Planning for Failure to Ensure Resilience

Even prior to the COVID-19 pandemic, the technology world was in the midst of a fundamental shift in how we operate. DevOps and cloud computing adoption growth was the vanguard of this change. With the pandemic, we identified where some of our gaps were to enabling people to work remotely, which would be the case in a real-life business continuity/disaster recovery situation. These gaps highlighted lessons we already knew:

- The Eight Fallacies of Distributed Computing[1] are still fallacies.
- Disconnected teams (teams in silos or stovepipes) tend to suffer with inflexibility and a lack of agility and velocity.
- Tribal knowledge[2] is a killer when a key resource leaves the organization or is otherwise unavailable.
- Tribal knowledge also means a new resource takes longer than necessary to onboard.
- Failure will happen. A failure to plan for it is a plan to fail.

Particular approaches help address these lessons, and while some of these approaches are not exactly innovative, they are necessary if the goal is to achieve real-world resilience in organizations. It is helpful to focus on three particular approaches:

1. Planning for failure
2. Documented common languages/interfaces/models
3. Infrastructure as Code (IaC)

Planning for Failure

Brainstorming is a long-used, useful technique to try to get our minds around a situation or problem. When I think about planning for failure, I like to brainstorm. Whether I am looking at strategic problems for the organization to solve or the detailed specifics for a system, I encourage brainstorming. Why? When everyone is throwing out possibilities that are simply recorded, sometimes we discover an issue we would otherwise overlook. I want to capture those issues. Therefore, if I were to start a brainstorming session about failure at a strategic level, here are some questions I would pose:

- What do we do if our organization unexpectedly cannot go into work?
- What do we do if an unexpected number of people in the organization are unavailable?
- What if a small but critical group is suddenly unavailable?
- What do we do if key technical resources are unavailable because our people generally are not available?
- What do we do if key third-party technical resources are unavailable because of overloading?
- What do we do if key technical resources are unavailable because they are under attack?

Why did I choose these questions? Is it because of the pandemic? Not specifically, because these are all standard business continuity/disaster recovery questions that we ask regardless of the cause. For instance, if an organization is located in an area prone to severe weather (e.g., tornadoes, hurricanes, typhoons, cyclones, blizzards), the likelihood that some personnel will be unavailable is high. Therefore, the questions around people are critical. If a weather incident affects a large region, then the overloading of third-party resources (e.g., cellular systems) is a strong probability as well.

To be resilient, we have to ask and answer these questions at all levels, from the most strategic to

K. BRIAN KELLEY | CISA, CDPSE, CSPO, MCSE, SECURITY+

Is an author and columnist focusing primarily on Microsoft SQL Server and Windows security. He currently serves as a data architect and an independent infrastructure/security architect concentrating on Active Directory, SQL Server and Windows Server. He has served in a myriad of other positions, including senior database administrator, data warehouse architect, web developer, incident response team lead and project manager. Kelley has spoken at 24 Hours of PASS, IT/Dev Connections, SQLConnections, the TechnoSecurity and Forensics Investigation Conference, the IT GRC Forum, SyntaxCon, and at various SQL Saturdays, Code Camps and user groups.

the highly tactical. Also, we must test our answers. This is not easy because it is time and resource consuming. However, if we do not do the planning and the subsequent testing, we have planned for our organizations to fail.

I Am on the Cloud. Am I Safe?

Cloud computing can address some concerns with regard to regional issues, but simply being on the cloud is not enough. Cloud computing provides a compelling set of services and offerings to most organizations. However, as with any other technical resource, proper planning is key. After all, every major cloud vendor has suffered outages to key platforms over the last few years. Also, assuming that I will be able to reach my cloud provider when needed falls into that distributed computing fallacy about the network always being available. Someone trying to work a key business process over a tethered smartphone when cellular service is overloaded may not be able to reach cloud resources.

I Am on Multiple Clouds. Am I Safe Now?

If an organization is on multiple clouds, for example, Azure and AWS or AWS and Oracle, then it is protecting against an outage for a single cloud. However, if the network is unavailable to get to cloud resources, it does not matter how many cloud providers the organization uses. From an audit perspective, this is an important business continuity issue to remember. What can I do if I lose access to Internet-based resources, to include cloud providers?

"When we ask ourselves how failure could occur, we have the ability to do the risk assessment necessary so that we can properly prioritize our efforts."

A Systematic Approach

When we ask ourselves how failure could occur, we have the ability to do the risk assessment necessary so that we can properly prioritize our efforts and be able to determine whether the remediation costs more than the asset or process we are trying to protect. That is the start of working through the potential issues in a logical manner.

However, we also should be looking at the potential problems and looking to see how we can simplify things. We also want to determine where we can accelerate restoration of services. That leads us to documented Common Languages/interfaces/models and IaC. Both help to reduce the complexity of the overall configuration. As a result of this, we should have a better understanding of our inventory and how each part of it is implemented. So let us look at these two in turn.

Common Languages/Interfaces/Models

A common problem in development where you have a primary system and a second one is that we must understand how each system works, for example, where you are trying to interface two separate inventory management systems. Likely, the data structures and the application programming interfaces (APIs) for each system are different. If I am an expert on one system, I still have to take the time to learn how the other system works to be able to connect the two. If both are important to my organization and they need to talk to each other, then, naturally, the organization is going to authorize me to do so. However, is this necessary?

What if, instead, each platform's team built a translation layer to a common model? For instance, an inventory management system should be able to add and remove items from the inventory. If there was a standard API definition applied on top of both applications, I would not have to learn how the other system worked. I could simply make the calls that I should already be familiar with because my system implements the exact same interface.

Having the same model means I am also speaking the same language when I do have to talk to the other team to try to solve a difficult problem; so when I say "product," there is no misunderstanding what I mean. The more people who are familiar with the model means a reduced likelihood of a severe impact if a particular person is missing.

"Having the same model means I am also speaking the same language when I do have to talk to the other team to try to solve a difficult problem."

Another reason for the common interface is I can bring a common set of tests to apply regardless of the underlying system. I do not have to build a separate set of tests per application, at least with regard to the API. Likewise, this helps with troubleshooting because it means that I can run a common set of tests and use the same tools to validate whether or not the interface is working as expected. That also assists with monitoring because, again, commonality means I can scale easier.

Finally, if someone new comes into the organization, they will have to learn their system and the basic models that are implemented, but that is significantly less to learn than if they have to educate themselves on details about each interfacing system. Likewise, if I need personnel to move to a different team, temporarily or permanently, they have some working knowledge because of that common interface.

Deciding on common models and common interfaces also means I should have appropriate documentation. Having documentation means I am converting tribal knowledge to internal standards and documented processes. This should increase consistency and quality.

IaC

IaC is a technology being used increasingly across industries to free people to do more valuable work, increase consistency in configurations and reduce human error—human error that could derail resilience efforts.

Code is not limited to development. Nowadays, there are tools to build out the entire infrastructure from code: storage, networks, servers, key services. This serves as both documentation and provides recoverability, so long as the code is properly protected. Also, it allows practitioners to quickly scale up or scale down resources as needed. More important, it allows us to fully automate deployments of key resources.

Thinking about multicloud scenarios, there are tools that allow practitioners to deploy the same pieces of infrastructure across different cloud providers. Even if we do not have those tools at our organization, if we know the specifications we are building toward, teams can develop code to deploy like resources across providers using only cloud provider-specific scripts and configurations to do so. This all adds up to increase our overall resiliency.

Also, if we are building out using IaC, then we should have common patterns and models, just as we looked at earlier. We gain the same advantages. We reduce the risk of key resources not being available. After all, if we have the web server builds automated, I do not have to have a particular admin who assisted with the last manual installation available. I just rerun the code with the configuration to create a new server.

There Is So Much More

These three approaches help with real-world resilience. There are tools available to help manage the computers of a remote workforce. Vendors provide secure smart device management, which allows an organization to centrally administer key policies and security configuration and push down necessary applications, virtual private network (VPN) connections and more, to allow workers to be as connected to the enterprise resources as they need to be. Some cloud providers have service offerings with backups of virtual machines in their environment so that, in a real disaster, you can spin up your resources there. However, if I were to go back to any particular approach to best ensure resilience, I would go back to the one that is not an emerging technology: plan for failure. Without this approach, the others might, well, fail.

LOOKING FOR MORE?

- Read *Optimizing Risk Response.* *www.isaca.org/optimizing-risk-response*
- Learn more about, discuss and collaborate on risk management in ISACA's Online Forums. *https://engage.isaca.org/onlineforums*

Endnotes

1 Singh, E.; "The Eight Fallacies of Distributed Computing," *Medium*, 29 September 2021, *https://medium.com/geekculture/the-eight-fallacies-of-distributed-computing-44d766345ddb*
2 Lucid Content Team, "Tribal Knowledge: Risks, Benefits, and What to Do About It," LucidChart, *https://www.lucidchart.com/blog/what-is-tribal-knowledge*