ Establish a repeatable infrastructure
- ❑ Avoid configuration drift through regular deployments
- ❑ Dynamically provision multiple environments
- ❑ Utilize as part of disaster recovery plan

Infrastructure configuration checklist

- ❑ Utilize the Azure data plane
- ❑ Bootstrap automation
  - ❑ Azure VM Extensions
  - ❑ cloud-init
  - ❑ Deployment scripts
- ❑ Configuration management
  - ❑ Azure Automation State Configuration
  - ❑ Chef
  - ❑ Puppet

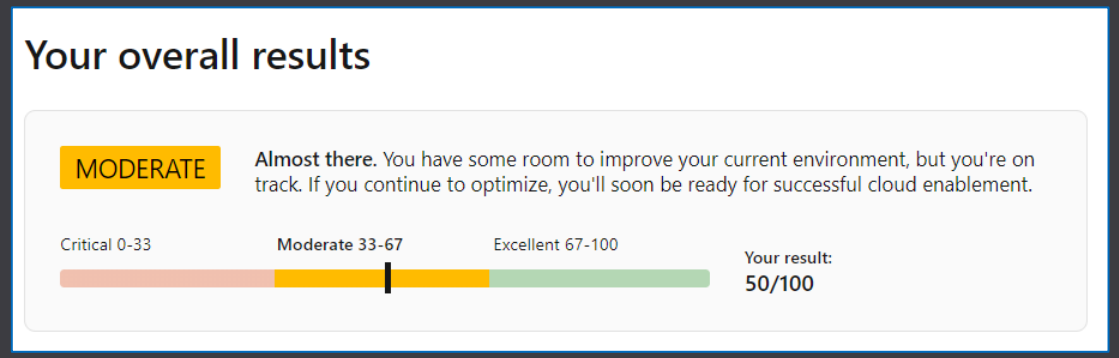
Operational task checklist

- ❑ Execute Runbooks on demand, on a schedule, or through a webhook.
- ❑ Use Azure Functions for running operational tasks written in a variety of languages
- ❑ Configure Azure Monitor autoscaling rules
- ❑ Configure Azure Kubernetes Service (AKS) scale operations
  - ❑ Pod scaling
  - ❑ Node scaling

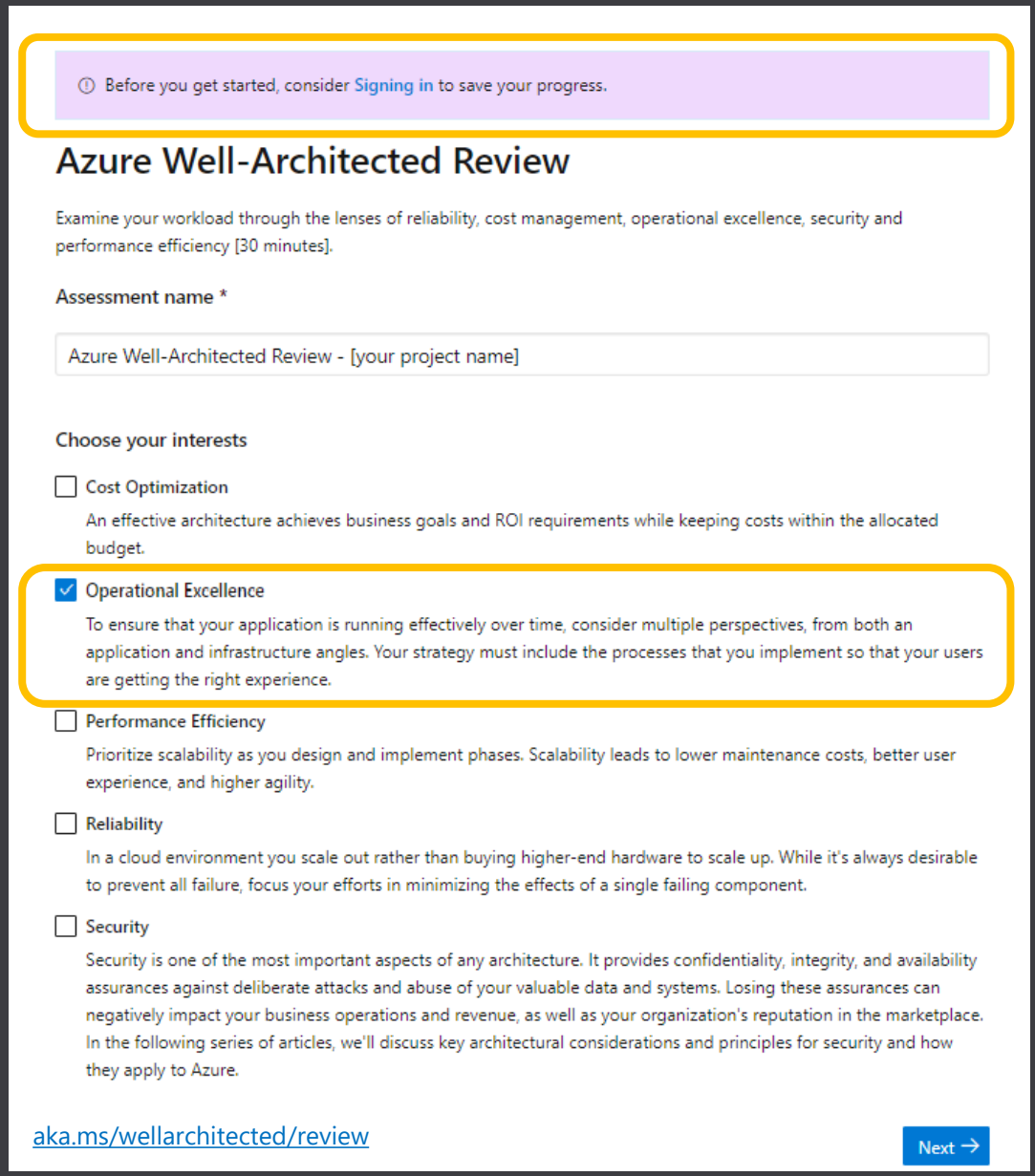
Azure Well-Architected Review

Assess workloads with the pillars of the Microsoft Azure Well-Architected Framework:

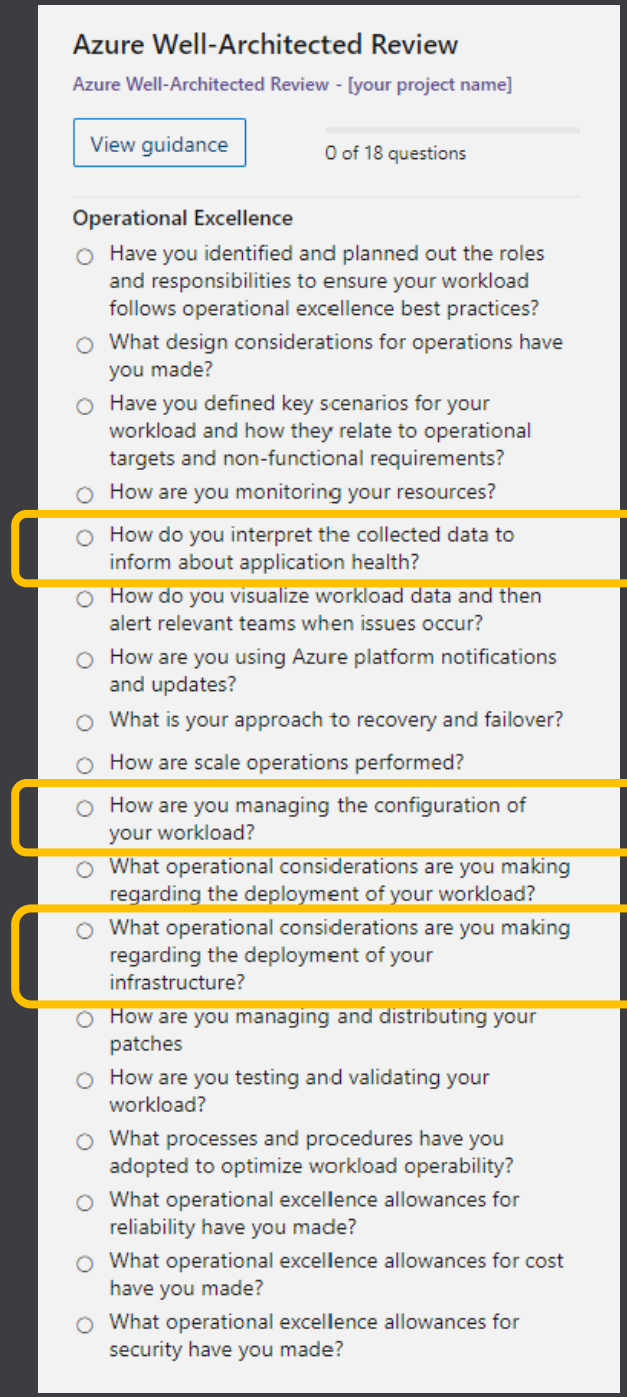
—Understand the Well-Architected level of your workload environment.



—Follow technical guidance for next steps of how to create and optimize your workloads.



Let's walkthrough some questions for Operational Excellence in the Well-Architected Review



How do you interpret the collected data to inform application health?

Log aggregation technologies should be used to collate logs and metrics across all workload components for later evaluation. Resources that logs are captured for may include Azure IaaS and PaaS services as well as 3rd-party appliances such as firewalls or anti-malware solutions used in the workload.

- A log aggregation technology, such as Azure Log Analytics or Splunk, is used to collect logs and metrics from Azure resources
- Azure Activity Logs are collected within the log aggregation tool
- Resource-level monitoring is enforced throughout the application
- Logs and metrics are available for critical internal dependencies
- Log levels are used to capture different types of application events.
- There are no known gaps in application observability that led to missed incidents and/or false positives.
- The workload is instrumented to measure customer experience.
- None of the above.

How are you managing the configuration of the workload?

Cloud-based applications often run on multiple virtual machines or containers in multiple regions and use multiple external services. How do you manage and store all your app's configuration settings, feature flags, and secure access settings?

- ❑ You monitor and take advantage of new features and capabilities of underlying services used in your workload.
- ❑ Application configuration information is stored using a dedicated management system such as Azure App Configuration or Azure Key Vault.
- ❑ Soft-Delete is enabled for your keys and credentials such as things stored in Key Vaults and Key Vault objects.
- ❑ Configuration settings can be changed or modified without rebuilding or redeploying the application.
- ❑ Passwords and other secrets are managed in a secure store like Azure Key Vault or HashiCorp Vault.
- ❑ The application uses Azure Managed Identities.
- ❑ The expiry dates of SSL certificates are monitored and there are processes in place to renew them.
- ❑ Components are hosted on shared application or data platforms as appropriate.
- ❑ Your workload takes advantage of multiple Azure subscriptions.
- ❑ The workload is designed to leverage managed services.
- ❑ None of the above.

What operational considerations are you making regarding infrastructure deployment?

As you provision and update Azure resources, application code, and configuration settings, a repeatable and predictable process will help you avoid errors and downtime.

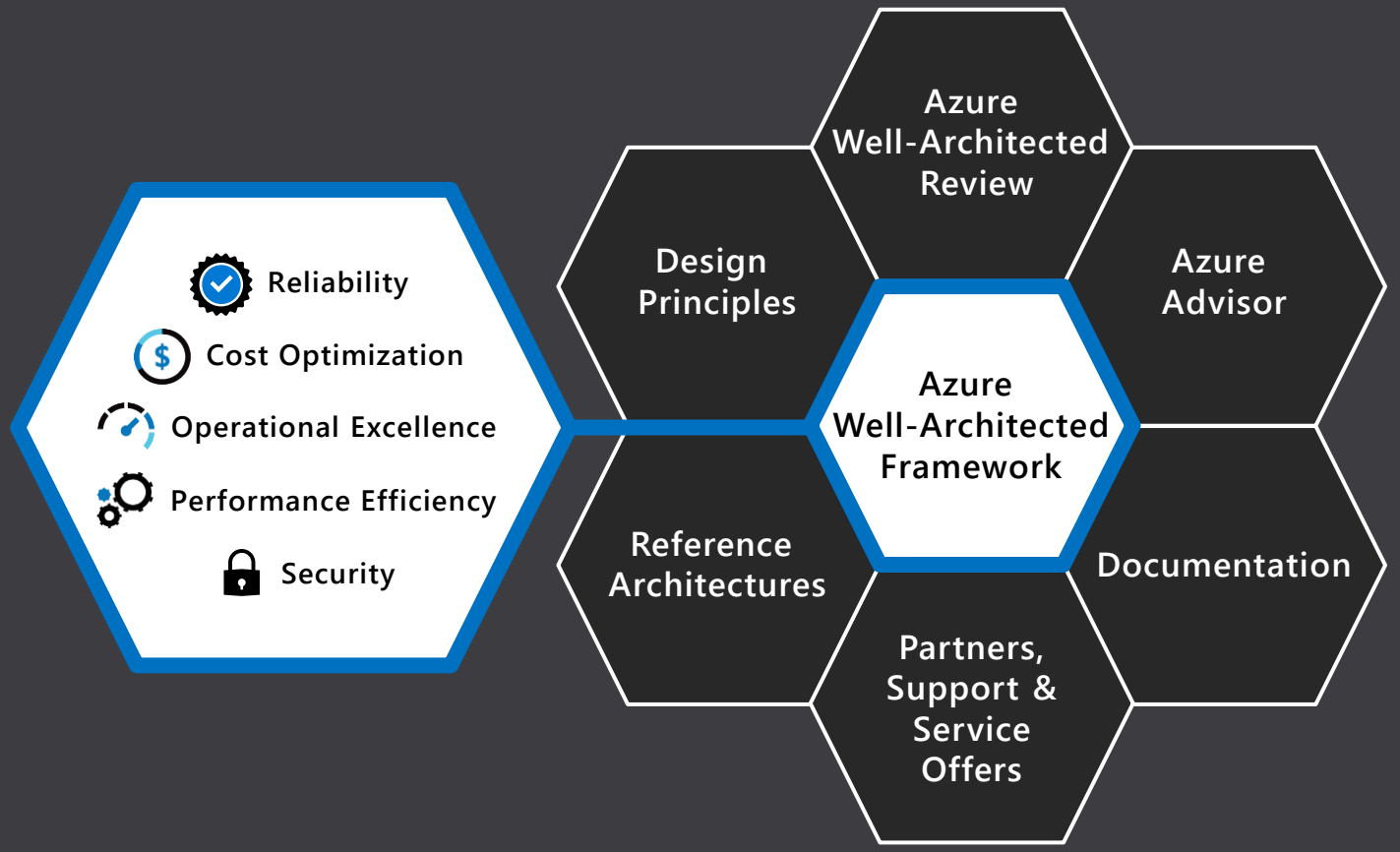
- ❑ The entire application infrastructure is defined as code
- ❑ No operational changes are performed outside of infrastructure as code
- ❑ Configuration drift is tracked and addressed
- ❑ The process to deploy infrastructure is automated
- ❑ Critical test environments have 1:1 parity with the production environment
- ❑ Direct write access to infrastructure is not possible and all resources are provisioned or configured through IaC processes.
- ❑ None of the above.

Microsoft

Reliability Pillar

Microsoft Well-Architected—

Build and manage high-performing workloads



Why is reliability important?

Because avoiding failure is impossible in the public cloud

Applications require resilience to respond to failures and deliver reliability

Reliability

The what—

- Ensuring availability of services = the goal for production systems.
- End goal = Maintain reliable systems with the appropriate level of availability (uptime).

Resilience

The how—

- How production systems achieve reliability.
- End goal = not to avoid all failures but to respond to failure in ways that avoid downtime and data loss.

Push Doctor

Customer:
Push Doctor
Industry:
Professional Services
Size:
50-999 employees
Country:
United Kingdom
Products and services:
Microsoft Azure
Microsoft Azure App Service
Microsoft Azure Application Gateway
Microsoft Azure Availability Zones
Microsoft Azure Monitor
Microsoft Azure Service Bus
Microsoft Azure SQL Database
Microsoft Power BI

[Read full story here]

Microsoft

"We've used Azure to build a resilient platform and help countless people get quick and easy healthcare access they can count on."
— Paul Smith, Enterprise Architect, Push Doctor

Situation:

Push Doctor, a patient/doctor video consultation platform based in the United Kingdom, needed highly available and scalable infrastructure that would provide the reliability its patients need to access remote healthcare support on their terms.

Solution:

Using Microsoft Azure platform as a service resources such as Azure App Service, Push Doctor's platform is now instantly scalable and highly secure, with an impressive 99.99 percent uptime. And thanks to duplicated workloads, it can seamlessly manage failovers.

Impact:

Push Doctor can now match patients with a general practitioner in a matter of hours, helping to potentially save the lives of people who would have otherwise waited much longer for a consultation—a service that has proved invaluable during the COVID-19 crisis.

Building reliable systems is a shared responsibility

Scope of Reliability Reviews

Your application

Your **app** or **workload**, built on the Azure platform.

Resiliency features

Optional Azure capabilities **you can** enable as needed—high availability, disaster recovery, and backup.

Reliable foundation

Core capabilities built into the Azure platform – how the foundation is designed, operated, and monitored to ensure availability.

Building reliable applications in the cloud

Enable systems to recover from failures and continue to function

Design for reliability

- Use Availability Zones where applicable to improve reliability and optimize costs.
- Design applications to operate when impacted by failures.
- Use the native resiliency capabilities of PaaS to support overall app reliability.
- Validate that required capacity is within Azure service scale limits and quotas.

Testing overall availability & resiliency

- Test regularly to validate existing thresholds, targets and assumptions.
- Verify how the end-to-end workload performs under failure conditions.
- Conduct load testing with expected peak volumes to test scalability and performance under load.
- Perform chaos testing by injecting faults.

Overall monitoring & diagnostics

- Define alerts that are actionable and effectively prioritized.
- Create alerts that poll for services nearing their limits and quotas.
- Use application instrumentation to detect and resolve performance anomalies.
- Troubleshoot issues to gain an overall view of application health.

Design for reliability

Principle: design applications to be resistant to failures

Use Availability Zones within a region

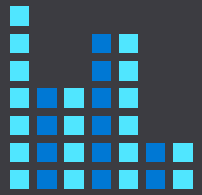
- If greater failure isolation than Availability Zones alone can offer, you should consider deploying to multiple regions.
- Multiple regions should be used for failover purposes in a disaster state.
- Additional costs—data, networking and the Azure Site Recovery service should be considered.

Design for failure recovery

- Resilient application architectures should be designed to recover gracefully from failures in alignment with defined reliability targets.
- Define an availability strategy to capture how the application remains available when in a failure state.
- Define a Business Continuity Disaster Recovery strategy for the application and/or its key scenarios.

Criteria for improving application reliability

- Use Platform as a Service (PaaS), which offers native resiliency capabilities to support overall application reliability.
- Design your application to automatically scale in and out.
- Review Azure subscription and service limits to validate that required capacity is within quotas.



Test for availability and resiliency

Principle: define, automate, and test operational processes

End-to-end workload testing

- Simulation testing involves creating real-life situations and demonstrates the effectiveness of proposed solutions.
- Use fault injection testing to check the system resiliency during failures—by triggering failures or by simulating them.
- Load testing is crucial for identifying failures that only happen under load, (e.g., an overwhelmed back-end database, or service throttling).

Build high availability & resiliency testing into strategy

- Resilient application architectures should be designed to recover gracefully from failures in alignment with defined reliability targets.
- Define an availability strategy to capture how the application remains available when in a failure state.
- Define a Business Continuity Disaster Recovery strategy for the application and/or its key scenarios.

Automate testing across BCDR strategy & prepare for failure

- Create and fully test a disaster recovery plan using the actual resources needed to restore functionality.
- Perform an operational readiness test for failover to the secondary region and for failback to the primary region.
- Codify the steps required to recover or failover to a secondary region to limit the impact of an outage.

Monitoring application health

Principle: define, automate, and test operational processes

Azure services & resources alerts & dashboards

- Azure Service Health provides a view into the health of Azure services and regions, as well as communications about outages and planned maintenance activities.
- Azure Resource Health provides information about the health of individual and is highly useful when diagnosing unavailable resources.
- Azure dashboards provides a consolidated view of data from Application Insights, Log Analytics, Azure Monitor metrics, and Service Health.

Scaling subscription & service targets

- If your application requires more storage accounts than are currently available in your subscription, create a new subscription with additional storage accounts.
- Identify scalability targets for VMs including VM size, number of disks, CPU, and memory.
- To avoid data throttling, review your Azure SQL Database requirements to ensure that they are adequate.

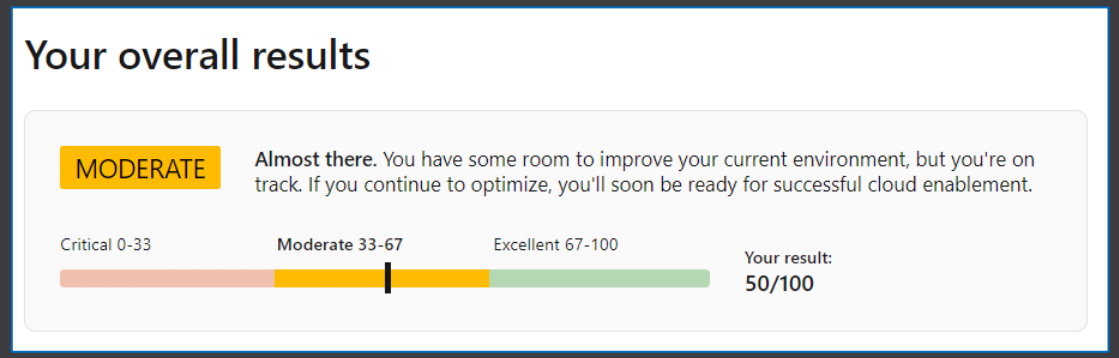
Fully test BCDR plan

- Create and fully test a disaster recovery plan using the actual resources needed to restore functionality.
- Perform an operational readiness test for failover to the secondary region and for failback to the primary region.
- Codify the steps required to recover or failover to a secondary region to limit the impact of an outage.

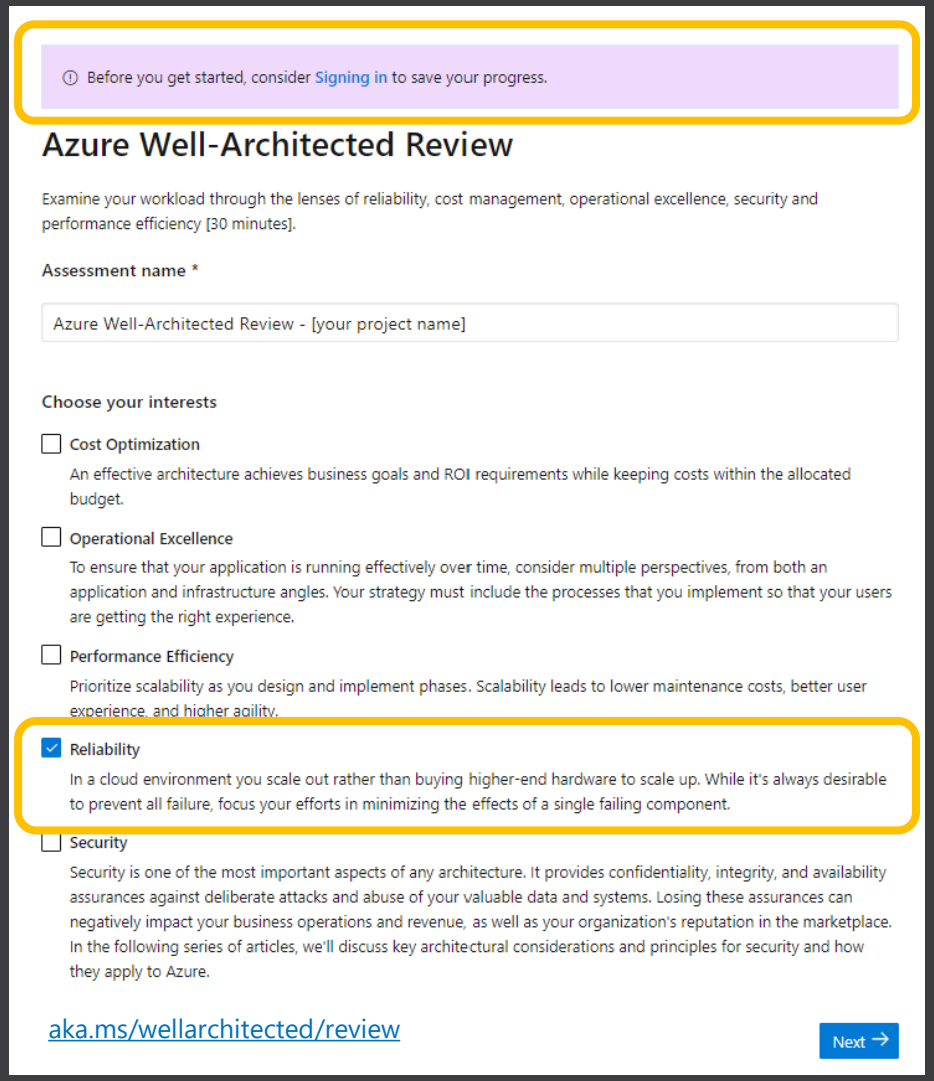
Azure Well-Architected Review

Assess workloads with the pillars of the Microsoft Azure Well-Architected Framework:

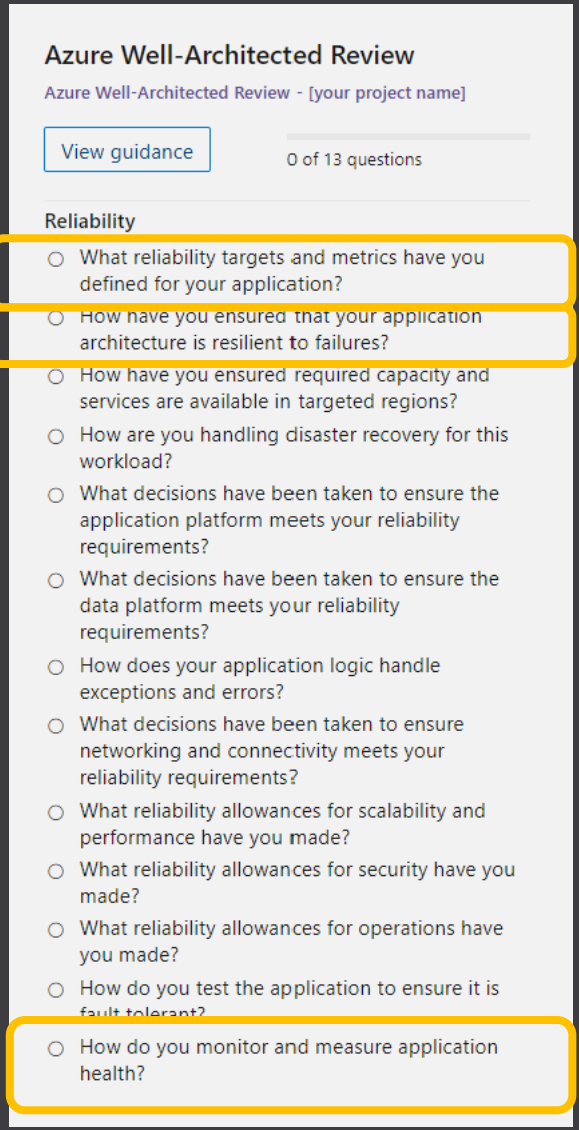
—Understand the Well-Architected level of your workload environment.



—Follow technical guidance for next steps of how to create and optimize your workloads.



Let's walkthrough some questions for **Reliability** in the Well-Architected Review



What reliability targets and metrics have you defined for your application?

Availability targets, such as Service Level Agreements (SLA) and Service Level Objectives (SLO), and Recovery targets, such as Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO), should be defined and tested to ensure application reliability aligns with business requirements.

- ❑ Recovery targets to identify how long the workload can be unavailable (Recovery Time Objective) and how much data is acceptable to lose during a disaster (Recovery Point Objective).
- ❑ Availability targets such as Service Level Agreements (SLAs) and Service Level Objectives (SLOs).
- ❑ Availability metrics to measure and monitor availability such as Mean Time To Recover (MTTR) and Mean Time Between Failure (MTBF).
- ❑ Composite SLA for the workload derived using the Azure SLAs for all relevant resources.
- ❑ SLAs for all internal and external dependencies.
- ❑ Independent availability and recovery targets for critical application subsystems and scenarios.
- ❑ None of the above.

How have you ensured that your application architecture is resilient to failures?

Resilient application architectures should be designed to recover gracefully from failures in alignment with defined reliability targets.

- ❑ Deployed the application across multiple regions.
- ❑ Removed all single points of failure by running multiple instances of application components.
- ❑ Deployed the application across Availability Zones within a region.
- ❑ Performed Failure Mode Analysis (FMA) to identify fault-points and fault-modes.
- ❑ Planned for component level faults to minimize application downtime.
- ❑ Planned for dependency failures to minimize application downtime.
- ❑ None of the above.

How do you monitor and measure application health?

Monitoring and measuring application availability is vital to qualifying overall application health and progress towards defined reliability targets.

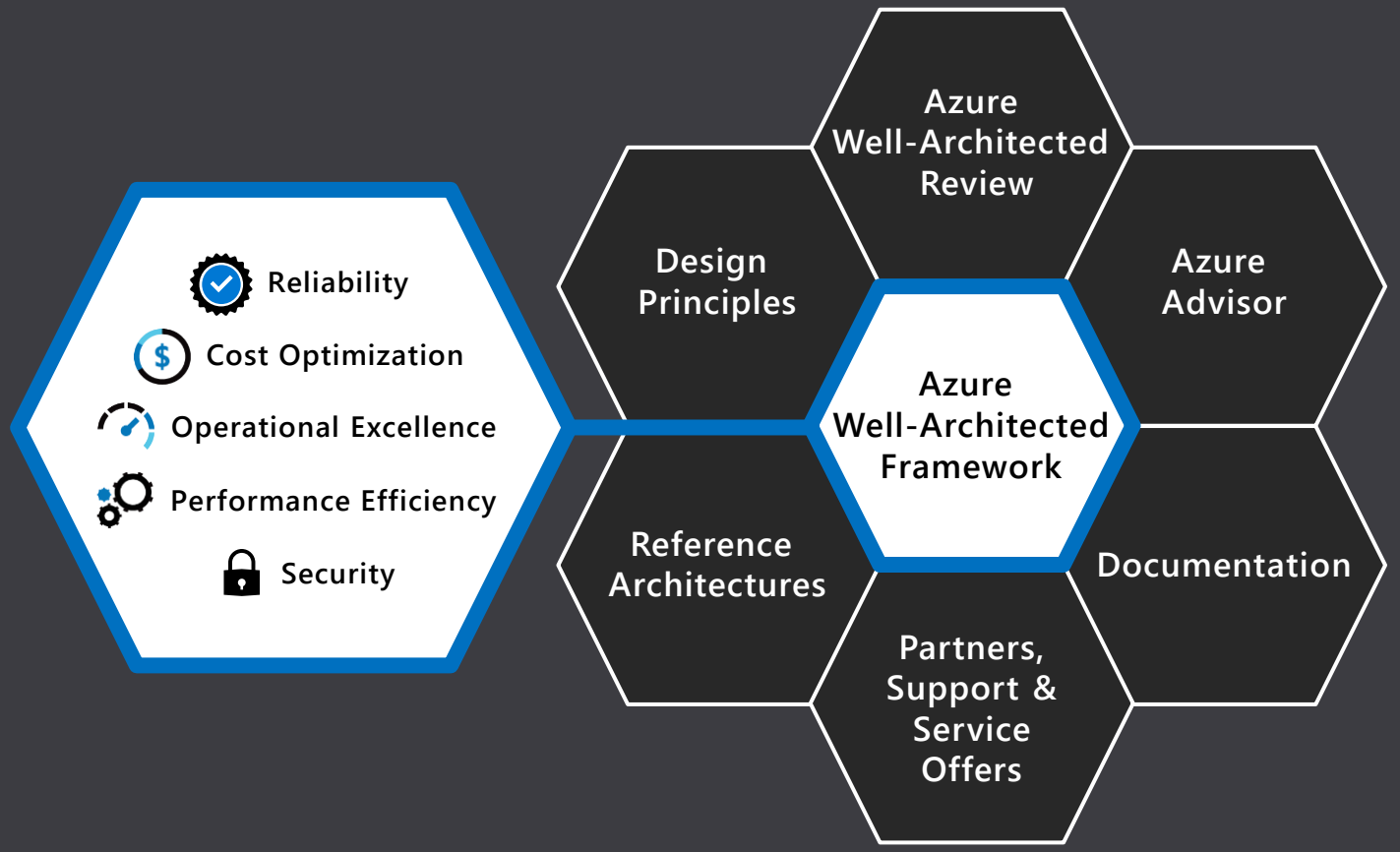
- ❑ The application is instrumented with semantic logs and metrics.
- ❑ Application logs are correlated across components.
- ❑ All components are monitored and correlated with application telemetry.
- ❑ Key metrics, thresholds, and indicators are defined and captured.
- ❑ A health model has been defined based on performance, availability, and recovery targets and is represented through monitoring dashboard and alerts.
- ❑ Azure Service Health events are used to alert on applicable Service level events.
- ❑ Azure Resource Health events are used to alert on resource health events.
- ❑ None of the above.

Microsoft

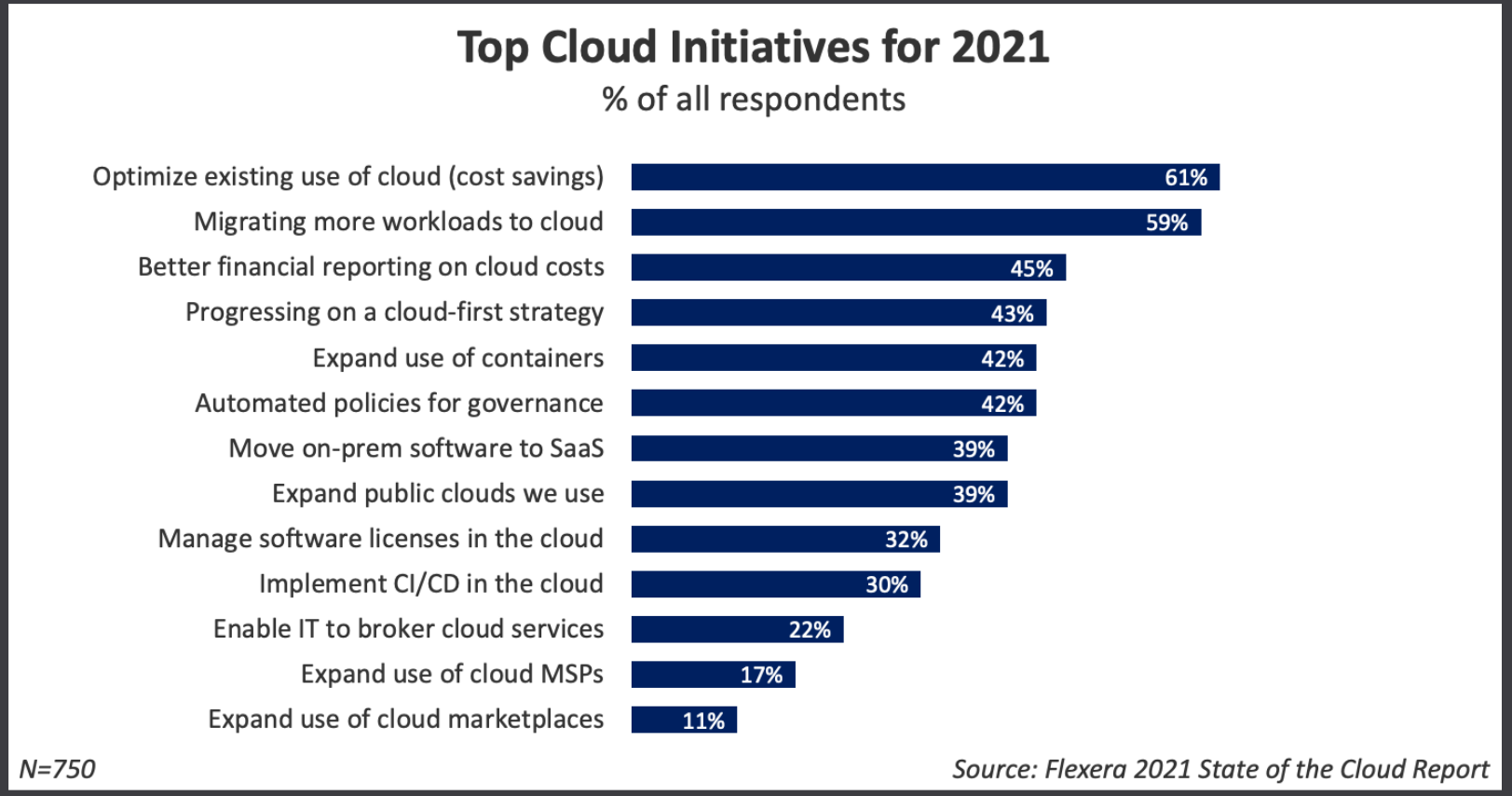
Cost Optimization Pillar

Microsoft Well-Architected—

Build and manage high-performing workloads



Cost optimization = top cloud initiative for the fifth year running



H&R BLOCK®

Customer:
H&R Block

Industry:
Professional Services

Size:
10,000+ employees

Country:
United States

Products and services:
Microsoft Azure
Microsoft Azure Advisor
Microsoft Azure Cost Management and Billing
Microsoft Azure Well-Architected Framework

Microsoft

"Our monthly spend year-over-year is nearly flat, while we now have approximately 30 percent more of our total compute in the cloud. Thanks to our partnership with Microsoft, our team has learned valuable techniques and strategies to continue optimizing our spend."

—Paul Clark, Director of Cloud, H&R Block

Situation:

As a leader in the effort to modernize the tax industry, H&R Block wanted to optimize its cloud infrastructure, and provide better service for its customers.

Solution:

By engaging with its Microsoft account team and operationalizing conceptual pillars of the Azure Well-Architected Framework, the company was able to optimize its investment—replatforming to cloud-native services and modernizing its operating models.

Impact:

H&R Block is now equipped to take control of its monthly spend and able to move its total compute to the cloud, using its capabilities to benefit its business and customers.

Optimize costs with tools, offers, and guidance

Cost optimization offers guidelines—accelerating time to market, while avoiding capital-intensive solutions

Understand and forecast your costs

- Monitor your bill, set budgets, and allocate spending to teams and projects with Azure Cost Management + Billing
- Forecast costs for future investments with the Azure pricing and TCO calculator

Cost optimize your workloads

- Optimize your resources with Azure Advisor
- Follow best practices for workload design with the Azure Well-Architected Framework
- Save with Azure offers and licensing terms like the Azure Hybrid Benefit and Reservations

Control your costs

- Establish spending objectives and policies using the Microsoft Cloud Adoption Framework for Azure
- Implement cost controls in Azure Policy so your teams can go fast while complying with policy

Optimize your costs with tools, offers, and guidance

Principle: Monitor and optimize

Use alerts to monitor usage and spending

- Budget alerts notify you when spending reaches predetermined thresholds.
- Credit alerts notify you when your Azure Prepayment is consumed.
- Department spending quota alerts notify you when quotas are reached.

Auto-scaling policies provide cost savings

- When workloads are highly variable, choose smaller VM instances, then scale out, rather than up, to get the needed performance.
- Many applications can be made stateless, then auto-scaled for cost benefits.

Reserved instances can reduce costs

- Use Azure Reservations to lower costs by pre-paying for capacity.
- Analyze existing pay-as-you-go usage data in Azure Portal before opting into reserved instances.

Optimize your costs with tools, offers, and guidance

Principle: Keep within cost constraints

Develop a cost model

- Map your organization's needs to specific offerings.
- Start with high-level requirements before considering design.
- Geographic and security decisions can have a huge impact on your costs.

Capture requirements

- Break down high-level goals into functional requirements,
- For each functional requirement, define metrics to estimate costs.

Cost tradeoffs

- Determine if the cost of high availability exceeds acceptable downtime.
- Increasing security of the workload will increase cost.
- Systems monitoring and automation might increase the cost initially but will reduce cost over time.

Design Checklist

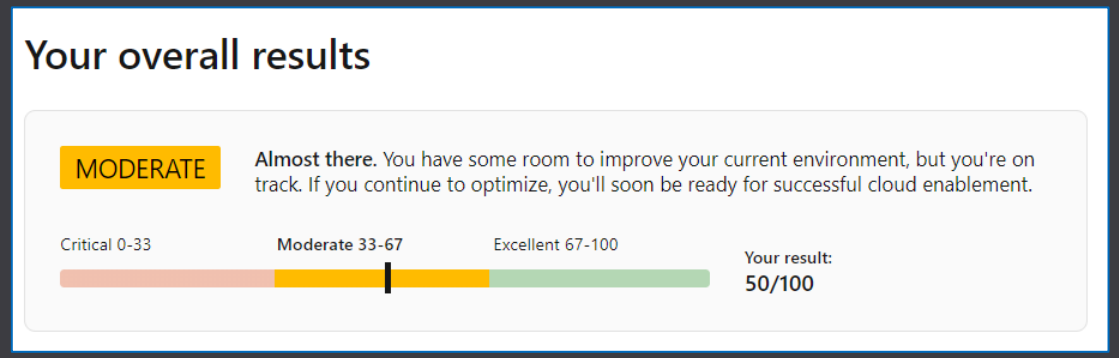
Principle: Aim for scalable costs

- ❑ Consider tradeoffs of cost savings versus security, scalability, resilience, and operability
- ❑ Choose managed services whenever possible
- ❑ Compare consumption-based pricing with pre-provisioned costs
- ❑ Choose an appropriate subscription level
- ❑ Use proof-of-concept deployments
- ❑ Optimize data-transfer
- ❑ Reduce server load

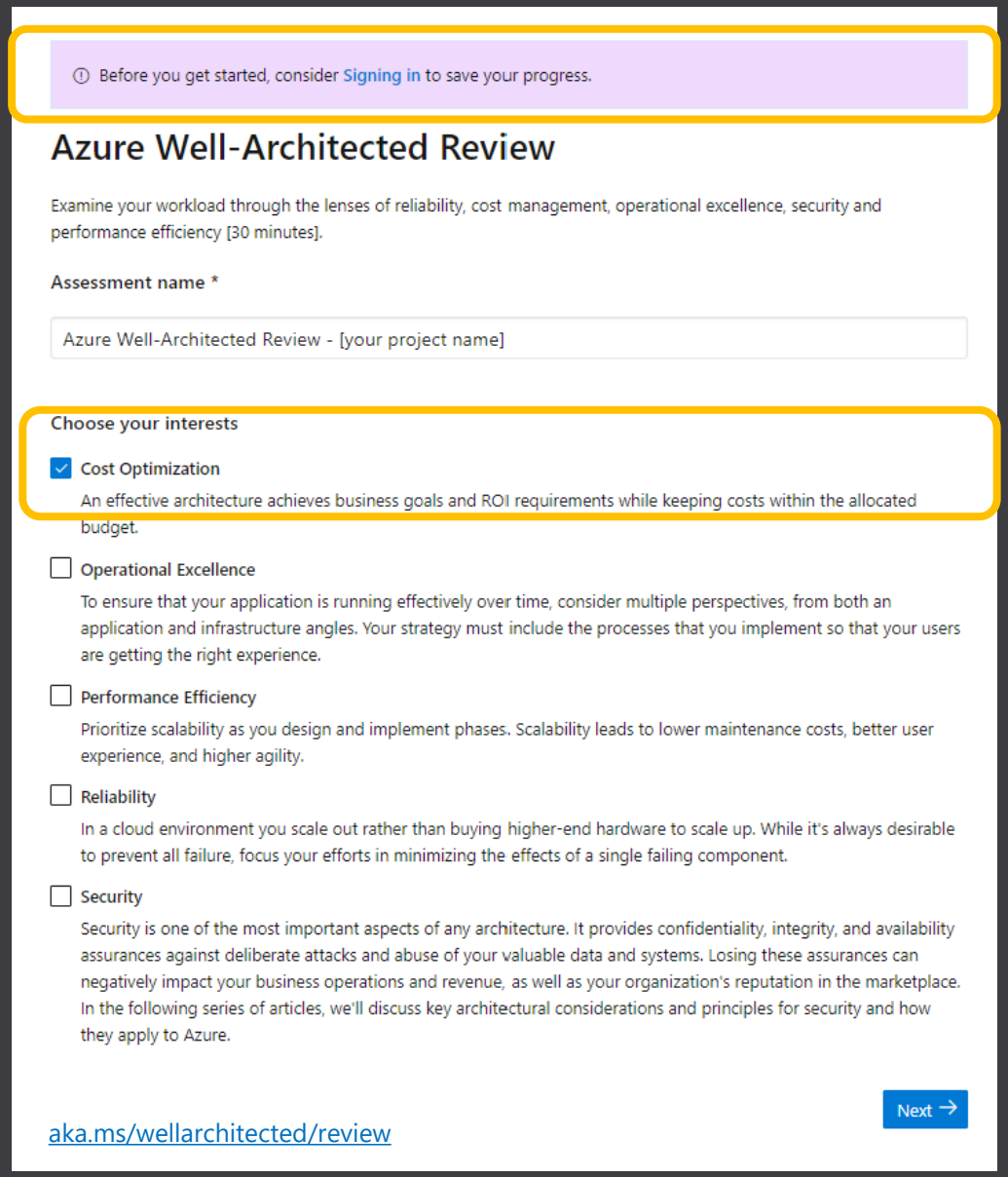
Azure Well-Architected Review

Assess workloads with the pillars of the Microsoft Azure Well-Architected Framework:

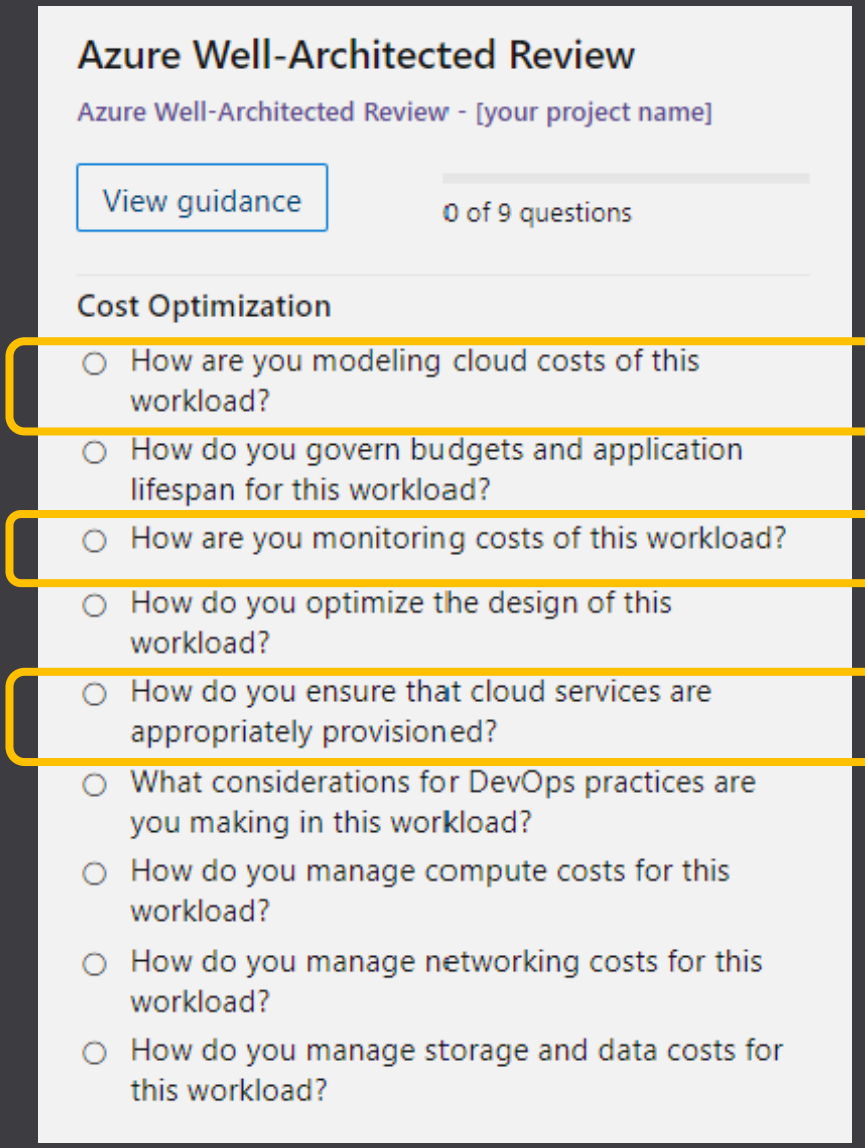
—Understand the Well-Architected level of your workload environment.



—Follow technical guidance for next steps of how to create and optimize your workloads.



Let's walk through some questions for Cost Optimization in the Well-Architected Review



How are you modeling cloud costs?

Cost modeling is an exercise where you create logical groups of cloud resources that are mapped to the organization's hierarchy and then estimate costs for those groups. The goal of cost modeling is to estimate the overall cost of the organization in the cloud.

- ❑ Cloud costs are being modeled for this workload.
- ❑ The price model of the workload is clear.
- ❑ Critical system flows through the application have been defined for all key business scenarios
- ❑ There is a well-understood capacity model for the workload.
- ❑ Internal and external dependencies are identified, and cost implications understood.
- ❑ Cost implications of each Azure service used by the application are understood
- ❑ The right operational capabilities are used for Azure services.
- ❑ Special discounts given to services or licenses are factored in when calculating new cost models for services being moved to the cloud.
- ❑ Azure Hybrid Use Benefit is used to drive down cost in the cloud.
- ❑ None of the above.

How are you monitoring costs?

Consider the metrics for each resource in the workload. For each metric, build alerts on baseline thresholds.

- ❑ Alerts are set for cost thresholds and limits.
- ❑ Specific owners and processes are defined for each alert type.
- ❑ Application Performance Management (APM) tools and log aggregation technologies are used to collect logs and metrics from Azure resources.
- ❑ Cost Management Tools (such as Azure Cost Management) are being used to track spending in this workload.
- ❑ None of the above.

How do you ensure that cloud services are appropriately provisioned?

Deployment of cloud resources of a workload is known as provisioning.

- ❑ Performance requirements are well-defined.
- ❑ Targets for the time it takes to perform scale operations are defined and monitored.
- ❑ The workload is designed to scale independently.
- ❑ The application has been designed to scale both in and out.
- ❑ Application components and data are split into groups as part of your disaster recovery strategy.
- ❑ Tools (such as Azure Advisor) are being used to optimize SKUs discovered in this workload.
- ❑ Resources are reviewed weekly or bi-weekly for optimization.
- ❑ Cost-effective regions are considered as part of the deployment selection.
- ❑ Dev/Test offerings are used correctly.
- ❑ Shared hosting platforms are used correctly.
- ❑ None of the above.

Next steps

- Assess your workload with a Well-Architected Review: https://aka.ms/wellarchitected/review
- Gather technical recommendations and optimize deployments with Azure Advisor: https://aka.ms/azureadvisor
- Learn how to build great solutions with Well-Architected Framework: https://docs.microsoft.com/en-us/learn/