Testing Azure applications for resiliency and availability

04/10/2019 • 4 minutes to read • Contributors

In this article

To test resiliency, you should verify how the end-to-end workload performs under intermittent failure conditions.

Run tests in production using both synthetic and real user data. Test and production are rarely identical, so it's important to validate your application in production using a blue-green or canary deployment. This way, you're testing the application under real conditions, so you can be sure that it will function as expected when fully deployed.

As part of your test plan, include:

- Automated predeployment testing
- Fault injection testing
- Peak load testing
- Disaster recovery testing
- Third-party service testing

Testing is an iterative process. Test the application, measure the outcome, analyze and address any failures that result, and repeat the process.

Perform fault injection testing

For fault injection testing, check the resiliency of the system during failures, either by triggering actual failures or by simulating them. Here are some strategies to induce failures:

- Shut down virtual machine (VM) instances.
- Crash processes.
- Expire certificates.
- Change access keys.
- Shut down the DNS service on domain controllers.
- Limit available system resources, such as RAM or number of threads.
- Unmount disks.
- Redeploy a VM.

Your test plan should incorporate possible failure points identified during the design phase, in addition to common failure scenarios:

- Test your application in an environment as close to production as possible.
- Test failures in combination.
- Measure the recovery times, and be sure that your business requirements are met.

- Verify that failures don't cascade and are handled in an isolated way.

For more information about failure scenarios, see Failure and disaster recovery for Azure applications.

Test under peak loads

Load testing is crucial for identifying failures that only happen under load, such as the back-end database being overwhelmed or service throttling. Test for peak load and anticipated increase in peak load, using production data or synthetic data that is as close to production data as possible. Your goal is to see how the application behaves under real-world conditions.

Conduct disaster recovery drills

Start with a good disaster recovery plan, and test it periodically to make sure it works.

For Azure Virtual Machines, you can use Azure Site Recovery to replicate and perform disaster recovery drills without affecting production applications or ongoing replication.

Failover and failback testing

Test failover and failback to verify that your application's dependent services come back up in a synchronized manner during disaster recovery. Changes to systems and operations may affect failover and failback functions, but the impact may not be detected until the main system fails or becomes overloaded. Test failover capabilities *before* they are required to compensate for a live problem. Also be sure that dependent services fail over and fail back in the correct order.

If you are using Azure Site Recovery to replicate VMs, run disaster recovery drills periodically by doing test failovers to validate your replication strategy. A test failover does not affect the ongoing VM replication or your production environment. For more information, see Run a disaster recovery drill to Azure.

Simulation testing

Simulation testing involves creating small, real-life situations. Simulations demonstrate the effectiveness of the solutions in the recovery plan and highlight any issues that weren't adequately addressed.

As you perform simulation testing, follow best practices:

- Conduct simulations in a manner that doesn't disrupt actual business but feels like a real situation.
- Make sure that simulated scenarios are completely controllable. If the recovery plan seems to be failing, you can restore the situation back to normal without causing damage.
- Inform management about when and how the simulation exercises will be conducted. Your plan should detail the time frame and the resources affected during the simulation.

Test health probes for load balancers and traffic managers

Configure and test health probes for your load balancers and traffic managers. Ensure that your health endpoint checks the critical parts of the system and responds appropriately.

- For Azure Traffic Manager, the health probe determines whether to fail over to another region. Your health endpoint should check any critical dependencies that are deployed within the same region.
- For Azure Load Balancer, the health probe determines whether to remove a VM from rotation. The health endpoint should report the health of the VM. Don't include other tiers or external services. Otherwise, a failure that occurs outside the VM will cause the load balancer to remove the VM from rotation.

For guidance on implementing health monitoring in your application, see Health Endpoint Monitoring pattern.

Test monitoring systems

Include monitoring systems in your test plan. Automated failover and failback systems depend on the correct functioning of monitoring and instrumentation. Dashboards to visualize system health and operator alerts also depend on having accurate monitoring and instrumentation. If these elements fail, miss critical information, or report inaccurate data, an operator might not realize that the system is unhealthy or failing.

Include third-party services in test scenarios

If your application has dependencies on third-party services, identify where and how these services can fail and what effect those failures will have on your application. Keep in mind the service-level agreement (SLA) for the third-party service and the effect it might have on your disaster recovery plan.

A third-party service might not provide monitoring and diagnostics capabilities, so it's important to log your invocations of them and to correlate them with your application's health and diagnostic logging using a unique identifier. For more information on proven practices for monitoring and diagnostics, see Monitoring and diagnostics guidance.

Next steps

Deploy for resiliency and availability