opentext™ | Cybersecurity

Position Paper

Moving NetIQ Access Manager to a Cloud Infrastructure

Table of Contents

Making the Transition Smooth

This paper describes the process and provides guidance on the factors you need to consider when migrating NetIQ Access Manager to a cloud-based deployment model.

NetIQ Access Manager Architecture

There are many differences between a traditional on-premises NetIQ Access Manager by OpenText™ deployment and a cloud-based deployment. Before we contrast the two, we need to understand the functional components of NetIQ Access Manager. NetIQ Access Manager has four high-level functional components: the Administration Console, the Analytics Server, the Identity Server, and the Access Gateway.

The Administration Console is used to manage, store, and distribute the configuration of the system. Multiple Administration Consoles share a replicated data store, making the system inherently fault tolerant and eliminating single points of failure.

The Analytics Server captures audit and operational data. It provides a configurable dashboard and powerful reporting capabilities. The visibility into the security, utilization, and performance of your access management system comes from the Analytics Server.

The Identity Server authenticates access to web workloads integrated with NetIQ Access Manager. It is also commonly called an identity provider (IDP) because it authenticates users using federation protocols. It supports all common federation protocols and can broker federation connections. The authentication framework utilizes a plug-in architecture. So adding custom authentication methods is easy if one of the out-of-the-box options doesn't meet your needs. While the Identity Server requires an Administration Console, it does not require the Access Gateway.

The Access Gateway service is a reverse-proxy whose primary purpose is to integrate the Identity Server with legacy applications that can't integrate directly. If the application, or application platform, can use protocols such a SAML2, OAuth, or WS-Federation, then the Access Gateway is optional. However, there are several reasons to include it. The Access Gateway provides an additional layer of security. It likely provides a smaller attack surface than your application servers, reducing possible vulnerabilities. Patching and vulnerability mitigation are simplified because you don't need to consider the dependencies of your application. The reverse proxy can cache web content, enhancing system performance.

It can also route traffic based on the requested path so that you can deliver multiple back-end applications and services as a single unified application. Finally, the Access Gateway provides centralized, policy-based access control that can replace or enhance the level of security your application offers.

While a separate product, NetIQ Advanced Authentication by OpenText is often deployed with NetIQ Access Manager to provide comprehensive multi-factor authentication capabilities. Advanced Authentication integrates seamlessly with on-premises or cloud-based NetIQ Access Manager deployments.

NetIQ Advanced Authentication integrates seamlessly with on-premises or cloud-based NetIQ Access Manager deployments.

NetIQ Access Manager version 4.x has supported two deployment models. The first is a soft-appliance model where all services that make up the product are deployed together on an appliance node. The second model is to deploy the services individually on separate nodes, running either a Linux or Windows-based operating system. NetIQ Access Manager version 5.0 provides a new Docker container-based deployment model that is ideal for cloud deployments and replaces the appliance deployment model. A container-based model is even more flexible than the appliance model. The required containers can be deployed together on a single operating system node if desired. Then the number of containers and nodes can be scaled dynamically without the limitations imposed by the previous appliance model. In the rest of this paper, we will refer to the non-container model as the “legacy” model. A typical legacy model deployment is shown in the diagram below.

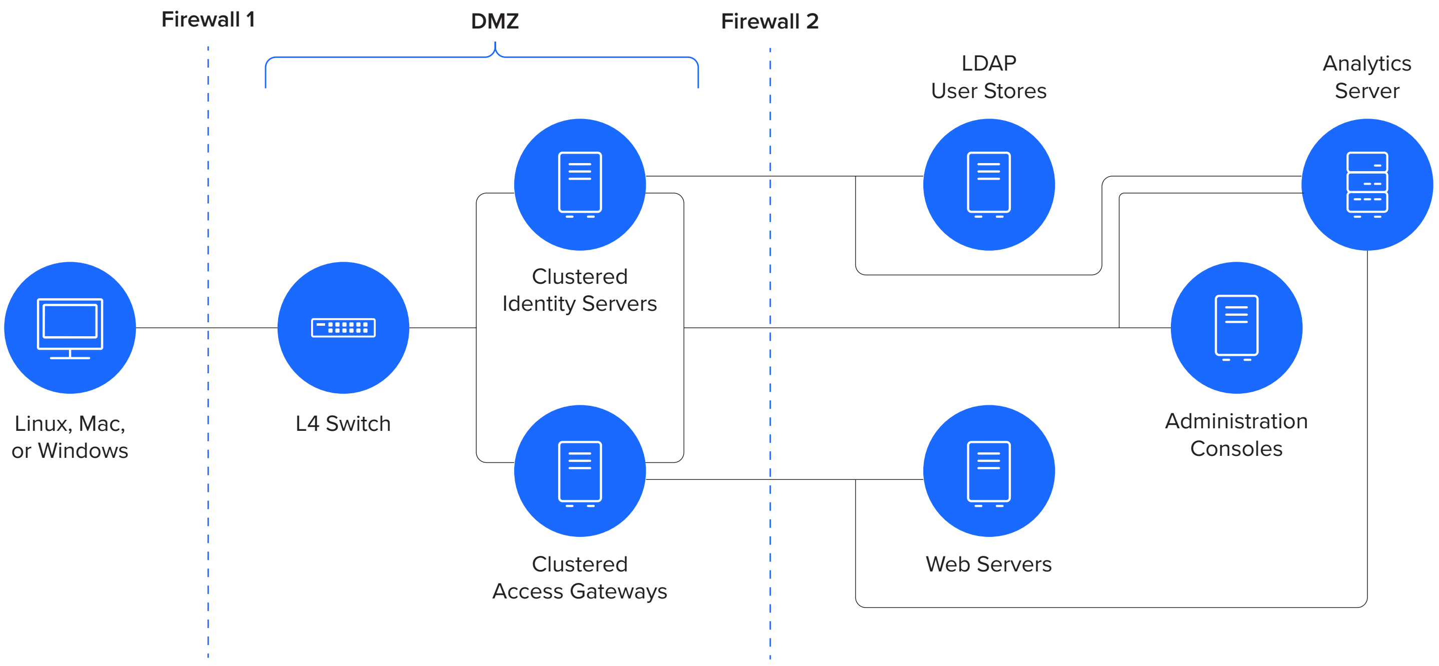

Figure 1. NetIQ Access Manager legacy model

In this model, there are multiple servers—physical or virtual—running redundant instances of NetIQ Access Managers' core services. You can only run a single instance of any of the services per node. It is possible to run the Administration Console service and the Identity Server service on the same node, but this is not commonly done in a production environment due to security concerns.

It is possible to use the legacy model with virtual machines (instances) running on a cloud infrastructure. There are some small details that differ from an on-premises deployment, but at a high level, it is the same architecture. It is also possible to add a level of auto-scaling of the infrastructure when deployed in this manner. This option might be attractive for organizations that are not ready to make the transition to containers, but it is not the best option for cloud-based deployment. You give up a lot of flexibility. And it's likely that this model will be more expensive because you need to dedicate virtual-machine instances with sufficient resources.

A best option for cloud deployment is to use a cloud-native architecture based on Docker containers and Kubernetes orchestration.

A better option for cloud deployment uses a modern cloud-native architecture based on Docker containers, Kubernetes orchestration, and Helm configuration management. There are many benefits to these technologies. But utilizing them might be a big change for organizations with limited cloud experience. However, these are the technologies behind all the major cloud-native services, and they are rapidly being adopted at the enterprise level. It makes sense to pursue this deployment model unless you're convinced that your organization is not ready to support it.

Let's dive into these technologies and discover their benefits.

Introduction to Using Container Technologies for NetIQ Access Manager

Docker Containers

What is a container? A container is an encapsulated runtime environment containing an application and all its dependencies. Containers allow you to manage your application independent of the operating system and infrastructure. This is a very powerful concept. Isolating support of the application from the underlying infrastructure allows you to manage both more effectively. Software release and upgrade cycles are shortened. System resources can be managed more efficiently. Applications become much more portable. Best of all, OpenText™ engineers fully control and support everything inside the NetIQ Access Manager container. To upgrade the application, you simply deploy a new version of the container.

Docker is an open-source project that provides the tools and framework needed to create and use containers. It is the de-facto industry standard that all major cloud vendors support. Using Docker allows the exact same application configuration to run on a developer's workstation and on a public cloud platform. In fact, the application can be moved from one to the other in minutes.

The diagram below illustrates the architectural differences between traditional virtualization and containers. Containers can be smaller, more portable, and require fewer resources. Dependencies on a specific operating system can be eliminated, both in terms of the host system and the guest operating systems. Containers might still contain all or part of an operating system, but the container developer manages that code. There is no need for operations personnel to maintain the operating system. It is completely transparent to them.

Kubernetes provides the framework for managing a full system consisting of many applications, services, and host nodes.

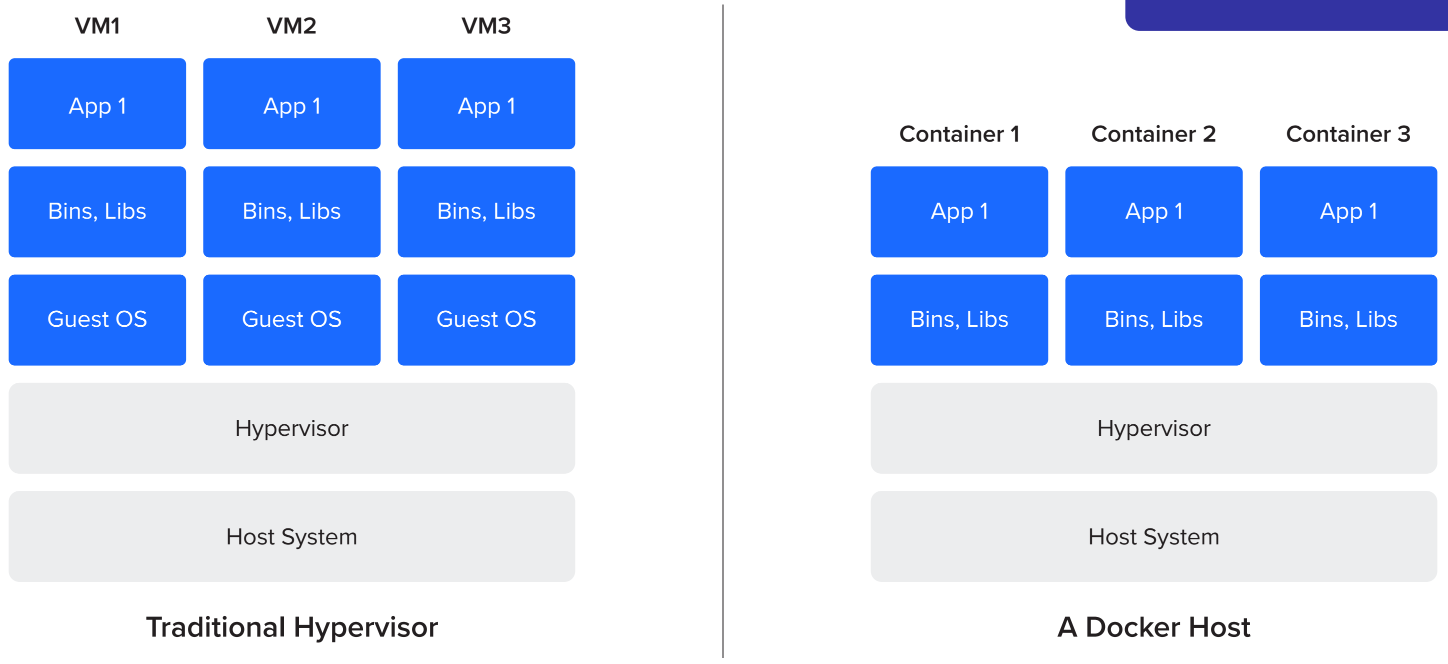

Figure 2. Traditional virtualization versus containers

Docker makes deploying and updating applications incredibly easy. A single command can deploy an entire application. A single command can also update an application to the latest version. If there are any issues with the update, a single command can roll the configuration back to the previous version.

Kubernetes Container Orchestration

A container-based NetIQ Access Manager deployment consists of multiple services in separate containers. As you scale the system, the number of containers increases. Along with managing the containers themselves, there is a need to manage the networking, load balancing, scaling, and other aspects of the system. This is where Kubernetes comes into play. Docker provides the tools to effectively manage a container-based application on a host node, while Kubernetes provides the framework for managing a full system consisting of many applications, services, and host nodes.

Kubernetes organizes logical groupings of containers into sets that are referred to as "pods." Each pod is managed as a unit that can be deployed to a host node based on the availability of resources. Kubernetes can monitor the status of the pod and restart or move the pod as needed.

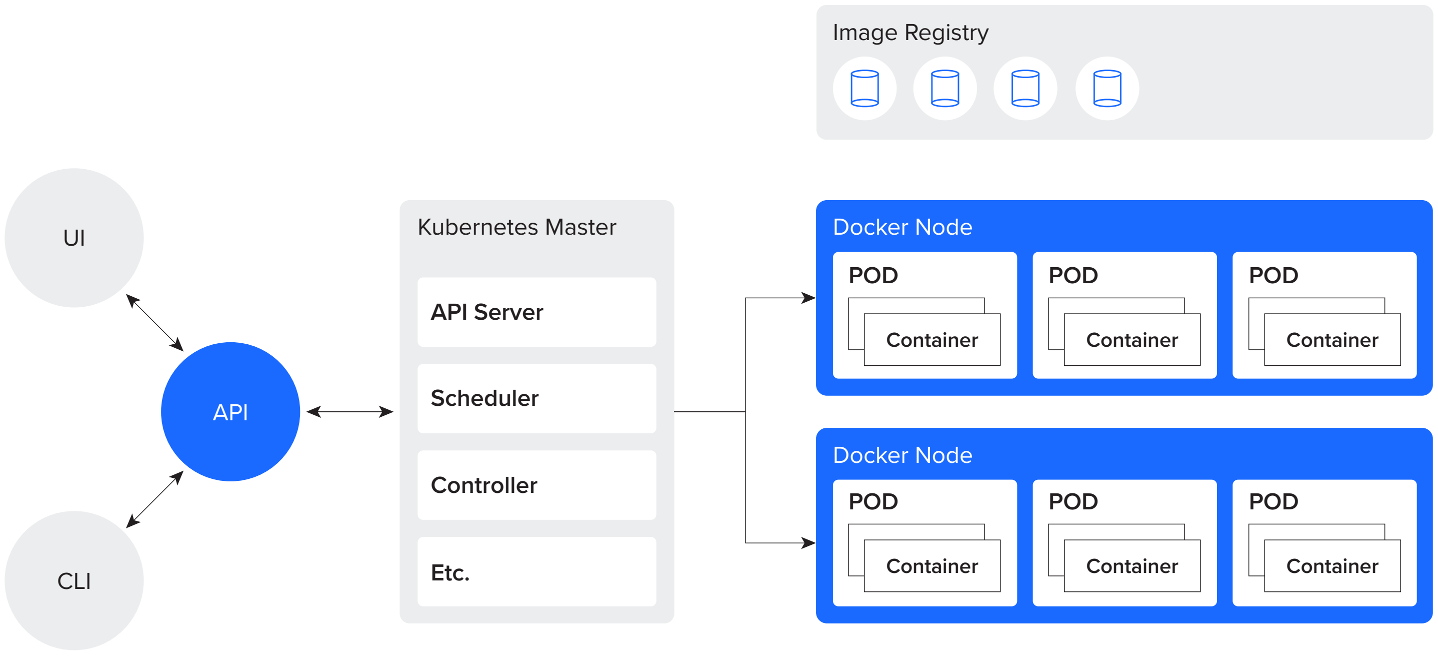

Figure 3. Kubernetes container orchestration

Helm Configuration Management for Container-Based Systems

Helm is an open-source project that provides a convenient way to describe complex Kubernetes systems in a repeatable and manageable way. You can think of it as a package manager for Kubernetes. The configuration is described in a document called a "chart." Using a Helm chart, you can deploy a full NetIQ Access Manager system to a Kubernetes cluster in just a few minutes. Updates are also quick and painless. Helm supports rolling back to the previous configuration as well.

Helm is an open-source project that provides a convenient way to describe complex Kubernetes systems in a repeatable and manageable way.

The system deployment process is shown below. The container images are maintained by OpenText and made available through an image registry. NetIQ Access Manager provides a Helm chart template that you customize to meet your specific requirements. This chart defines how many containers are deployed, where they are deployed, network configuration, monitoring, fault tolerance, persistent storage, etc. The Helm chart is applied to Kubernetes to deploy NetIQ Access Manager. Kubernetes downloads the specified version of the container images from the repository and creates containers based on the images. The containers execute initialization scripts that complete the initial setup of NetIQ Access Manager. This process takes only minutes to complete.

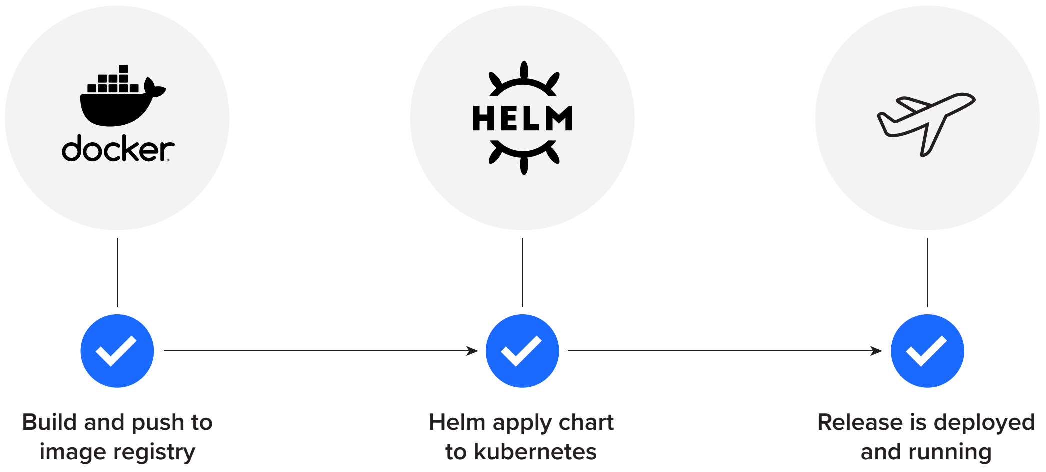
docker
HELM
Build and push to image registry
Helm apply chart to kubernetes
Release is deployed and running

Figure 4. Container-based deployment

Migrating a complex NetIQ Access Manager system can be a daunting task but good planning will pay you back tenfold in simplifying the process.

Updating to the latest version of NetIQ Access Manager is equally simple. You update the Helm chart to reference the desired version and apply the updated chart. This process also takes only minutes to complete.

All this cutting-edge technology might sound complex and expensive. Fortunately, you don't have to build and maintain it yourself. Full-featured Kubernetes services are available from all the major public cloud providers. All the private-cloud vendors also provide support for Kubernetes. Amazon Elastic Kubernetes Service (EKS) is a managed Kubernetes service that makes it easy to deploy Kubernetes based systems to AWS without the need to install and operate your own Kubernetes clusters. Microsoft provides a similar service with Azure Kubernetes Services (AKS). Deployment of NetIQ Access Manager is documented, supported, and certified on both systems.

Migrating NetIQ Access Manager

Migrating a complex NetIQ Access Manager system can be a daunting task. Good planning will simplify the process. For a successful migration, you must:

1. Assess the current implementation.
2. Identify future-state requirements.
3. Select a cloud deployment platform.
4. Determine the migration approach.
5. Identify an operational model and provide any needed training.
6. Design the implementation.

7. Deploy a test environment.
8. Test for function and performance.
9. Deploy production system.
10. Test and validate the production system.
11. Migrate applications and functions.

Assess the Current NetIQ Access Manager implementation

Understanding what your current system is doing and how it is configured is the first step. In an ideal world, most of the information you need would already be available in your normal operational documentation. However, this is rarely the case; few organizations do a good job of maintaining the needed documentation. Even with proper documentation, access management integrations are often completed as a project. Once the project is over, the team disbands. There might not be anyone who is responsible for or capable of modifying and testing the integration.

You will need to:

- Discover which NetIQ Access Manager components are in use and which are not.
- Know if federation to external identity providers is being used.
  - Which protocols?
  - Who in your organization "owns" the relationship with each identity provider?
  - What is the testing process for each provider?
  - Who has the ability to make changes to the identity provider configuration, and what is the process to implement needed changes?
- Create a catalog of federated service providers.
  - Who has access to modify the service provider?
  - Is contract support required to implement any configuration changes?
  - What is the process needed to implement changes, and what is the lead time required?
  - Who in your organization is responsible for the application? Who is the "business owner"?
  - Who has the capability and responsibility to test the integration? Is there a documented testing process?

Identify Future-State Requirements

After you have a good understanding of your current system, the next activity is to identify any needed or upcoming changes to system functionality. You need to look at not only what new integrations and functionality might be needed but also at what integrations and functionality can be retired. An example of this is legacy integrations based on form-fill or header injection where federation might now be possible. There might also be integrations using older federation protocols that now support modern protocols.

Questions to answer include:

- What federation protocols will be needed going forward? It is likely that some new integrations might require OAuth and OpenID Connect.
- Will the Access Gateway proxy be required? While using the proxy brings additional security and capabilities, moving to a fully federated model simplifies operations.
- Are new applications coming?
- Are there any significant changes expected to system load?

You need to look at not only what new integrations and functionality might be needed but also at what integrations and functionality can be retired.

Select Cloud Deployment Platform(s)

The first step in determining your cloud deployment platform is to decide on either a legacy or container-based model. OpenText recommends using containers if at all possible. If you choose to use a legacy model, then both Azure and AWS are certified platforms with a documented process for virtual machine deployments.

If you choose to go with containers, then you have the following options:

- Amazon Elastic Kubernetes Services (EKS)
- Azure Kubernetes Services (AKS)
- Another Kubernetes platform (including creating your own cluster infrastructure on cloud-based virtual machines)

Both EKS and AKS are fully supported and certified by OpenText. You can also use other Kubernetes infrastructures, and OpenText will provide best-effort support for the orchestration.

Identify Operational Model and Provide Training

Operating an Access Management system can be very challenging. This is especially true when you're dealing with applications that other groups within—or outside of—your organization own and manage. Moving to a cloud-based deployment brings new technologies into the mix. These challenges need to be addressed as part of your migration planning. You need to identify the people within your organization that will support the new system and provide them with knowledge and training. OpenText training and professional services can help with both standard and customized training programs.

Select a Migration Approach

In this section, we will examine the options for migrating from an existing access management system to a new one. The concepts presented here apply to the migration of any access management system to another. But the primary focus is on moving from an existing NetIQ Access Manager system to a new cloud-based NetIQ Access Manager system. There is an assumption that user data and credentials are maintained either by continuing to use the existing data store or by synchronizing the existing data store with the new data store. If this is not possible, contact OpenText™ Professional Services for assistance.

Below is a conceptual diagram of an NetIQ Access Manager system, showing the elements that you need to consider when planning the migration.

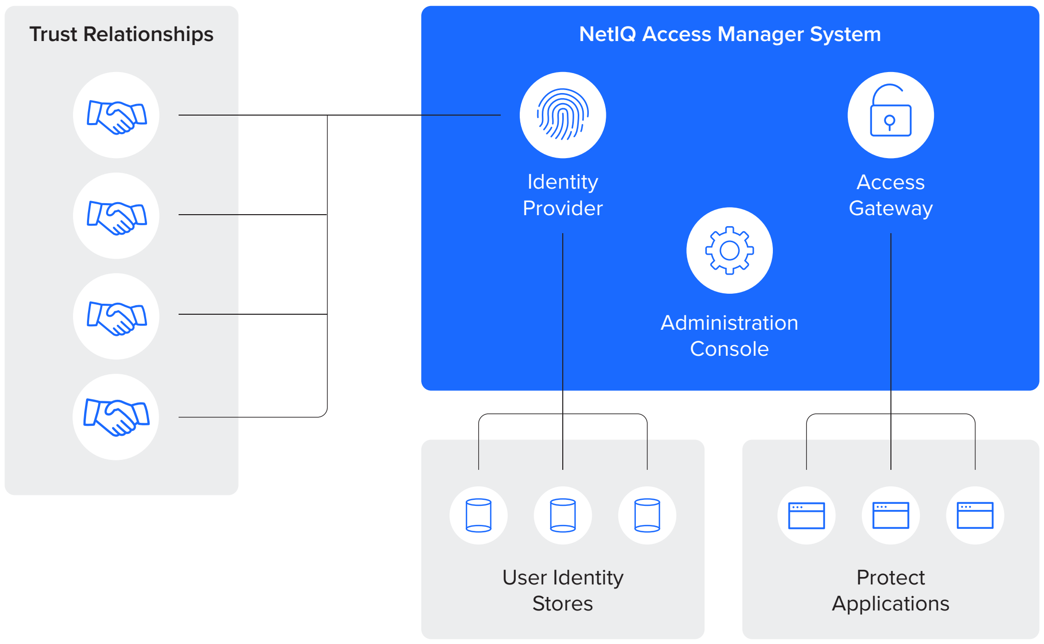

Figure 5. Elements to consider when planning your migration

The primary issue when migrating from one access management system to another is maintaining any existing trust relationships through the migration. There are systems with hundreds, even thousands, of trust relationships. Such scale makes an abrupt cutover impossible. Even a reasonably small number of relationships can impact migration because modification and testing generally require coordination with external parties or other parts of the organization.

Any changes to the "identity" of the NetIQ Access Manager system affects these trust relationships. Exactly how the system's identity is linked to the trust relationship varies among the different federation protocols. For SAML, the key factors are the SAML entity ID, the service endpoint URLs, and the certificates being used for signing or encryption. A change to any of these requires modifications to the configurations of all the service providers that have a trust relationship with the identity provider.

Your first thought might be that it would be best to avoid any issues by not changing anything that impacts the system identity. This is a viable approach but is sometimes not possible because of infrastructure differences, legal limitations, or other factors. Even if your situation allows you to maintain the existing system identity, how do you test and verify that everything still works as it should in a large complex system? Abrupt cutover, even when the system identity is maintained, is rarely an acceptable option.

Migrating any applications protected by the Access Gateways is also a consideration. The number of applications and the complexity of verifying that each continues to function properly will determine if a quick cutover is possible or if a more methodical approach must be taken.

Another factor when determining a migration approach is risk tolerance and acceptable downtime. A situation where some downtime is acceptable is very different from one where no downtime is acceptable. Each approach differs in the options available to manage risk. Some allow you to break the migration process into smaller activities where the scope and impact of each activity are reduced.

Choose one of these migration approaches that best fits your needs:

- Full cutover to a new system that maintains current entity identity
- Full cutover to a distinct new system
- Non-integrated phased migration
- Spanned cluster migration

The options available for migration approach are:

1. Full cutover to a new system that maintains current entity identity.
2. Full cutover to a distinct new system.
3. Non-integrated phased migration (no single sign-on [SSO] across systems during migration).
4. Integrated phased migration (SSO across systems during migration).
5. Spanned-cluster migration (maintains current entity identity).

APPROACH 1: FULL CUTOVER TO A NEW SYSTEM THAT MAINTAINS CURRENT ENTITY IDENTITY

This approach results in the identity provider maintaining its current entity identity. It avoids any need to update the configuration of trusted partners. You will still need to coordinate testing of each trust relationship, but the risk of issues occurring is very low. Applications protected by the Access Gateways can be fully tested before migration by using host files or alternate DNS servers. Once testing is complete, change DNS to point to the new service. This common approach works well if all integrations can be tested ahead of cutover and if the scale of system validation is manageable. The problem with this approach is that it is all or nothing. An issue with any one integration requires you to roll back the entire migration. Fortunately, you only need to direct traffic back to the old system to roll back.

This option allows you to avoid upgrading the existing system. It can be left in its current state, which should reduce the total level of effort needed. This approach might also be a good option when the existing system is not NetIQ Access Manager yet is highly compliant with federation standards.

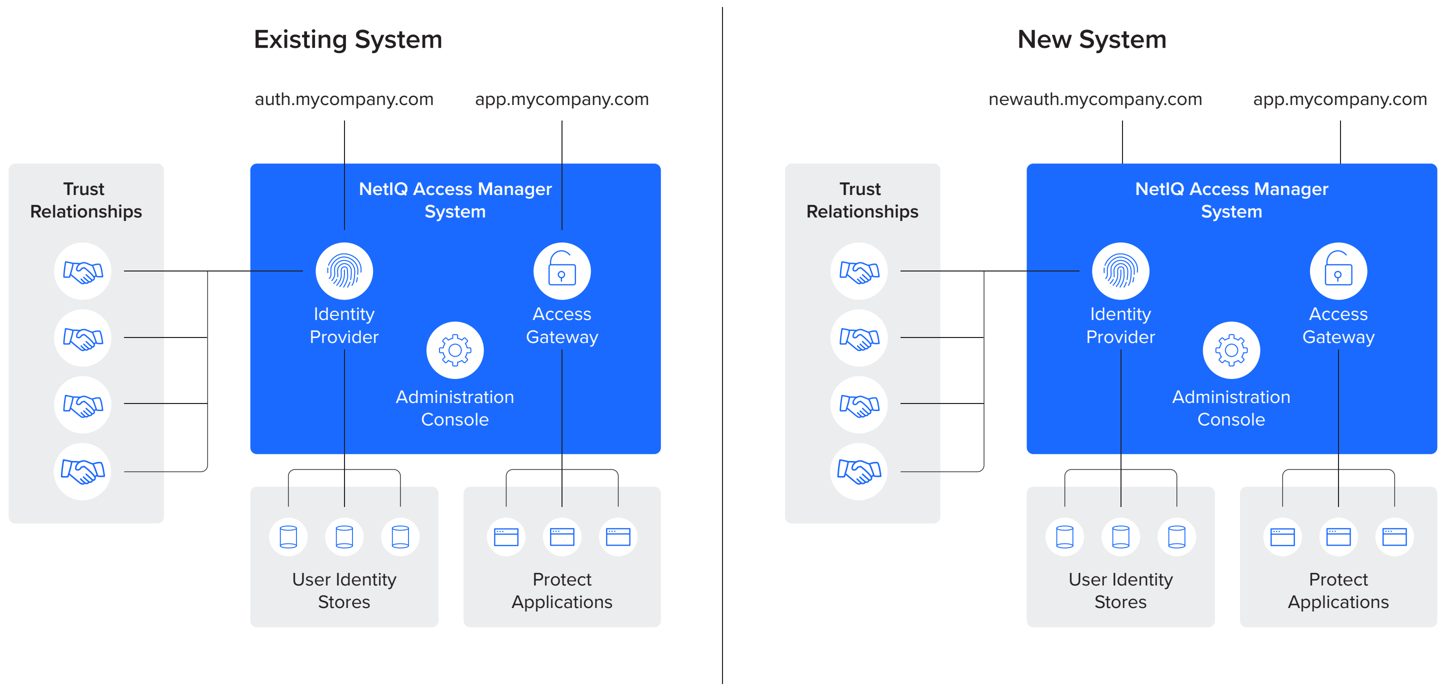

Figure 6. Approach 1—Cutover to new system that maintains identity entity

Implementation process:

- Set up the new system and configure it with the same identity provider DNS name.
- Configure reverse proxies on the new Access Gateways that replicate those on the old system.
- Test the new system as fully as possible using host file entries.
- Cutover all federation partners and Access Gateway clusters at once. In this case, there is no need to modify any configuration of trusted partners.
- Test all applications and federations.

APPROACH 2: FULL CUTOVER TO A DISTINCT NEW SYSTEM

This approach results in the identity provider being a completely new entity. It's only practical when federation is not in use or when you have full control to modify the configuration of all federation partners. The coordination required to update multiple federations with third-party providers within a single change window is almost impossible. Adequate testing of federated apps before cutover can be difficult, and rolling back in case of issues during the cutover can be problematic. Still, this approach is a viable option for systems supporting few applications where the organization has the requisite span of control. You also avoid upgrading the existing system. This approach is included here for completeness and because it is the basis for more manageable approaches, which are described later.

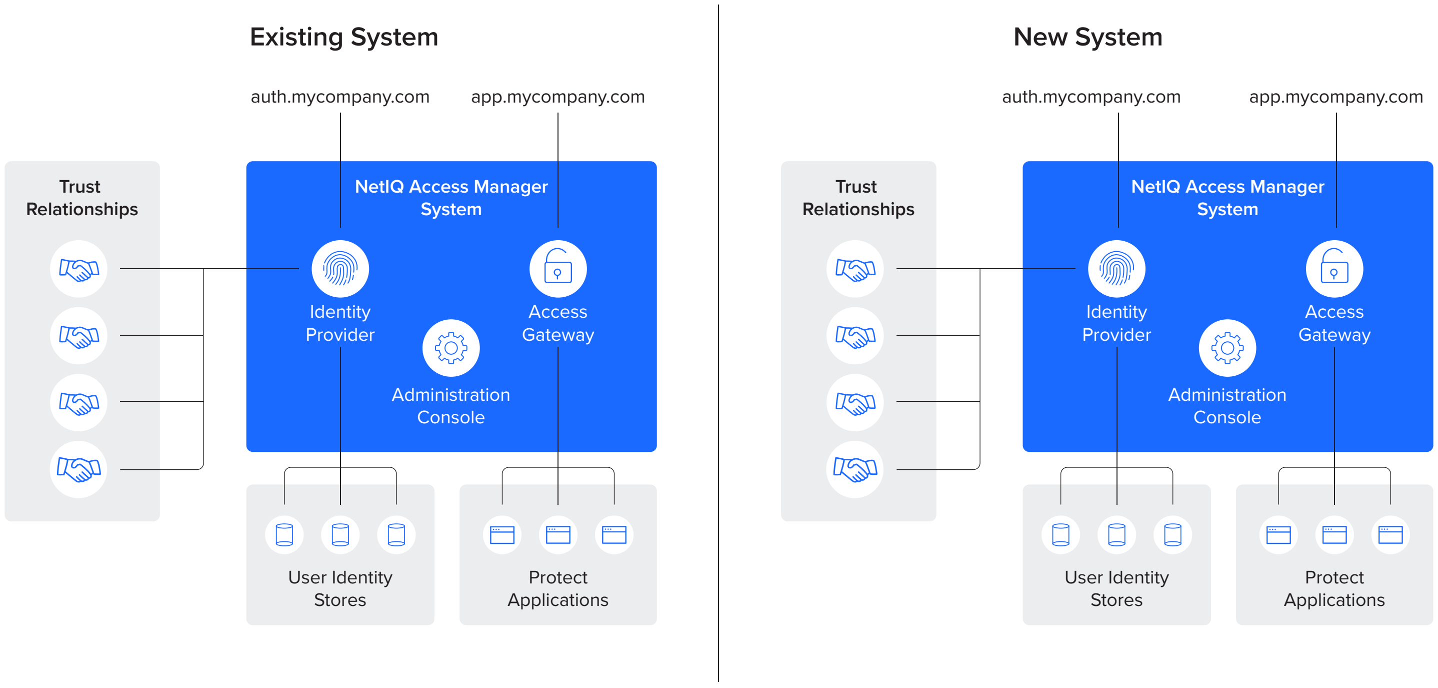

Figure 7. Approach 2—Full cutover to a distinct new system

Implementation Process:

- Set up the new system and configure it with a new Identity Provider DNS name.
- Configure reverse proxies on the new Access Gateways that replicate those on the old system.
- Test the new system as fully as possible using host file entries.
- Cutover all federation partners and Access Gateway clusters at once.
- Verify all applications and federations.

APPROACH 3: NON-INTEGRATED PHASED MIGRATION

This approach is an extension of Approach 2. Instead of migrating the entire system, all trust relationships, and all protected applications at once, you migrate them piecemeal over time. Approach 3 greatly decreases the scope and risk of each migration activity. But it can have an, often unacceptable, impact on the user experience. Users might need to log into both systems, and there might be interdependencies between applications that force your migration activities to have a larger scope than you desire. This option works well when you have large numbers of trust relationships, as long as you're comfortable operating both systems for as long as it takes to migrate all of them.

Implementation Process:

- Set up the new system and configure it with a new Identity Provider DNS name.
- Configure reverse proxies on the new Access Gateways that replicate those on the old system.
- Test the new system as fully as possible using host file entries.
- Identify sets of protected applications that can be migrated and verified together in a reasonable change window. This can often be done at the reverse proxy level so that you migrate an entire domain name at a time.
- Methodically migrate each trusted relationship to the new system.

APPROACH 4: INTEGRATED PHASES MIGRATION

This approach is the same as Approach 3 except that we establish a trust relationship between the existing system and the new system. This allows you maintain the single sign-on experience for the users. This can actually be more complex than it sounds and requires a high level of federation expertise. You must evaluate all the possible access scenarios for each trusted partner and each application. There are corner cases where getting the user authenticated through the trust relationship chain may not be possible. This approach is most applicable when you will be operating both systems for an extended period of time.

Implementation Process:

- Set up the new system and configure it with a new Identity Provider DNS name.
- Implement a trust relationship between the existing system and the new system.
- Configure reverse proxies on the new Access Gateways that replicate those on the old system.
- Test the new system as fully as possible using host file entries. Each access use case will need to be evaluated to ensure it functions smoothly through the trust relationship between the systems.
- Identify sets of protected applications that can be migrated and verified together in a reasonable change window. This can often be done at the reverse proxy level so that you migrate an entire domain name at a time.
- Methodically migrate each trusted relationship to the new system.

APPROACH 5: SPANNED CLUSTER MIGRATION

This approach is unique to migrating from an existing NetIQ Access Manager system to a new NetIQ Access Manager system. You must upgrade your current system to the latest version and patch level before proceeding. In this case, you are not starting over with a clean system, and there is no change to the system identity. You are simply replacing the existing cluster nodes with new cluster nodes. The complexity in this scenario is in ensuring adequate network connectivity and managing both global and local load balancing.

In many cases, you can continue to operate the legacy infrastructure and the new infrastructure concurrently. This “hybrid” model can provide additional fault tolerance and can allow you to ease into a fully cloud-based deployment. The diagram below shows a hybrid infrastructure with the cloud components deployed in AWS. The AWS nodes can be deployed as either containers or virtual machines.

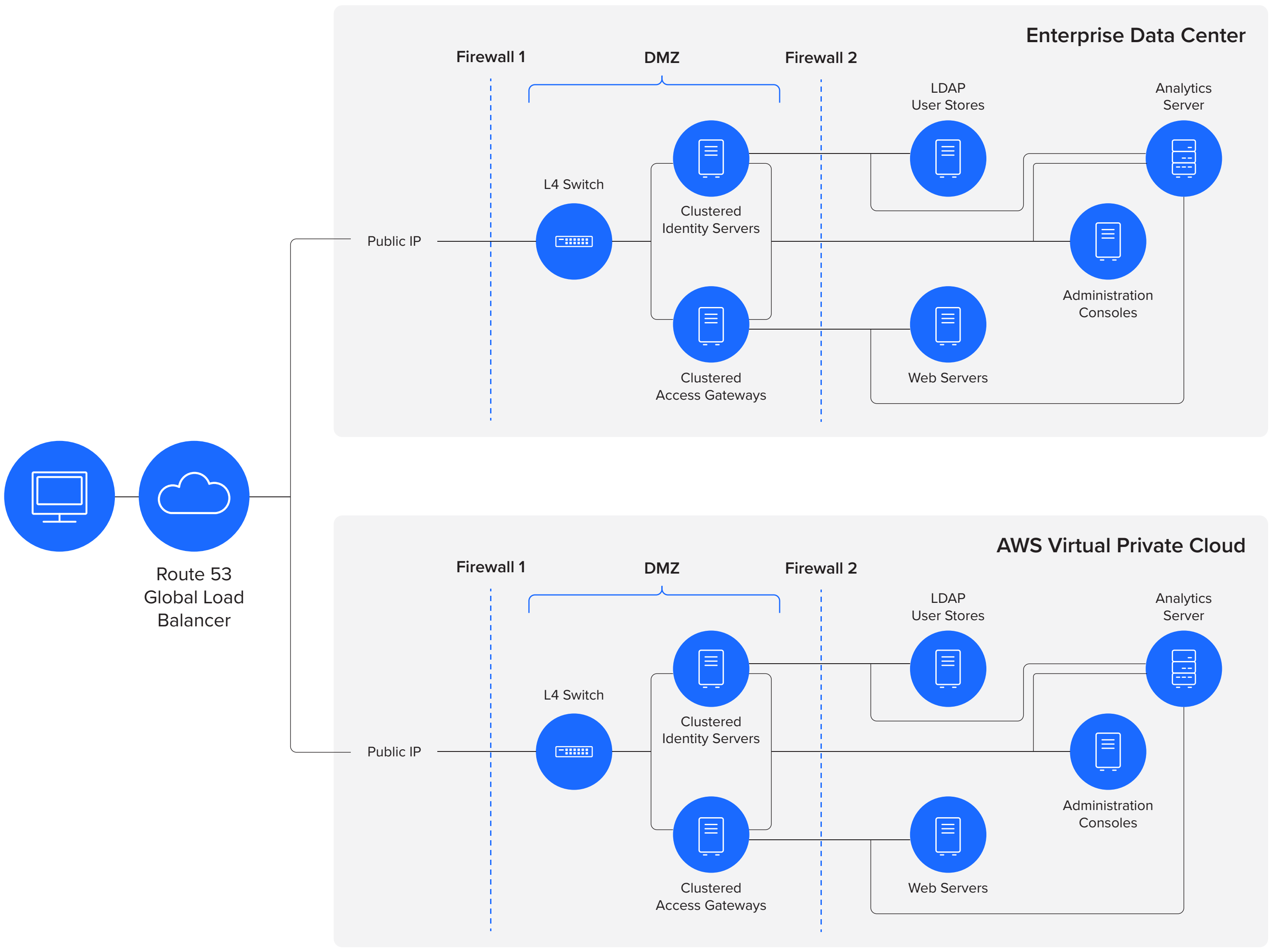

Figure 8. The hybrid infrastructure model

Implementation Process:

- Ensure that network connectivity exists between the new environment and the existing environment. See the NetIQ Access Manager documentation for the connectivity required for each component. Pay particular attention to the network performance requirements when an on-premises application is proxied through cloud-based Access Gateways.

- Set up the new Identity Server and Access Gateway nodes in the new environment.
- Set up a new secondary Administration Console in the new environment.
- Add the new nodes to the existing cluster configurations.
- Test the new nodes as fully as possible using host file entries.
- Add the new nodes to your load balancing scheme if possible. If this is not possible, then each DNS name can be migrated independently.
- Test each application and federation relationship. (Watch for issues caused by IP-based whitelisting.)
- Retire legacy nodes and infrastructure if desired.

Design the System Implementation

Once you have selected a migration approach, the next step is to design your implementation. The good news is that NetIQ Access Manager has done much of this for you in the provided Helm chart templates. The first step is to determine your network and fault tolerance infrastructure. Both EKS and AKS have guides to help you:

- **https://docs.aws.amazon.com/eks/latest/userguide/eks-networking.html**
- **https://aws.amazon.com/quickstart/architecture/amazon-eks/**
- **https://docs.microsoft.com/en-us/azure/aks/concepts-network**

For a production deployment, your Kubernetes cluster should consist of at least two worker nodes. In this minimal configuration, you would configure containers for the Administration Console, Identity Server, and Access Gateway to run on each of the two nodes. Additional nodes can then be added to run additional instances of the Identity Server and Access Gateway, as dictated by load and performance requirements. In general, you can assume that you will require the same number of instances as your current NetIQ Access Manager clusters. If you had four Identity Servers and six Access Gateways, then you will likely need the same number of containers. You might actually need less total worker nodes since a node can host both an Identity Server and an Access Gateway.

The next step is to decide how you want to route requests to your containers. In EKS, you have the option of assigning a public IP routing traffic to it, just like any other public IP. However, doing so limits flexibility. The alternative is using the Kubernetes service called Ingress. The Ingress service is essentially an application-level firewall and load balancer with the ability to route traffic based on the status of the container. In AKS, you must use the Ingress service because there is no option for public IP addresses. In both EKS and AKS, there are additional options for Global Load Balancing to route traffic to multiple data centers or availability zones.

Each of the containers requires persistent storage for configuration files and logs. This storage might be local to the worker node, or a Kubernetes Volume driver might provide it. Both EKS and AKS provide options that virtualize the storage, allowing it to be easily maintained independent of a specific node. Because inter-node communication is based on IP address, it might be simpler to spin up a new instance than to move the storage and IP address to a new worker node. NetIQ Access Manager's native cluster architecture removes any dependence on any single node.

Once complete, your design is expressed as a Helm chart so that it can be applied to the Kubernetes cluster. Note that there might be some infrastructure elements, such as the Global Load Balancing configuration, that Kubernetes doesn’t manage. These elements will need a separate configuration. In EKS, most of these elements can be specified using an AWS CloudFormation template. AKS has similar capabilities.

Deploy, Test, Migrate

Deployment is incredibly simple. Here are the high-level steps required for EKS:

1. Create an AWS account if you don’t already have one.
2. Check to see if deploying the system will exceed any resource limits on your account. There are limits on the number of VPCs, Security Groups, IAM Roles, Auto Scaling groups, and instances. This should not be an issue if your account is not already using significant AWS resources.
3. Create a new VPC for NetIQ Access Manager (An existing VPC can be used).
4. Create an EKS cluster in your VPC.
5. Create worker nodes for your cluster.
6. Create persistent storage volumes.
7. Create security groups and IAM Roles to manage access to the system.
8. Configure the kubectl management utility on your local workstation to work with your AWS cluster.
9. Install Helm on your workstation.
10. Apply your Helm chart.

Once the Helm Chart has been applied, the container images will be downloaded from the NetIQ Access Manager repository and run on the worker nodes. The process will take about 10 minutes. When that process is finished, you will have a complete NetIQ Access Manager system ready for configuration.

Before you proceed with the final configuration, you should spend some time getting familiar with operating an NetIQ Access Manager system in Kubernetes. Practice destroying and rebuilding the system. Practice adding and removing containers. Practice upgrading the NetIQ Access Manager images. You need to be familiar and comfortable with all these processes. At this point, you should also test auto-scaling, load balancing, and fault tolerance.

Once you’re confident the infrastructure is solid, then you should fully implement your Identity Server and Access Gateway configurations. At this point, DNS is still directing traffic to the existing system, but you can test using host files or an alternate DNS server. Remember that if you use host files, you will also need to ensure that DNS resolution from the containers results in the correct test addresses.

Once you have validated your configuration, you should perform load testing to ensure the new system can match or exceed the performance of the existing system. You should establish a performance baseline that can be used for ongoing capacity planning.

The last step is to migrate applications and integrations based on your chosen migration approach.

As digital transformation works deeper into organizational processes and expands across the business, an increasing number of NetIQ Access Manager customers are moving the management of that underlying infrastructure to the cloud. It allows them to offload the cost and overhead of maintaining NetIQ Access Manager disaster recovery environments and ensures high predictability of service performance. Hopefully, this position paper has made your access management transition to the cloud a simpler one.

About OpenText

OpenText has completed the purchase of Micro Focus, including CyberRes. Our combined expertise expands our security offerings to help customers protect sensitive information by automating privilege and access control to ensure appropriate access to applications, data, and resources. NetIQ Identity and Access Management is part of OpenText Cybersecurity, which provides comprehensive security solutions for companies and partners of all sizes.

Connect with Us

www.opentext.com

opentext™ | Cybersecurity

762-000046-004 | O | 01/24 | © 2024 Open Text