Microsoft

SRE 價值再進化！快速推動高效能企業系統環境

Thomas Huang
Microsoft Taiwan
Cloud Solution Architect

Site Reliability Engineering is an engineering discipline devoted to helping an organization sustainably achieve the appropriate level of **reliability** in their systems, services, and products.

reliability

Site Reliability Engineering is an engineering discipline devoted to helping an organization sustainably achieve the appropriate level of reliability in their systems, services, and products.

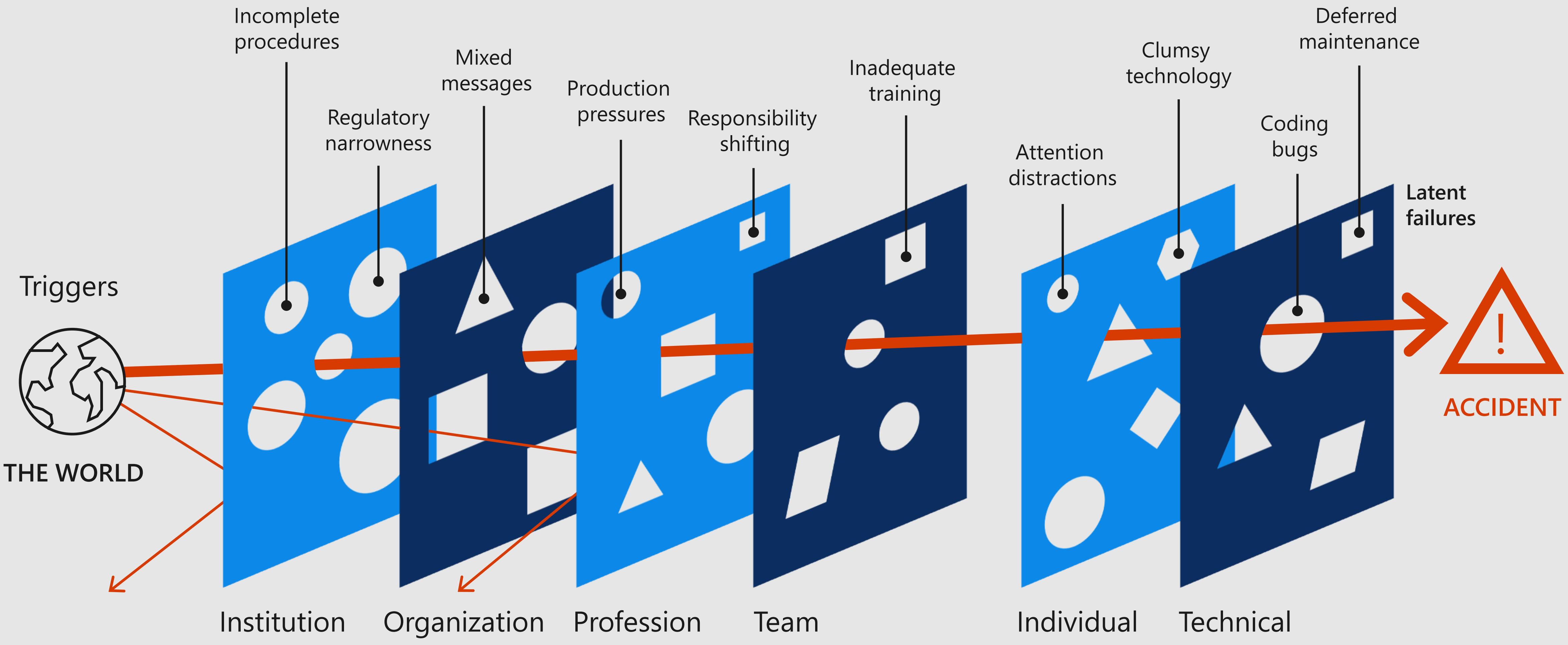
appropriate

sustainably

https://docs.microsoft.com/en-us/azure/site-reliability-engineering/

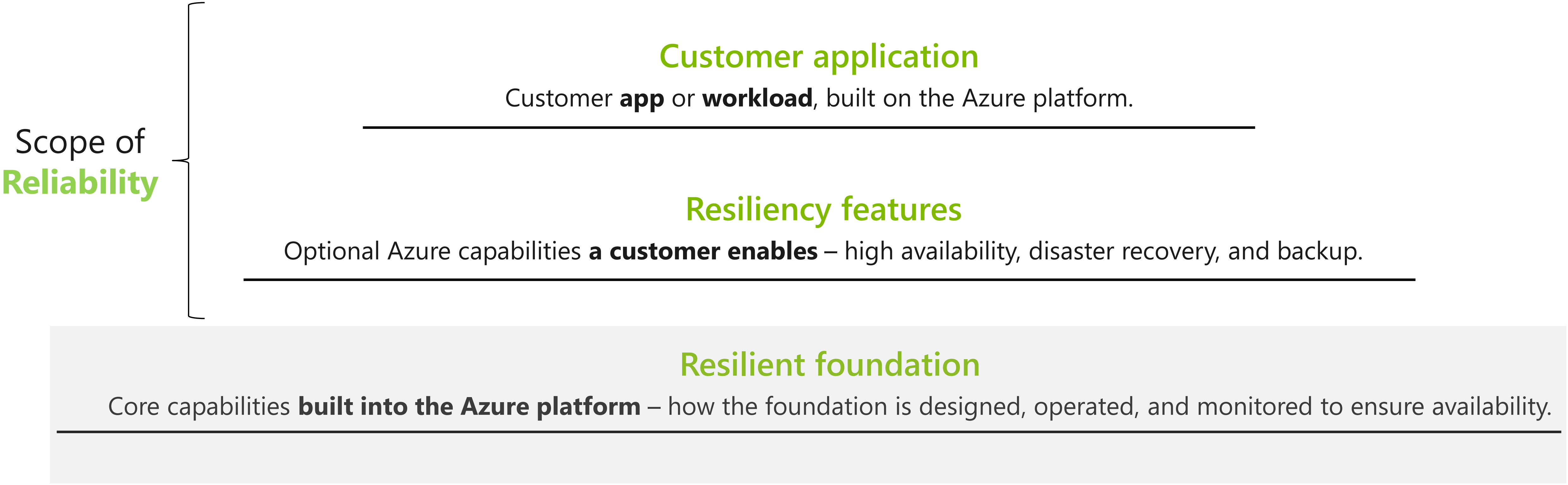
Why do bad things happen?

Modified from
Reason, 1991



LAYERS OF DEFENSE

Building reliable systems is a **shared responsibility**



Scope of **Reliability**

Customer application

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

Resiliency features

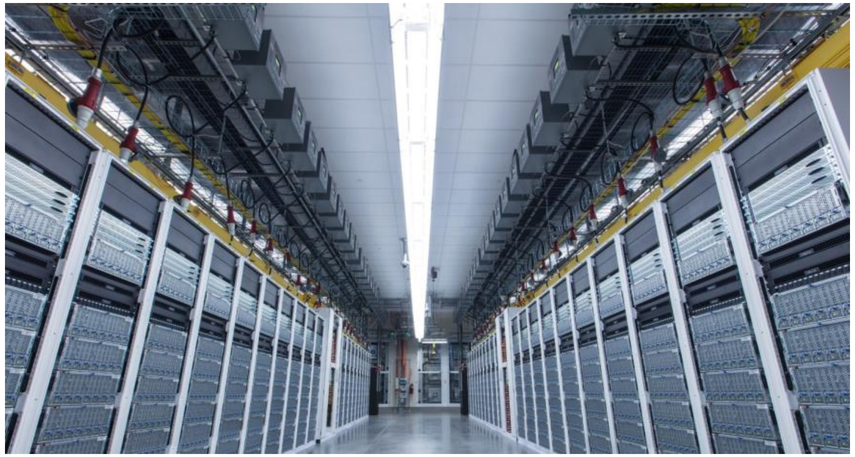
Optional Azure capabilities **a customer enables** – high availability, disaster recovery, and backup.

Resilient foundation

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

Resilient foundation

Our investments in global infrastructure, service management, and ensuring transparency



Design

Global network

Datacenter infrastructure

Storage protection

Operate

Safe deployment

Maintenance & control

ML & failure prediction

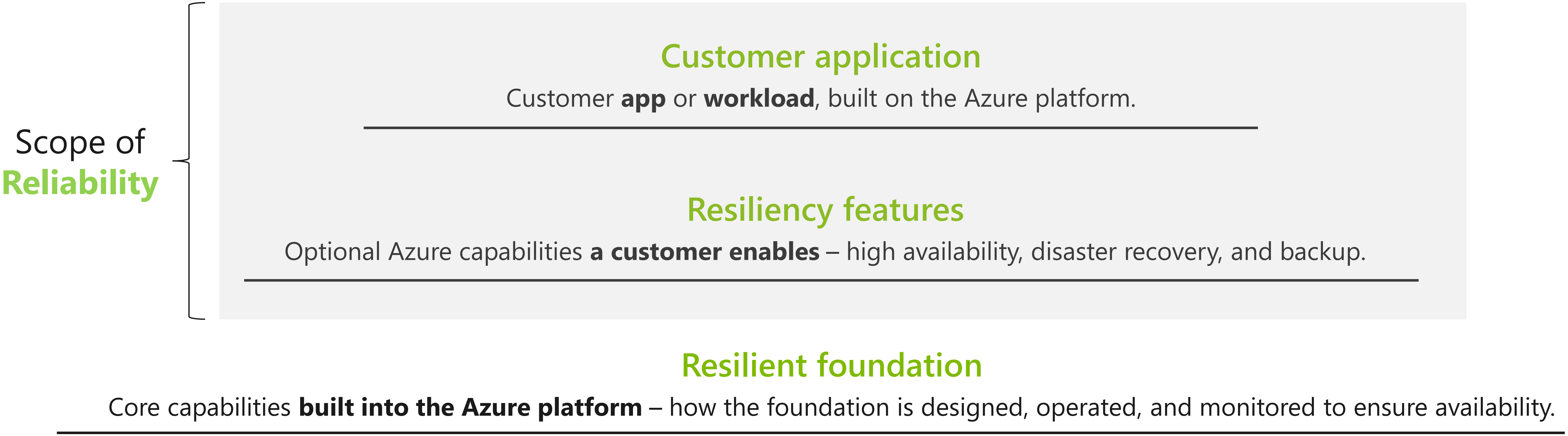
Observe

Communications philosophy

Service health & alerts

Scheduled events

Building reliable systems is a **shared responsibility**



Scope of **Reliability**

Customer application

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

Resiliency features

Optional Azure capabilities **a customer enables** – high availability, disaster recovery, and backup.

Resilient foundation

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

Why is Reliability Important?

Failures happen.

Reliable applications require *resilience*

Reliability

Reliability is the '**what**'.

It is the goal for production systems, to ensure availability of their services.

The goal is to maintain reliable systems, with the appropriate level of availability/uptime.

Resilience

Resilience is the '**how**'.

It is the way in which production systems can achieve reliability.

The objective is not to avoid any and all failures – it is to ***respond to failure in a way that avoids downtime and data loss***.

Microsoft Azure Well-Architected Framework

Architecture guidance and best practices to optimize the quality of Azure workloads, based on 5 aligned and interconnected pillars

- Cost Optimization
- Operational Excellence
- Performance Efficiency
- **Reliability**
- Security

Learn more https://aka.ms/architecture/framework

Key Stakeholders

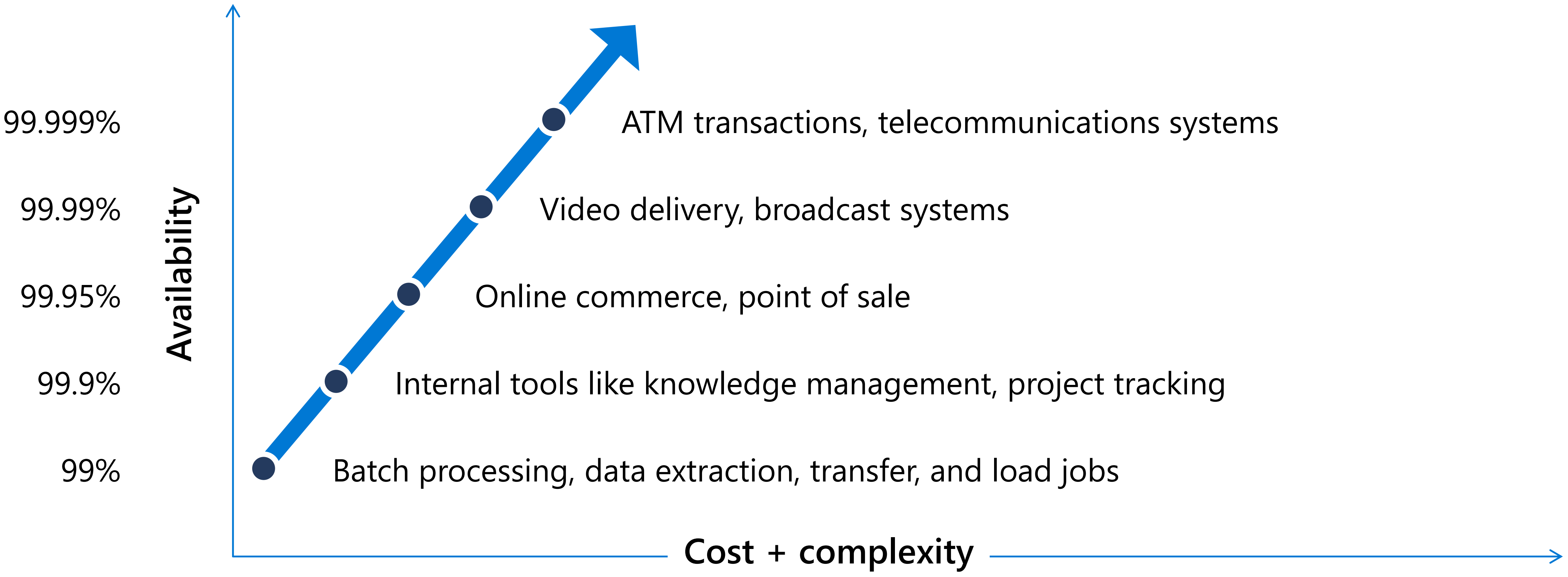
- □ Cloud Architect
- □ SecOps
- □ Project Manager
- □ Identity & Access
- □ Data Architect
- □ Network engineering
- □ Solution owner
- □ DevOps manager
- □ SRE Lead
- □ Governance
- □ Compliance manager

Common Definitions

- Important Targets and Measures
  - **RTO** (Recovery Time Objective) – The duration of an outage after which the system is expected to have recovered
  - **RPO** (Recovery Point Objective) – The duration of data loss that is allowable during an outage
  - **SLA** (Service Level Agreement) – The availability, usually as a percent, that the system or component contractually provides, often within a specified scope.
  - **SLO** (Service Level Objective) – The availability, like SLA, that the system internally sets as an objective. This is usually not published but must be greater-or-equal to the SLA
  - **Attainment Interval** – The period over which the SLA is measured (for Azure, one month)
  - **MTTD** (Mean Time To Detect) – Average time to detect a failure after it has occurred
  - **MTTR** (Mean Time To Recover) – Average time to recover from a failure once it occurs
- We care most about obtaining and working on **SLA/SLO/RTO/RPO**

Application availability needs

Examples of applications commonly seen at each availability tier



Availability

99.999%	ATM transactions, telecommunications systems
99.99%	Video delivery, broadcast systems
99.95%	Online commerce, point of sale
99.9%	Internal tools like knowledge management, project tracking
99%	Batch processing, data extraction, transfer, and load jobs

Cost + complexity

Strategies to Reduce RTO

- In many cases, the straightforward SLA will *not* meet the RTO
- A first step is to improve stage-by-stage
  - Use the checklists by technology
  - There are strategies for web, application, load balancing, network, database and more
  - Focus on automatic removal of failed components to restore service (e.g. failover)
- Focus on “blast radius” by creating slices of application that can fail separately
  - Smaller failures are usually much less impactful and don’t require multiple regions
  - This may also help with Blue/Green deployments
- Mean Time To Detection (MTTD) is an important measure
  - You can’t fix what isn’t detected
- Understand where manual intervention is needed and make sure it’s reasonable
  - An RTO of 5-minutes with manual intervention is not possible
- Assume that some repairs may require deployment
  - Don’t ever skip analysis of operations and deployment pipelines

Failure Mode Analysis (FMA)

A process for building resiliency into a system, by identifying possible failure points

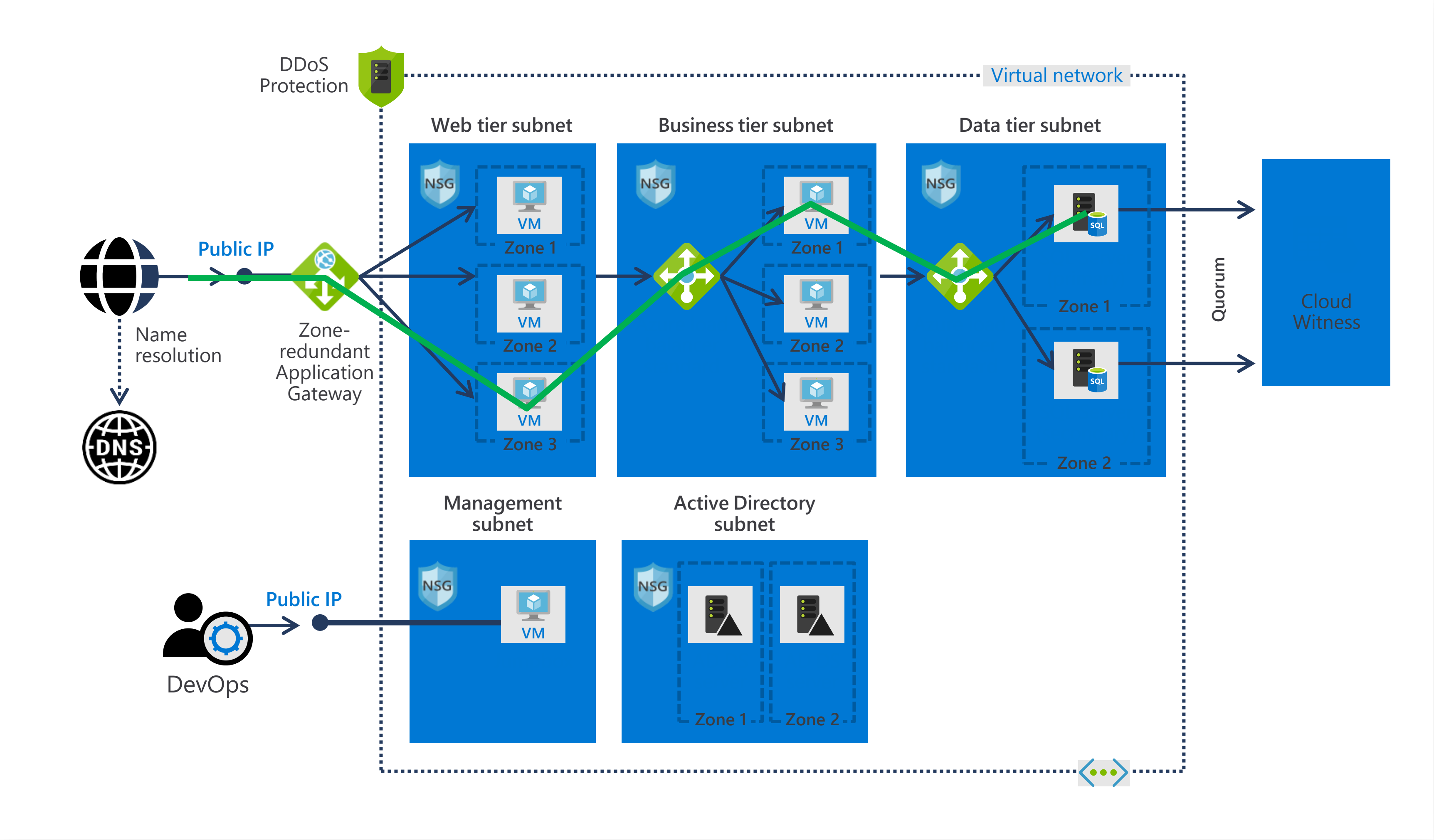
FMA should be part of the architecture/design phases, to build failure recovery in from the outset.

Here is the general process to conduct an FMA:

1. Identify all of the components in the system.
2. For each component, identify potential failures that could occur.
3. Rate each failure mode according to its overall risk.
4. For each failure mode, determine how the application will respond and recover.

The ***Azure Architecture Center*** *includes a catalog of potential failure modes and their mitigation steps. The catalog is organized by technology or Azure service, plus a general category for application-level design. The catalog is not exhaustive, but covers many of the core Azure services.*

Failure Mode Analysis Walk-Through



- Identify each potential failure
- Rate failure according to overall risk
- Determine how application will respond and recover

Microsoft Azure

Reliability with Microsoft Azure

Building reliable systems on Azure is a shared responsibility. Microsoft is responsible for the reliability of the cloud platform, including our global network and datacenters. Our customers and partners are responsible for the reliability of their cloud applications, using architectural best practices based on the requirements of each workload.

No matter what your service-level objectives are, Azure can help you achieve your organization's reliability goals. Design and operate mission-critical systems with confidence by taking advantage of built-in features for high availability, disaster recovery, and backup.

High availability

Maintain acceptable continuous performance despite temporary failure in services, hardware, or datacenters—as well as fluctuation in load—using Azure Availability Zones and availability sets.

Disaster recovery

Protect against the loss of an entire region through asynchronous replication for failover of virtual machines and data using services like geo-redundant storage and Azure Site Recovery.

Backup and restore

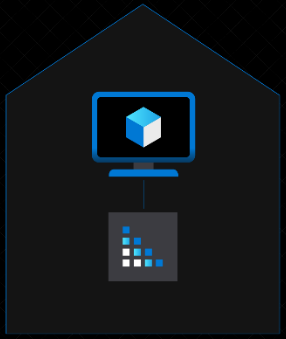
Replicate virtual machines and data to one or more regions using Azure Backup, and conduct self-service recoveries of Azure VMs or disks from a secondary region during an outage.

Single VM

Improve the availability of **single-instance VMs** by using premium/ultra disks to qualify for an availability SLA.

99.9% SLA (3 9s)
VM availability (monthly)

Single VM 🔗
with premium/ultra disks



99.999999999% (11 9s)
Storage durability (annually)

Locally Redundant Storage (LRS)* 🔗

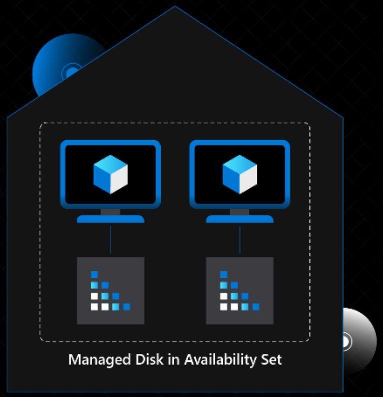
- Virtual machine | Compute options
- Storage account | Storage options
- * Optional: Azure Backup
- 🔗 Link

Local redundancies

Protect against failures with redundancy **within a single datacenter** in the event of hardware malfunctions or software update cycles.

99.95% SLA (3½ 9s)
VM availability (monthly)

Availability Set (2+ VMs) 🔗
within a datacenter



99.999999999% (11 9s)
Storage durability (annually)

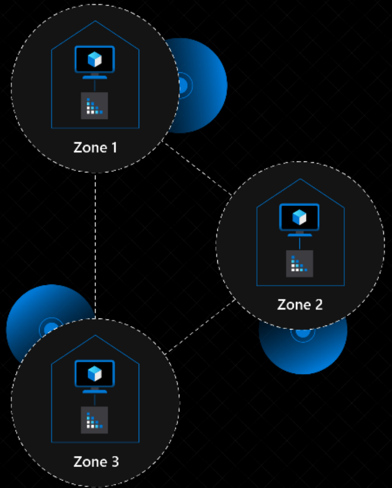
Locally Redundant Storage (LRS) with Azure Managed Disks* 🔗

Zonal redundancies

Protect against datacenter failures through redundancy **within a single region** in the event of power, cooling, or networking issues.

99.99% SLA (4 9s)
VM availability (monthly)

Availability Zones (2+ VMs) 🔗
within a region



99.9999999999% (12 9s)
Storage durability (annually)

Zone-Redundant Storage (ZRS) 🔗

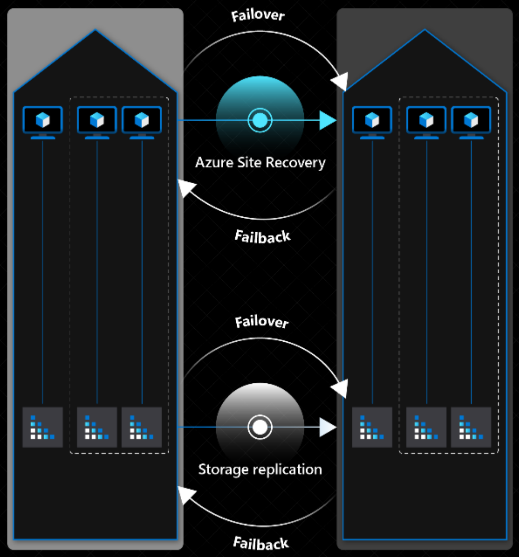
Regional redundancies

Protect against entire-region failures with redundancy **beyond a single region** in the event of a tornado, earthquake, or other large-scale disaster.

Industry-Leading
RPO and RTO

Azure Site Recovery 🔗

Primary region
Secondary region



99.99999999999999% (16 9s)
Storage durability (annually)

Geo-Redundant Storage (GRS)* 🔗

Download this infographic at www.aka.ms/ReliabilityInfographic

How can we test Failure Modes?

- “Natural Causes”
  - Environment is configured to produce the failure
    - e.g. Create a file and then test an API trying to create an existing file so it can fail
  - This approach is very limited and fragile
- Fault Injection
  - Errors are injected from the dependencies of the component, causing a failure mode
  - This approach allows for a much wider range of testing, tied to implementation
  - Start simple – Don’t overthink or overbuild from the beginning
- Common methods for injecting Azure service “faults”
  - Compute: Role restarts, Scale-out, Scale-in
  - Networking: NSG rules to block/unblock communication to dependent services
  - Storage: Customer initiated failover
  - SQL: Manual failover of SQL database instances

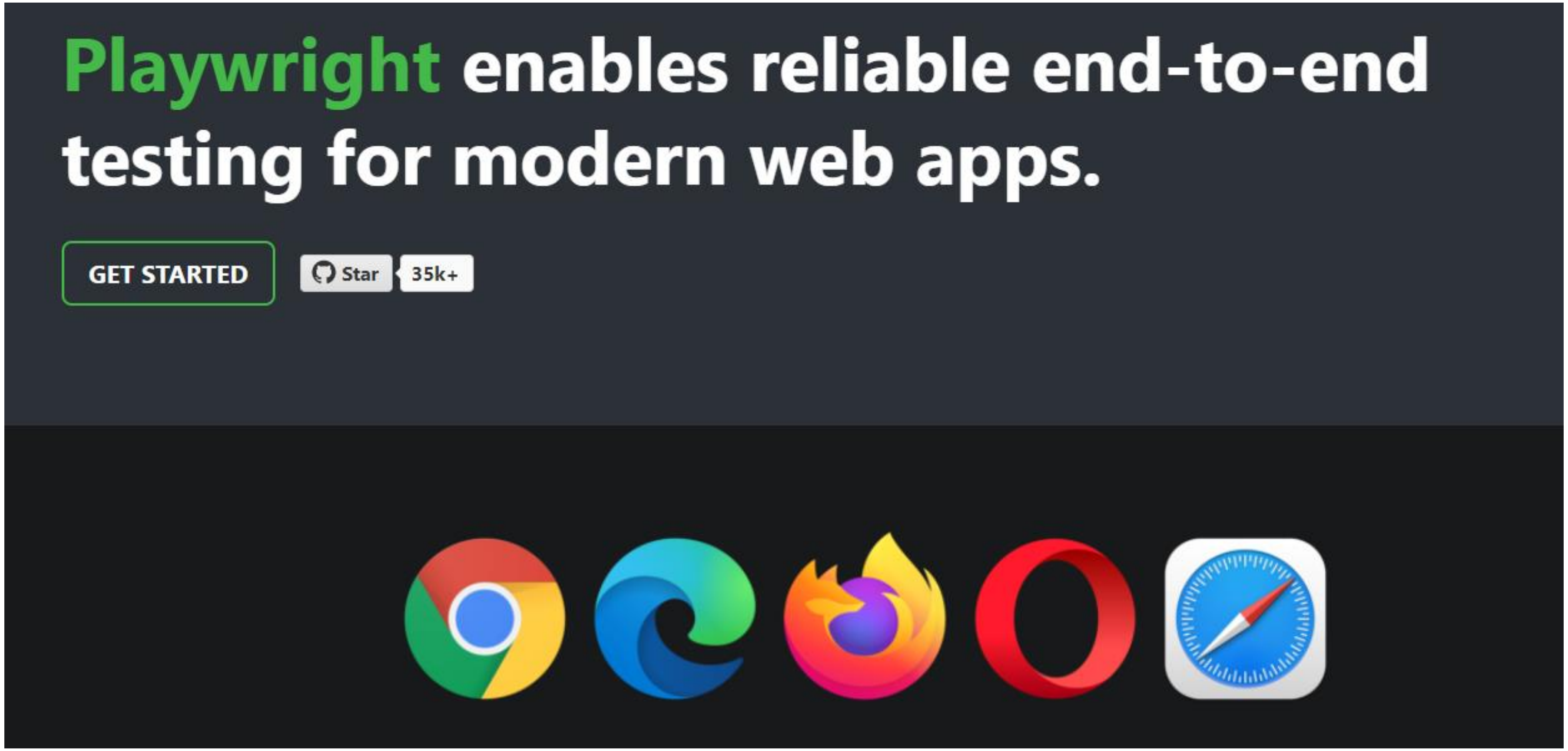
Testing for reliability

Regular testing should be performed as part of each major change and if possible, on a regular basis to validate existing thresholds, targets and assumptions.

Testing should also ensure the validity of the health model, capacity model, and operational procedures.

- ✓ Test regularly to validate existing thresholds, targets and assumptions
- ✓ Automate testing as much as possible
- ✓ Verify how the end-to-end workload performs under intermittent failure conditions
- ✓ Test the application against critical non-functional requirements for performance
- ✓ Conduct load testing with expected peak volumes to test scalability and performance under load
- ✓ Perform chaos testing by injecting faults

Playwright



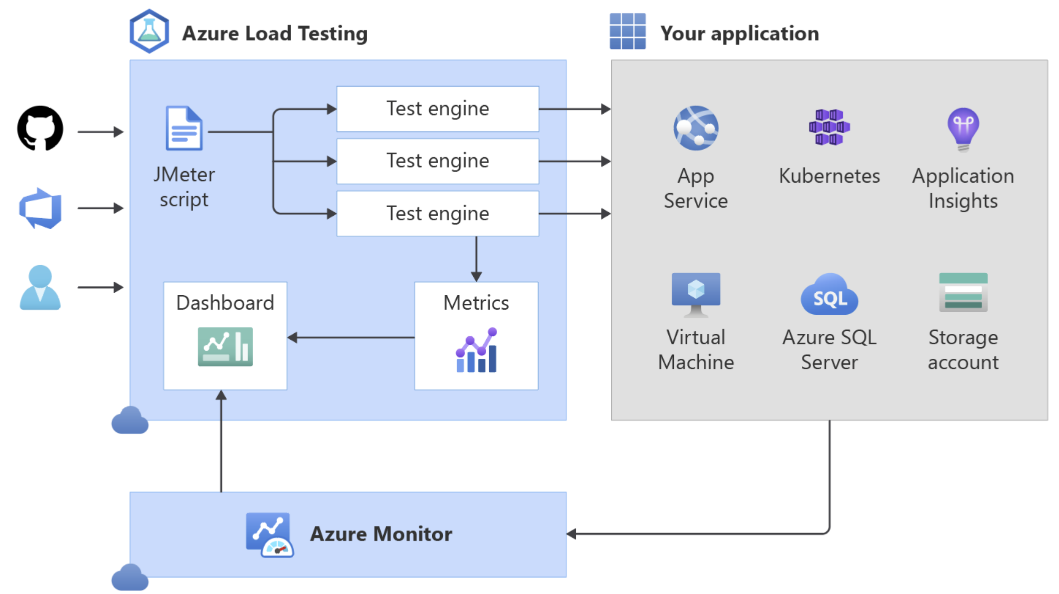
Playwright enables reliable end-to-end testing for modern web apps.

https://playwright.dev/
https://github.com/microsoft/playwright
https://www.youtube.com/watch?v=VMl8aV-ddMA

Azure Load Testing

Generate high-scale load without the need for complex infrastructure

- Run existing test scripts with high-fidelity JMeter support
- Eliminate infrastructure needs with a fully managed service
- Experience frictionless testing on Azure



Chaos engineering

The practice of subjecting cloud applications and services to real world failures and dependency disruptions in order to build and validate resilience.

Fault injection

The deliberate introduction of a failure into a system in order to validate robustness and error handling.

https://docs.microsoft.com/en-us/azure/architecture/framework/resiliency/chaos-engineering

Two main 'use cases' for Chaos engineering

Pre-release validation
"Shift left" (test, stage)

Explore service dependencies in a **controlled environment**

Gate production code flow with **CI/CD pipeline** automation

Perform **incident fix validation**

Harden **release pipeline**

Certify **new hardware**

Perform **BCDR Drills**

Host **Game Days**

Continuous production validation
"Shift right" (canary, production)

Simulate Availability Zone or Region **outages**

Use for **Error Budget testing**

Past incident **regression testing**

Validate on call and **live site processes**

Azure Chaos Studio

Measure, understand, improve, and maintain product resilience

Public Preview

Hosted multitenant service

- ✓ Chaos resource provider
- ✓ Automated and manual chaos experiments
- ✓ REST API + SDKs
- ✓ Azure Portal-integrated UI
- ✓ Orchestrated experiments with parallel and sequential fault actions
- ✓ Expandable fault library
- ✓ Telemetry integration
- ✓ Experiment templates

Current focus

Service Fault Injection—dependency disruption with three ways to inject faults:

- Windows and Linux agent-based faults
- Service-direct (agentless) fault providers
- In-process fault injection: application instrumentation for managed code injection and API interception—working with Microsoft Research

www.aka.ms/AzureChaosStudio

Chaos experiments

Orchestrated multi-step scenarios with faults applied to subscription resource targets while under load

Hypothesis

What is being validated? What are possible outcomes?

Experiment

Orchestrated execution of workload + faults

Run against subscription resource targets.

Analysis

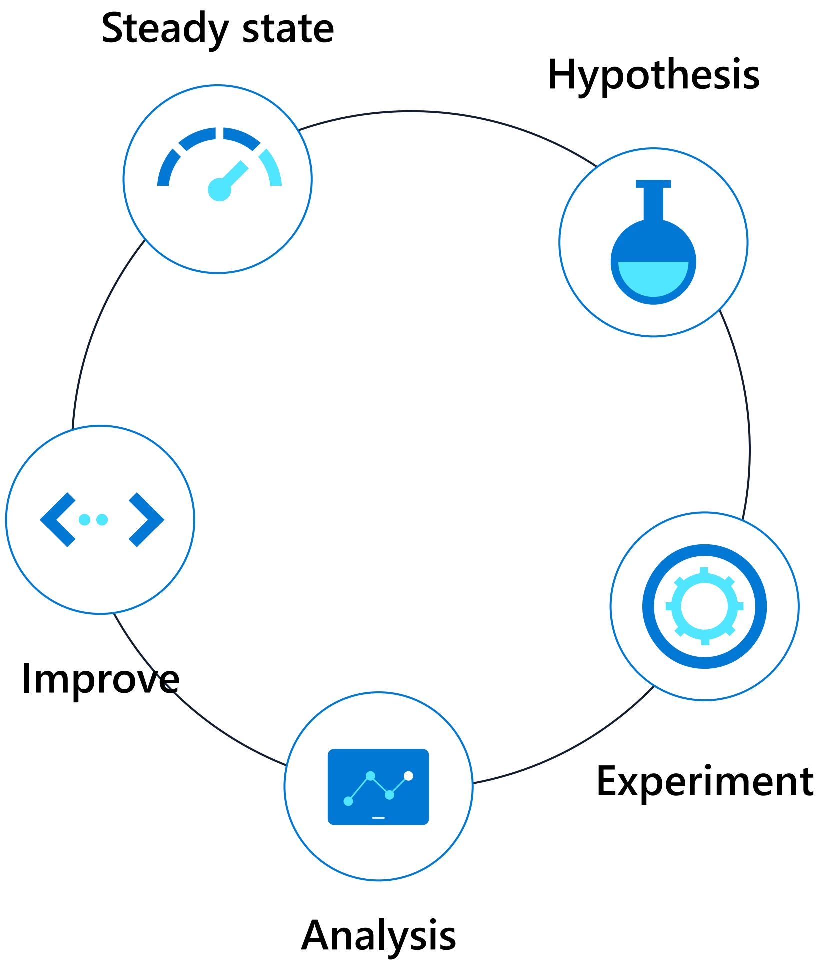
Baseline performance, monitoring telemetry, recovery time.

Improvement

New code, code changes. People and process changes.

Steady state

Continuous production monitoring + validation.



Monitoring for reliability

Monitoring and diagnostics are crucial for resiliency.

If something fails, you need to know that it failed, when it failed—and why.

- ✓ The application is instrumented with semantic logs and metrics
- ✓ All components are monitored and correlated with application telemetry
- ✓ A health model has been defined based on performance, availability, and recovery targets
- ✓ Azure Resource Health events are used to alert on resource health events
- ✓ Application logs are correlated across components
- ✓ Key metrics, thresholds, and indicators are defined and captured
- ✓ Azure Service Health events are used to alert on applicable service level events
- ✓ Monitor long-running workflows for failures

Alerting

Alerts are notifications of system health issues that are found during monitoring. Alerts only deliver value if they are actionable and effectively prioritized by on-call engineers through defined operational procedures…

Outages and service impacting events

When an unplanned outage is happening (and what we know)

- This includes outages, maintenance, service changes, retirements
- When we understand that impact is at the service or platform level
- Communications are typically sent via Azure Service Health
- Post-Incident Review (PIR) provided, once understood

→ Azure Service Health alerts

Issues impacting specific resources

When a resource is down or otherwise unhealthy (but not necessarily why)

- We have detected resource level impact, regardless of whether this is a localized or widespread issue
- Communications typically sent via resource health (within Azure Service Health)
- This data is being augmented into the Service Health experience

→ Azure Resource Health alerts

Customized alerting, logging and monitoring

When customized alerts trigger (based on your configured rules/thresholds)

- This depends on custom SLIs/SLOs as defined by customers and partners
- Azure diagnostic logs and VM logs feed into Azure Log Analytics
- Application metrics and customer metrics feed into Application Insights

→ Azure Monitor alerts

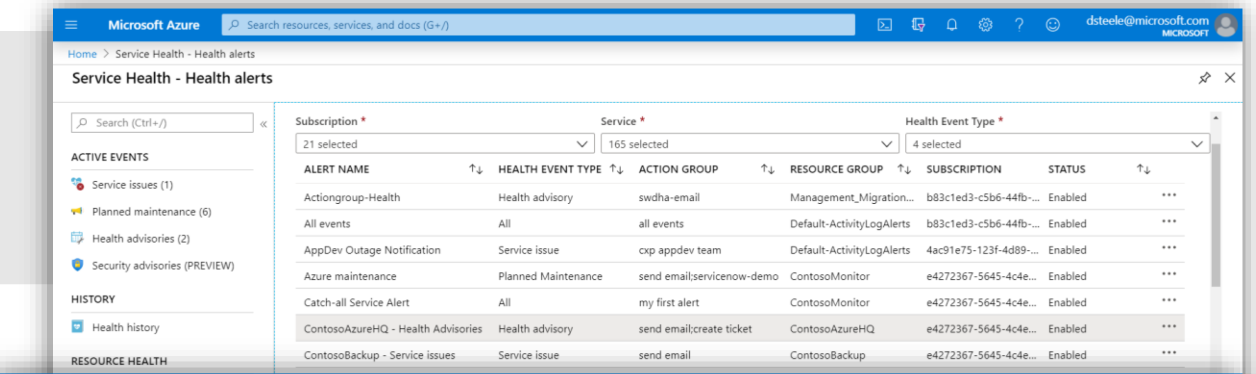
Azure communicates incidents, maintenance, and health advisories via Azure Service Health and Service Health alerts

Azure Service Health alerts are strongly recommended for production systems

Examples of alerts include:

- An alert to email your dev team when a resource in a test/dev subscription is impacted.
- An alert to update ServiceNow via webhook when a resource in production is impacted.
- An alert to send an SMS to a specific number when resources in a given region are impacted.

Note that the public status.azure.com page is only used to communicate issues with widespread impact.



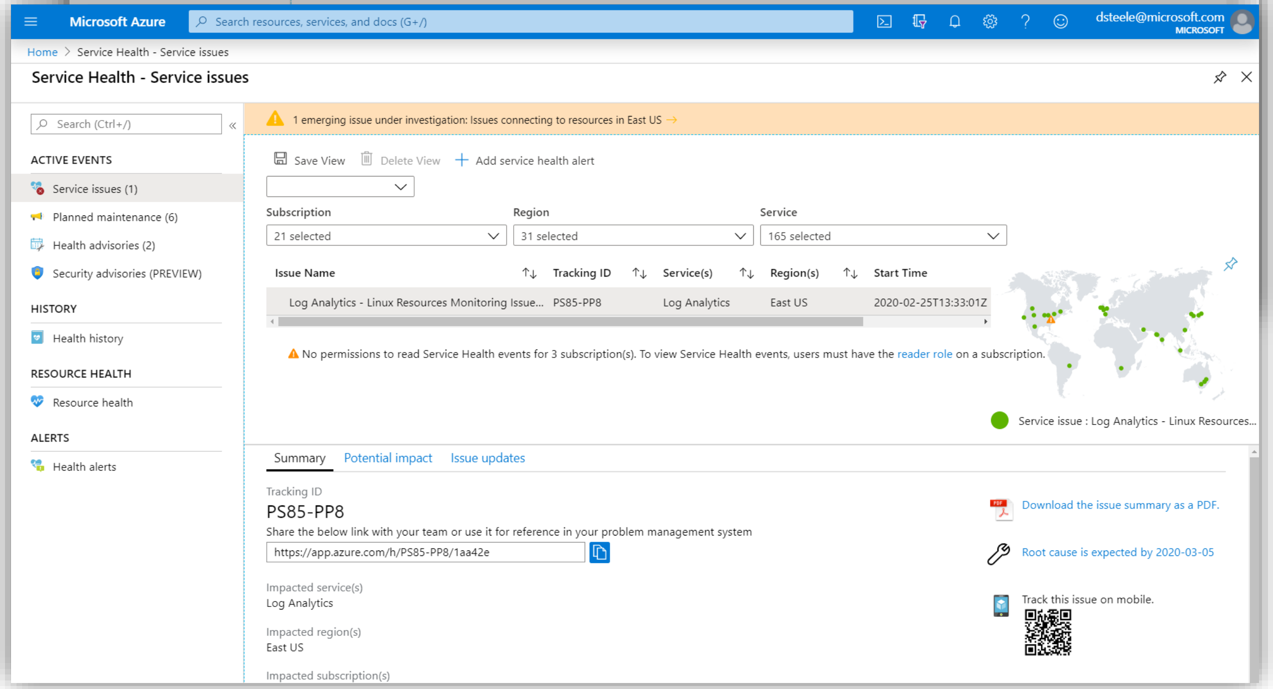
Service Health - Health alerts

Subscription * 21 selected

Service * 165 selected

Health Event Type * 4 selected

ALERT NAME	HEALTH EVENT TYPE	ACTION GROUP	RESOURCE GROUP	SUBSCRIPTION	STATUS
Actiongroup-Health	Health advisory	swdha-email	Management_Migration...	b83c1ed3-c5b6-44fb-...	Enabled
All events	All	all events	Default-ActivityLogAlerts	b83c1ed3-c5b6-44fb-...	Enabled
AppDev Outage Notification	Service issue	cxp appdev team	Default-ActivityLogAlerts	4ac91e75-123f-4d89-...	Enabled
Azure maintenance	Planned Maintenance	send email;servicenow-demo	ContosoMonitor	e4272367-5645-4c4e...	Enabled
Catch-all Service Alert	All	my first alert	ContosoMonitor	e4272367-5645-4c4e...	Enabled
ContosoAzureHQ - Health Advisories	Health advisory	send email;create ticket	ContosoAzureHQ	e4272367-5645-4c4e...	Enabled
ContosoBackup - Service issues	Service issue	send email	ContosoBackup	e4272367-5645-4c4e...	Enabled



Service Health - Service issues

⚠ 1 emerging issue under investigation: Issues connecting to resources in East US →

Subscription 21 selected

Region 31 selected

Service 165 selected

Issue Name	Tracking ID	Service(s)	Region(s)	Start Time
Log Analytics - Linux Resources Monitoring Issue...	PS85-PP8	Log Analytics	East US	2020-02-25T13:33:01Z

⚠ No permissions to read Service Health events for 3 subscription(s). To view Service Health events, users must have the reader role on a subscription.

● Service issue : Log Analytics - Linux Resources...

Tracking ID
PS85-PP8

Share the below link with your team or use it for reference in your problem management system

https://app.azure.com/h/PS85-PP8/1aa42e

Download the issue summary as a PDF.

Root cause is expected by 2020-03-05

Track this issue on mobile.

Impacted service(s)
Log Analytics

Impacted region(s)
East US

Impacted subscription(s)

System notifications of imminent maintenance

Azure Scheduled Events let your VM react to maintenance events before they impact your resources

System notification of upcoming maintenance

- A local endpoint with a simple REST API
- Visibility to upcoming events across different resource types: VMs/Cloud Services/Availability Sets/VMSS
- Includes a 'NotBefore' time (10–15 minutes notification)
- Acknowledge completion to expedite

Potential use cases

- Graceful shutdown—save state, drain node, suspend jobs
- Proactive failover—fasted failover (skip detection)
- Adjust thresholds—avoid failover in the case of VM-preserving maintenance

Covers all maintenance scenarios

- Platform initiated
- In-place low-impact maintenance and Live Migration
- Interactive user calls (e.g., restart a VM)
- New: hardware failure notifications predicted by ML

```
curl -H Metadata:true
http://169.254.169.254/metadata/scheduledevents?
  api-version=2017-08-01


{
    "DocumentIncarnation": {IncarnationID},
    "Events": [
        {
            "EventId": {eventID},
            "EventType": "Reboot" | "Redeploy" | "Freeze",
            "ResourceType": "VirtualMachine",
            "Resources": [{resourceName}],
            "EventStatus": "Scheduled" | "Started",
            "NotBefore": {timeInUTC},
        }
    ]
}
```

Deep dive into key technical domains

Application Design

- Design
- Failure Point and Mode Analysis
- Dependencies

App/Infra Platform

- Service/SKU Configuration
- App State and Config
- Compute Availability

Data Platform

- Service/SKU Configuration
- Consistency
- Replication and Redundancy

Networking and Connectivity

- Network Topology
- Network Component Availability
- Regional and DC Connectivity

Reliability and Recovery

- Recovery Strategy and Design
- Availability Targets
- Recovery Targets

Availability & Scalability

- App Availability
- Data Latency and Throughput
- Data Size/Growth
- Network Throughput and Latency

Monitoring and Measurement

- Health Modelling
- Service and Resource Monitoring
- Application Instrumentation and Monitoring
- Telemetry Pipelines
- Key Metrics and Thresholds
- Alerting and Dashboards

DevOps

- Deployment and Automation
- Environment Builds
- Testing and Validation

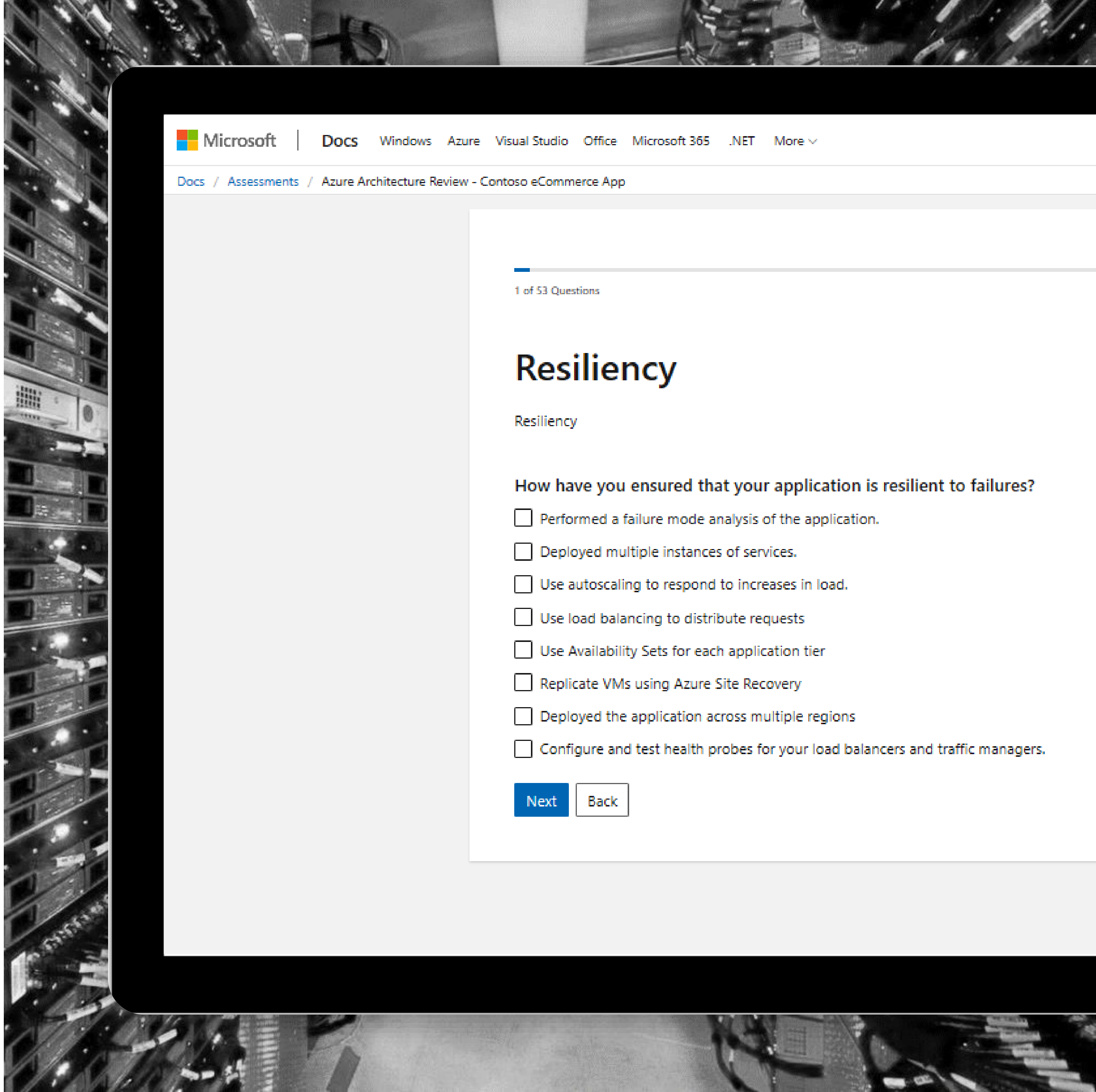
Security

- Identity and Access
- Network Security
- Secrets Management

Microsoft Azure Well-Architected Review

The Azure **Well-Architected Framework** and the associated Azure Architecture Assessment are tools for customers to optimize their workloads across the five pillars—Cost, DevOps, Scalability, Resiliency, and Security.

https://aka.ms/ReliabilityChecklist



Resiliency

Resiliency

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

□ Performed a failure mode analysis of the application.

□ Deployed multiple instances of services.

□ Use autoscaling to respond to increases in load.

□ Use load balancing to distribute requests

□ Use Availability Sets for each application tier

□ Replicate VMs using Azure Site Recovery

□ Deployed the application across multiple regions

□ Configure and test health probes for your load balancers and traffic managers.

Microsoft