

Reference Architecture for B2C and G2C Systems with Micro Focus IAM

Implementing an access management system that provides services to the public presents many challenges that differ from typical enterprise implementations. Chief among these are issues of scale because public-facing systems typically serve hundreds of thousands to millions of users

Table of Contents

page

Introduction	1
Configuring Access Manager for the Largest Environments	1
Reference Architecture	3
System Implementation	6
Account Management Services	12
Support and Operations Functionality	15
System Tuning	18
Performance Testing	19
Conclusion	22

“There is no such thing as CIAM, at least not as a separate discipline within IAM. There are technologies of higher relevance when dealing with customers and consumers than when dealing with employees. But there neither are technologies that are required for CIAM only, nor is there any benefit in trying to set up a separate CIAM infrastructure.”*

MARTIN KUPPINGER

Analyst
KuppingerCole Analysts

** There is no Consumer Identity & Access Management at all—at least not as a separate discipline*

Introduction

Designing an access management system that provides services on a massive scale presents many challenges, including delivering reliable performance. This reference architecture serves as a baseline from which you can plan your architecture.

Configuring Access Manager for the Largest Environments

It's not just the technical infrastructure that needs to scale. Business-to-Consumer (B2C) and Government-to-Citizen (G2C) systems require processes, tools, and methodologies that allow a high degree of automation and self-service. Manual administration of such large numbers of user identities is simply not practical.

So, does meeting the challenges of B2C or G2C require different access management technologies than you are already using within your enterprise? Not if you're using the Micro Focus® Identity and Access Management (IAM) suite of products. Micro Focus Access Manager has been used in many large-scale implementations around the world. Governments and businesses have had tremendous success deploying public-facing applications protected by Access Manager.

In this paper, we will identify the challenges typical of public-facing access management systems and an approach to solving those challenges. We will present a reference architecture that is the result of decades of experience, and we will show through large-scale testing just how powerful such a system can be while still being economical and manageable. As shown by recent G2C implementations that experienced very high-profile issues, simply throwing money and resources at the problem is not enough. It's critical that scalability, reliability, and automation be built in from the start.

New GDPR requirements:

- Inform users of data breaches without undue delay (within 72 hours) after they become aware of it.
- Give end users the right to request a copy of their Personally Identifiable Information (PII) in a portable format that can also be transmitted electronically from one processing system to another.
- Provide the right to erasure: the end user can request all PII be deleted if there are no legitimate grounds for retaining it.
- Obtain valid consent to collect PII, consent which can also be withdrawn.
- Obtain regulatory approval to transfer PII outside of the EEA to countries not approved as having adequate data protection measures in place.

- Appoint a data protection officer to ensure compliance (likely applicable to companies with more than 250 employees and/or those who process more than 5,000 data subjects within 12 months, and all public bodies).
- Publish contact information for the data controller.
- Build data protection into business process, product, and service development (Privacy by Design)

Public-facing systems typically require the following capabilities:

■ **Account management**

- **Self-registration:** A process for the user to create their own account or to request that an account be created.
- **Account claiming:** A process where the user provides some information that uniquely identifies them and matches them to an account in an existing system. The identifying data could be personal information known only to them or an access code that was provided to them.
- **Linking of an existing social media account:** Allowing a user to authenticate to an existing social media account. Information may then be received from the linked account instead of requesting it from the user. This can reduce data entry and can be used to improve the user experience.
- **Identity proofing:** In its simplest form, this is validating that the provided email addresses or phone numbers belong to the user. It can be extended to include any process that verifies identity data.
- **Account deletion:** In many jurisdictions, service providers are required by law to provide a way to fully delete a user account. The “Right to be forgotten” must be supported.
- **Profile data update:** Users need a way to update their account information.

■ **Credential management**

- Password changes
- Forgotten password/password recovery
- User name/ID recovery
- The ability to use federated credentials from social media, third-party identity providers, or government identity providers
- Management of multi-factor/strong authentication credentials: phones, tokens, certificates, smart cards, etc.

■ **Account deletion—Right to be forgotten**

■ **Scalability and availability**

- Millions of identities
- Very high request volumes
- High availability
- Ability to be geographically and functionally distributed
- Can be updated and maintained without requiring downtime

New GDPR Requirements:

- Inform users of breaches within 72 hours
- Give users the right to a copy of their PII
- Provide the right to erasure
- Obtain consent to collect PII
- Obtain approval to transfer PII to countries without data protection measures
- Appoint a data protection officer
- Publish data controller’s contact information
- Build data protection into process, product, and service development

Public-Facing Systems

Typically Require:

- Account management
- Credential management
- Account deletion
- Scalability and availability
- Security
- Mobile application integration
- Scalable support functionality
- APIs and web services
- Logging, auditing, and analytics

- **Security**

- Strong cryptography
- Encryption of both in-transit and at-rest data
- Minimized and hardened attack surface
- Risk-based authentication and authorization
- Multi-factor authentication

- **Mobile application integration**

- APIs that enable mobile applications to utilize the authentication and authorization services

- **Scalable support functionality**

- A help desk/support interface for account administration
- Session-specific logging
- End user impersonation

- **APIs and web services to enable automation and integration with management systems**

- **Logging, auditing, and analytics**

- Dashboard showing system health and utilization metrics
- Integration with SIEM and reporting systems
- Comprehensive catalog of audit events

Reference Architecture

All the listed requirements can be met through the use of the Micro Focus Access Management suite of products, with minimal or no custom development. All the necessary functionality is available out of the box, requiring only configuration and branding. Requirements unique to your business can be implemented using the web services, APIs, and development frameworks provided.

The architecture described below has been widely implemented in conventional data centers, but increasingly we have seen requirements for cloud-based deployments. In this paper we will use Amazon Web Services (AWS) as the implementation platform.

We can implement a system providing all the needed functionality using only the following:

- Micro Focus Access Manager®
- Micro Focus Self-Service Password Reset
- Micro Focus Advanced Authentication
- Micro Focus eDirectory™

Access Manager

Access Manager has three high-level functional components: the Administration Console, the Identity Server, and the Access Gateway.

The Administration Console component 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 Identity Server is the service responsible for authentication and federation. It supports all common federation protocols and also has the ability to 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.

The Access Gateway service is a reverse-proxy whose primary purpose is to provide integration from the Identity Server to legacy applications that don't have the ability to integrate directly. If the application, or application platform, is able to use protocols such as SAML2, OAuth, or WS-Federation, then the Access Gateway is optional. However, there are a number of reasons you want 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 is 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 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 that provided by your application.

Access Manager currently supports two deployment modes. The first is a soft-appliance model where all the services that make up the product are deployed together on each appliance node. The second mode is to deploy the services individually on separate nodes, running either a Linux or Windows-based operating system. There are distinctive benefits to each model, which we will discuss later. Additional container-based deployment models are under active development.

The appliance deployment model reduces the required hardware footprint and has some performance benefits because the services are collocated. You can deploy a fully redundant Access Manager system with only two virtual appliances. However, there is a practical limit to how many appliance nodes can be deployed as a single cluster. This model is still capable of meeting the performance requirements of all but the most demanding loads. Appliances are also more complicated to deploy in the cloud because you must first create an AWS deployment image and then provision that image.

Micro Focus

Access Management Suite:

- Micro Focus Access Manager
- Micro Focus Self-Service Password Reset
- Micro Focus Advanced Authentication
- Micro Focus eDirectory

A service-based deployment is more flexible and allows scaling of each component separately.

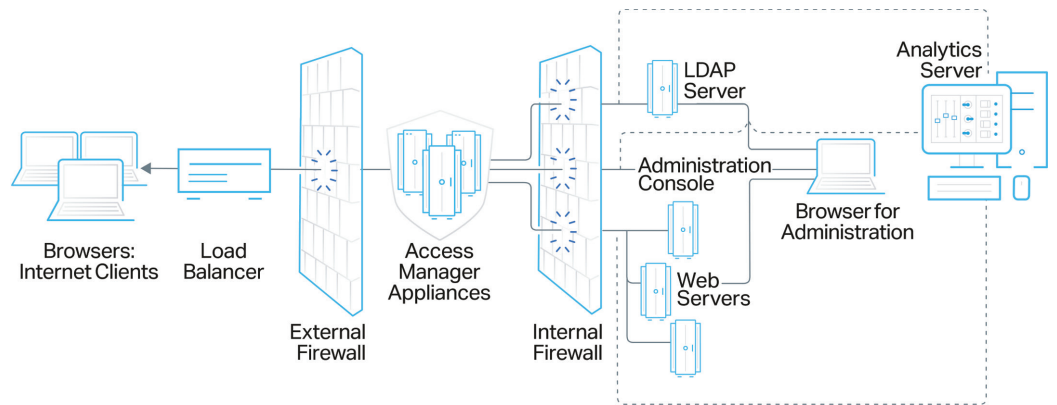


Figure 1. Access Manager Appliance Deployment

The service-based deployment is more flexible and allows scaling of each component separately. It does require a larger hardware footprint, but there are fewer limitations on the number of nodes per cluster and it allows multiple clusters to be managed as a single system. A minimum of four virtual machines are required for a fully redundant services-based implementation. Cloud-based deployment is simpler than the appliance mode, enabling you to deploy standard AWS operating system images and then install the services on the provided operating system. This paper will illustrate a service-based cloud deployment.

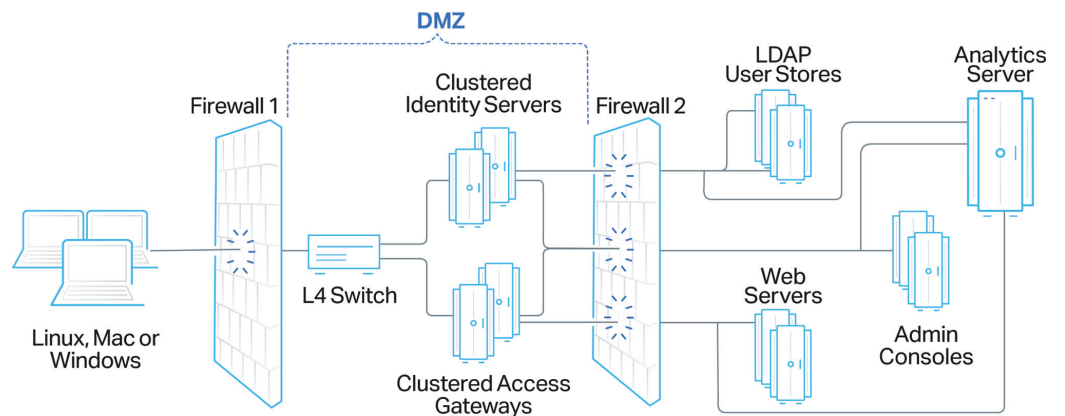


Figure 2. Standard Access Manager Deployment

Self-Service Password Reset (SSPR)

SSPR is an incredibly powerful password management system that has decades of development and refinement behind it. While the name is Self-Service Password Reset, the product actually provides a much broader range of functionality. Account creation and claiming, profile update, and even a support desk interface is included. Email verification and SMS integration enable you to verify account data and

communicate with your users. SSPR has the ability to perform custom web service and LDAP actions, so you can easily enable any needed business process. While an excellent user interface is provided, all functionality is also available through web services, so integration with your applications can be done easily. Full licensing of SSPR is included with the Access Manager license.

eDirectory

Access Manager supports multiple LDAP directories, including Azure AD, but eDirectory version 9 was an ideal choice for this reference implementation. It is incredibly fast and scalable, and capable of supporting massive deployments on relatively modest hardware. And, its multi-master replication is efficient and configurable. The data can be partitioned among multiple servers so that there are no limits on scalability and deployment options. eDirectory also supports strong, attribute-level encryption and fine-grained access control to fully secure your user data. The default configuration uses a public-private key mechanism for passwords that is more secure than simple hashing mechanisms. License for up to 250,000 users is included with Access Manager.

Databases can also be used for authentication and authorization, but in our experience, scaling a database to match the performance of an LDAP directory is complex and considerably more expensive. Directories, especially eDirectory, enable us to scale to hundreds of millions of users relatively easily.

Advanced Authentication

Multi-factor authentication is becoming a must-have feature. Advanced Authentication enables a full range of authentication methods. One of the most powerful and frequently implemented is smartphone-based authentication. An authentication app is available for iOS, Android, and Windows phone. Advanced Authentication is tightly integrated with Access Manager, so enabling it allows you to use any of the strong authentication methods provided.

System Implementation

Servers and Networking

Access Manager currently supports cloud deployment on either Red Hat Enterprise Linux (RHEL) or SUSE Linux Enterprise Server (SLES). The reference implementation described in this paper uses SLES 12 SP3.

There are multiple ways of implementing redundancy in AWS, and the complexity increases as you choose options that provide higher guarantees of availability. For our reference implementation, we chose to distribute the cluster nodes across two AWS availability zones and to enable Cross-Zone Load Balancing (<https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/enable-disable-crosszone-lb.html>). You could also choose to deploy to multiple AWS regions if your application demands doing so. Using multiple regions would require DNS-based global load balancing; this is because Elastic Load Balancers alone cannot route traffic across regions.

eDirectory is an ideal choice—it's incredibly fast and scalable, and capable of supporting massive deployments on relatively modest hardware.

Creating a public subnet and a private subnet within each availability zone allows additional network security policies to be placed around the data stores and administration consoles. The private subnet in each zone will contain an Access Manager administrative console, at least one replica of the user directory, and any databases needed by the application.

Within each availability zone, we have created a public subnet and a private subnet. This allows additional network security policies to be placed around the data stores and administration consoles. The private subnet in each zone will contain an Access Manager administrative console, at least one replica of the user directory, and any databases needed by the application.

The public subnet will contain the Access Manager Identity Servers and the Access Manager Access Gateways (reverse proxies). Each of these services is deployed as a cluster, with the number of nodes determined by the load and redundancy requirements. The number of nodes in each cluster may be scaled up or down as needed. Cluster scaling can be done manually or may be automated (Contact Micro Focus Support for more information). Typically, the web and application servers supporting the application being delivered would also be placed in the public subnet. Because we are focusing on the Access Management components, we will not address the web and application servers as part of this reference implementation. The diagram below shows the access management components:

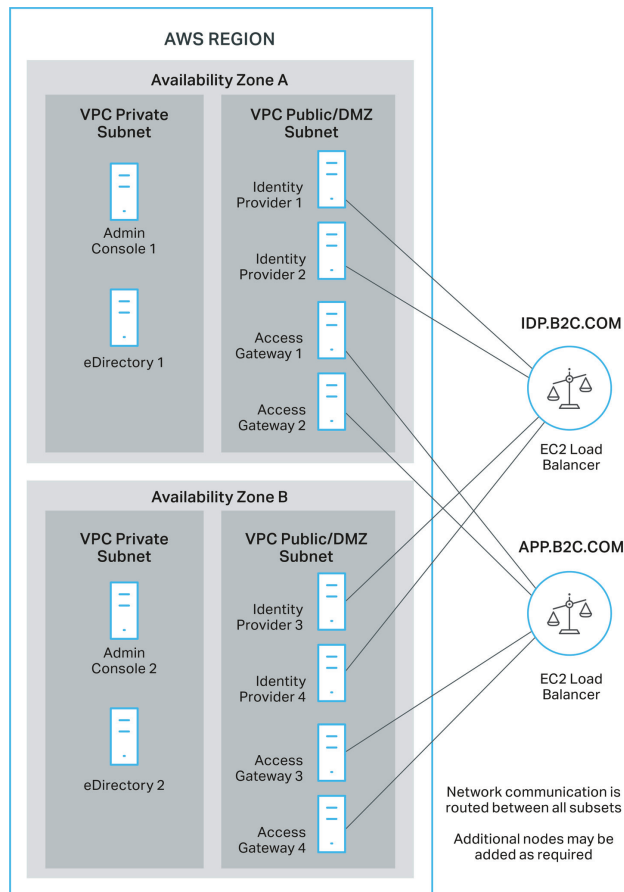


Figure 3. AWS Region

The second availability zone is a mirror image of the first. All the components are replicated to provide full redundancy. The Identity Server and Access Gateway clusters span both availability zones so that they share configuration and session information. Both zones can be used together in a fully active-active configuration.

The HTTP requests are distributed using AWS Elastic Load Balancing. AWS provides multiple types of load balancers. For this reference implementation, we are using what they refer to as a "Network Load Balancer." We could have used what they call an "Application Load Balancer" instead, but the Access Gateways provide capabilities that negate the need for the more expensive and complex Application Load Balancer.

The load balancers are configured to concurrently send traffic to all the active nodes in both availability zones. This ensures that the load is evenly distributed across all the cluster nodes. As shown below, with 10 total nodes each node receives 10% of the load.

The load balancers are configured to concurrently send traffic to all the active nodes in both availability zones, ensuring that the load is evenly distributed across all the cluster nodes.

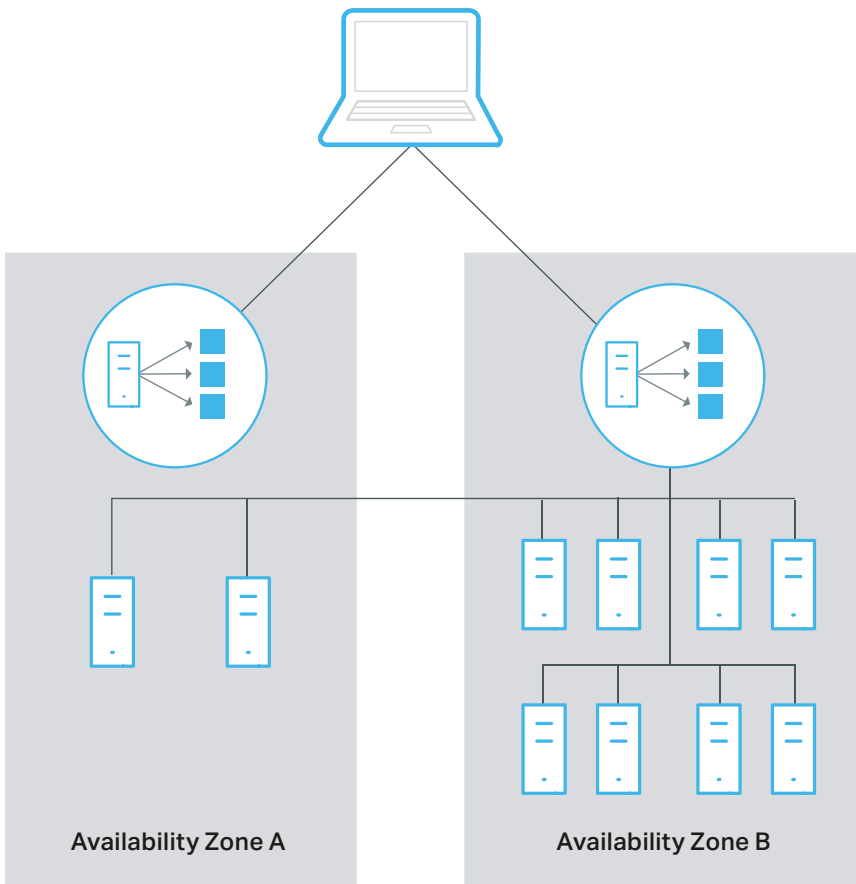


Figure 4. Load balancers

Access Gateway Proxies Are Optional, but Can Be Useful in Addressing:

- Application integration
- Path-based request routing
- Security Augmentation
- Application performance

The load balancers are configured so that a user is directed to the same node for the life of the session whenever possible. While this is not necessary, because all the nodes in the cluster share session information, it does provide added efficiency. Note that while Access Manager can manage session fail-over transparently, the application being protected might not. For fully transparent fail-over, all the components will need to maintain session state.

An additional, internal load balancer could be configured between the Access Gateway proxies and the web servers, but the proxy has built-in load balancing capabilities that usually eliminate the need for the added complexity.

The cluster nodes are both vertically and horizontally scalable, so we can determine the number to deploy based on the best balance of cost and redundancy. As mentioned earlier, we could have chosen the Appliance model and had redundancy for Access Manager using only two servers—one in each availability zone. Such a configuration could be scaled vertically by choosing an E2C instance type with more processors and memory. Alternatively, we could select smaller instance types and scale horizontally using the non-appliance model.

The Access Gateway proxies are actually optional. You may choose not to use the proxy in your configuration. We have included them in the reference architecture because they can be used to address the following issues:

- **Application integration:** Most modern application platforms are capable of directly communicating with the Identity Server using standard protocols, but in some cases, especially with legacy applications, it makes sense to use the Identity Injection or form-fill capabilities provided by the proxy.
- **Path-based request routing:** This can be used to implement composite applications where disparate back-end systems are answering requests, but you want the user to see it all as a single unified application.
- **Security Augmentation:** The proxy can be used to enforce centralized access policies that are often more flexible and powerful than those that the application platform provides. The proxy can also hide the vulnerabilities presented by the application platform. You have one relatively hardened attack surface presented to the internet, which makes responding to newly discovered vulnerabilities easier.
- **Application performance:** The Access Gateways provide a very high-performance reverse proxy cache. Utilizing this cache can dramatically reduce the load on the web and application servers.

Selecting the proper AWS instance type for each component can be challenging. In addition to available system resources and performance, you need to take into account the operational cost of each instance type. You also need to decide between horizontal scaling, vertical scaling, or a balanced approach. For the reference implementation, we have chosen a configuration that is a good compromise between the vertical and horizontal scaling approaches.

Let's start by establishing some performance goals for our hypothetical application:

- 600,000 concurrent user sessions
- 40,000 requests per second
- 2000 authentications per second
- 100,000,000 user accounts

While these numbers might seem extreme, there are real-world implementations that experience these kinds of loads (for example, a national tax filing system that sees tremendous loads during tax season). The number of user accounts primarily impacts the storage needed to house the data, but it also means that the system must be able to maintain authentication performance as the number of accounts increases. Managing a directory with hundreds of millions of user accounts has implications that are beyond the scope of this paper, but if you need assistance Micro Focus design services are available directly or through skilled partners.

For the reference implementation, we have chosen a configuration that is a compromise between the vertical and horizontal scaling approaches. Based on experience, we know that nodes with 8 CPUs and 24-32GB of RAM perform very well and can be tuned based on well-understood patterns. Larger nodes can be used, but it requires more expertise and testing to get the most out of them. Additionally, fewer large nodes provide less redundancy than more instances of a smaller configuration.

The next step is to identify the available AWS instance types that might match our requirements. Because network performance is critical, we will limit our selection to those types that provide 10Gb network interfaces. For the Identity Servers and Access Gateways, we can further limit our selection to the General Purpose and Compute Optimized categories. We can also limit our selection to the types that provide consistent processing power and at least four processors. The SLES12 instance types and the pricing current as of this writing are shown below.

General Purpose—Current Generation

Type	vCPU	ECU	Memory	Storage	Price	Comments
m5.xlarge	4	15	16	EBS Only	\$0.292 per Hour	Suitable for smaller deployments
m5.2xlarge	8	31	32	EBS Only	\$0.484 per Hour	Just Right!
m5.4xlarge	16	61	64	EBS Only	\$0.868 per Hour	Would be an option if we need to service more requests per node but more expensive than c5.4xlarge
m5.12xlarge	48	173	192	EBS Only	\$2.404 per Hour	Too big to manage
m5.24xlarge	96	345	384	EBS Only	\$4.708 per Hour	Too big to manage

Managing a directory with hundreds of millions of user accounts has implications that are beyond the scope of this paper, but if you need assistance, please contact us and we will be happy to discuss it with you in detail.

Access Manager and eDirectory can have periods of extremely high disk I/O when used in high-volume applications. AWS provides a number of options to improve storage performance, such as reserved I/O bandwidth (Provisioned IOPS) and dedicated SSD volumes.

The Instance Type we selected for each access management component is shown in the table below:

	Instance Type	vCPUs	Memory	Storage	Network
Admin Console	c5xlarge	4	8GB	EBS General Purpose SSD (gp2)	Up to 10 Gb
Identity Server	m5.2xlarge	8	32GB	EBS General Purpose SSD (gp2)	Up to 10 Gb
Access Gateway	m5.2xlarge	8	32GB	EBS General Purpose SSD (gp2)	Up to 10 Gb
eDirectory	i3.2xlarge	8	61GB	Dedicated nvme SSD*	Up to 10 Gb

*NOTE: nvme SSD storage does not persist when an instance is stopped, so you must take this into account when managing your operations.

Access Manager and eDirectory can have periods of extremely high disk I/O when used in high-volume applications. Even at normal log levels, Access Manager might produce many gigabytes of log data each hour when under heavy load. In normal operation, the directory updates user attributes on each authentication. Because of this, latency in the storage system can have serious performance impacts to the system as a whole. Unfortunately, consistent storage performance is difficult to guarantee in the AWS environment. If your log level is high, consider using EBS Provisioned IOPS or other EC2 storage options to ensure high performance.

We experienced such I/O issues while loading millions of test users into the directory when using normal EBS storage. The initial 500,000 users loaded quickly even though we were seeing I/O wait of 10%-25%. After that point we saw I/O wait climb to over 90% and loading would pause periodically. AWS provides a number of options to improve storage performance, such as reserved I/O bandwidth (Provisioned IOPS) and dedicated SSD volumes. We moved the directory to an instance type with dedicated SSD storage and performance and consistency improved dramatically. We did not experience any issues with the Identity Server or Access Gateway nodes, but you should monitor I/O wait and consider your storage options when deploying a high-volume system.

The selection of the i3.2xlarge instance type was based on the requirement to support 100,000,000 user accounts. This type was the best available balance of CPU, available SSD storage, and memory. You could get by with a less capable type if your scalability requirements are lower. You could also utilize directory partitioning and reduce the size of the database on each server.

There is a base configuration that is needed in order for the system to be redundant. To meet this requirement, the Identity Server cluster will consist of four nodes with two in each zone. Likewise, the Access Gateway cluster will consist of four nodes with two in each zone. We will use the results from load testing to determine if additional nodes are needed.

Note that we are only considering the scalability of the access management components in this paper. It's highly recommended that you do effective load testing of the system once it is integrated with your application. We often find that the performance of the protected application becomes the limiting factor. Your usage patterns might have both positive and negative impacts on Access Manager performance.

Account Management Services

Account Registration

This reference implementation will allow users to create their own accounts. We allow them to do this in two ways: they can either enter all the required information into a form or they can link an existing Facebook account. If they choose to use their Facebook account, some of the profile data will be entered for them automatically.

We will configure a profile in the SSPR New User Registration module for each registration process. For the stand-alone registration process, we will ask the user to complete the form shown below:

The selection of the i3.2xlarge instance type was based on the requirement to support 100,000,000 user accounts. This type was the best available balance of CPU, available SSD storage, and memory. You could get by with a less capable type if your scalability requirements are lower.

The screenshot shows a web browser window with the address bar displaying "Secure | https://b2csp1.pointbluetech.com/sspr/public/newuser". The page title is "Self Service Password Reset". The main content area is titled "New User Registration" and contains the following form fields and instructions:

To register a new account, please complete the following form.

User Name*

First Name*

Last Name*

Email Address*

Mobile

New Password*

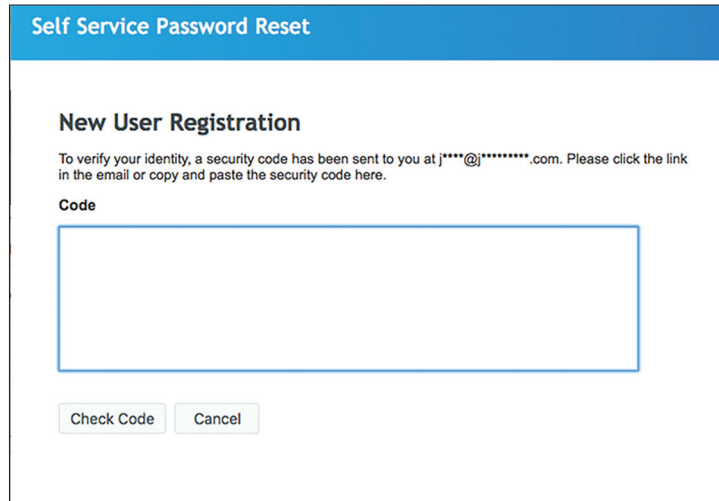
- Password is case sensitive.
- Must be at least 4 characters long.
- Must be no more than 12 characters long.
- Must not include any of the following values: password test
- Must not include part of your name or user name.
- Must not include a common word or commonly used sequence of characters.

Continue Go Back Cancel

Figure 5. Self Service password reset

There are many ways that new user registration processes can be implemented, using the very flexible options included with SSPR.

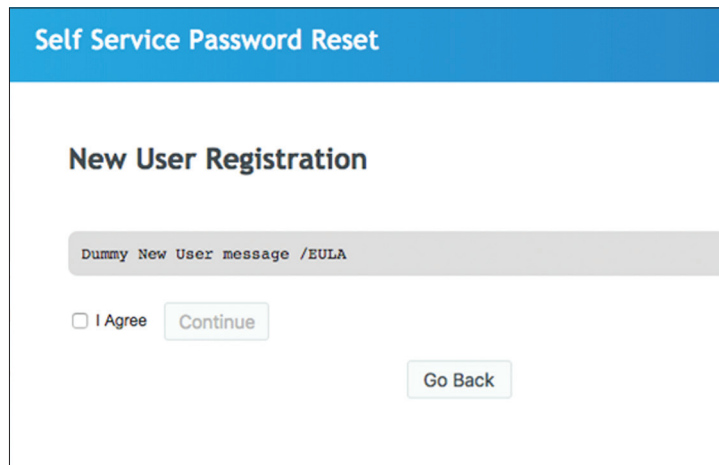
After the form is submitted, the system is configured to send an email to validate the email address and activate the account. SSPR also supports validation of the phone number via SMS. The user is presented with the form shown below to complete the verification process:



The screenshot shows a web interface with a blue header bar containing the text "Self Service Password Reset". Below the header, the main content area has the title "New User Registration". Underneath the title, there is a paragraph of text: "To verify your identity, a security code has been sent to you at j****@j*****.com. Please click the link in the email or copy and paste the security code here." Below this text, the word "Code" is displayed above a large, empty rectangular input field. At the bottom of the form, there are two buttons: "Check Code" and "Cancel".

Figure 6. New user registration code

We can also choose to include a license agreement or other notification as part of the registration process.



The screenshot shows a web interface with a blue header bar containing the text "Self Service Password Reset". Below the header, the main content area has the title "New User Registration". Underneath the title, there is a gray horizontal bar containing the text "Dummy New User message /EULA". Below this bar, there is a checkbox labeled "I Agree" followed by a "Continue" button. At the bottom right of the form, there is a "Go Back" button.

Figure 7. New user registration agreement

Registering using a Facebook account is similar. The user clicks on a link inviting them to use their Facebook account. An authentication request is sent to Facebook. The user authenticates if needed and is asked if they want to share account information with our application. Facebook returns an authentication response

that can be used to provision the user account in our user database. In this case, we want to have more information than Facebook provides, so once the user has been authenticated by Facebook, we present a form where we can collect additional information.

The fields with data provided by Facebook are pre-filled. The user does not need to set a password, since we are allowing them to use their Facebook credentials. As before, we are presenting them with a notification/agreement and then we complete the provisioning of their account.

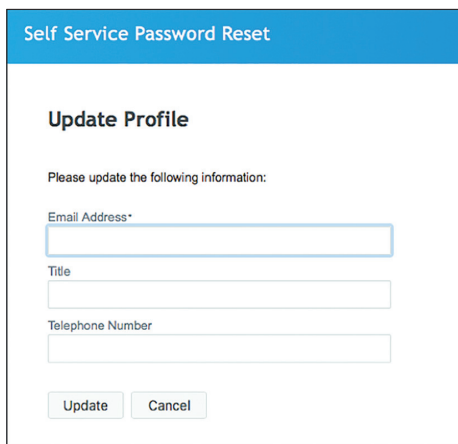
Note that for either process we could have also required a CAPTCHA response during verification by simply selecting that option in the user registration profile.

Let's assume we have another class of users, such as agents who sell our services. For these users we already have information about them in our systems. We need a process to set them up for access. This is done using the SSPR Account Claiming module. In this case we will create stub accounts for them based on the identity data we already have. Then we will notify them and direct them to the application. Included with the notification will be a One-Time Password or activation code that they can use as part of the account claiming process. During registration, they enter the code and any other information we might want to use for verification. If the entered information matches our records, they are allowed to select a user ID and set a password.

There are many ways that these registration processes can be implemented, using the very flexible options included with SSPR. What we have implemented here required absolutely no custom code.

Profile Update

SSPR provides a profile update service that we will configure to allow users to update their name and contact information.



The screenshot shows a web interface titled "Self Service Password Reset" in a blue header. Below the header is a white box titled "Update Profile". Inside this box, there is a prompt: "Please update the following information:". Below the prompt are three input fields: "Email Address*" (with an asterisk indicating it is required), "Title", and "Telephone Number". At the bottom of the form are two buttons: "Update" and "Cancel".

Figure 8. Update profile

The ability to delete an account is now a legal requirement in many places. We make this simple for users by configuring the Delete Account module in Self-Service Password Reset.

In some cases it can be helpful to see exactly what the user sees when they access your application. Doing this securely and with the user's consent can be difficult. Access Manager provides a role-based and fully auditable method for user impersonation.

Account Deletion

The ability to delete an account is now a legal requirement in many places. We will make this simple for users by configuring the Delete Account module in SSPR. We will present the user with a friendly warning message to make sure they know what they are doing.

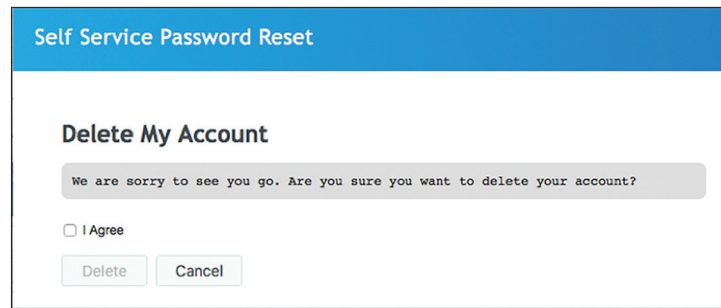


Figure 9. Deleting an account

Support and Operations Functionality

Help Desk

Providing effective self-service capabilities is fundamental to supporting large user populations, but you will likely still require some manual administration capability. SSPR provides a Help Desk interface that is perfectly suited for customer support personnel. You can even implement multiple levels of administrator access if desired.



Figure 10. Self-service password reset

User Impersonation

In some cases it can be helpful to see exactly what the user sees when they access your application. Doing this securely and with the user’s consent can be difficult. Access Manager provides a role-based and fully auditable method for user impersonation. Support personnel can be assigned a role that allows them to impersonate users that are in a specified role. The screenshot below shows a configuration that allows the “Service Desk Operator” role to impersonate users in the “Normal User” role, while requiring the “Service Desk Supervisor” role to impersonate a user in the “High Security User” role.

For planning purposes, we recommend that you assume 50KB per user account. Using a custom class might allow you to reduce the storage requirements.

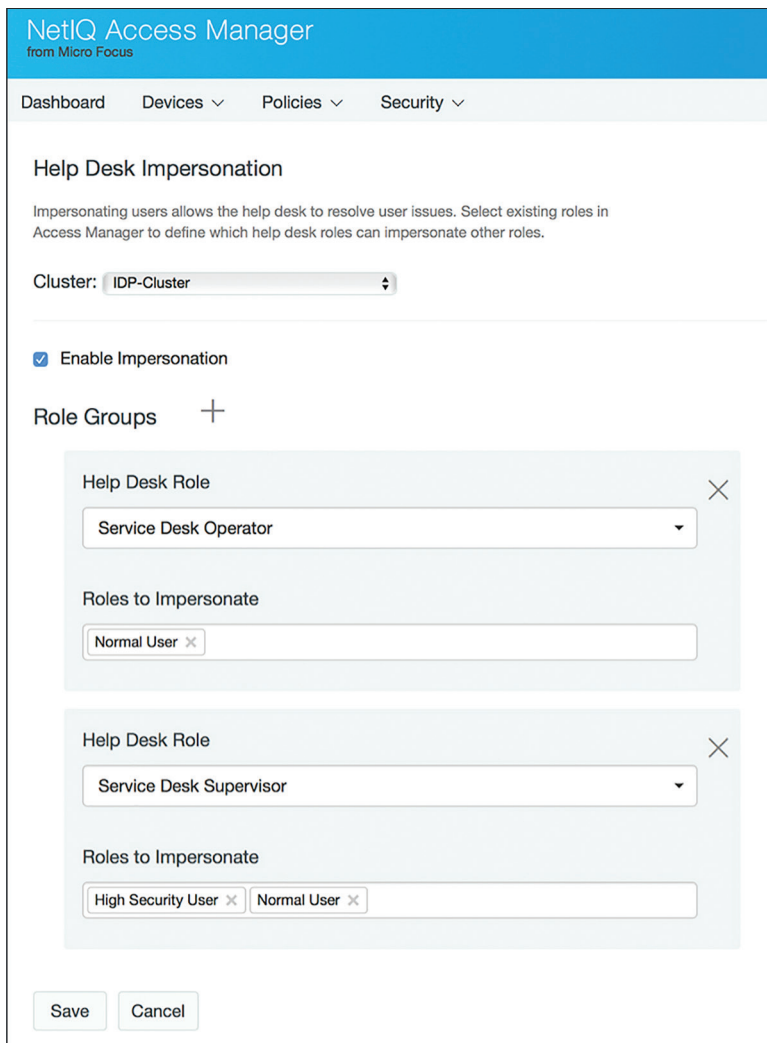


Figure 11. Access Manager

The Access Gateway is comprised of two major functional processes: an Apache-based reverse proxy that is responsible for delivering content and a Tomcat-based component where the federation service provider and other access control logic is implemented.

The user must grant permission before the impersonation session can begin:

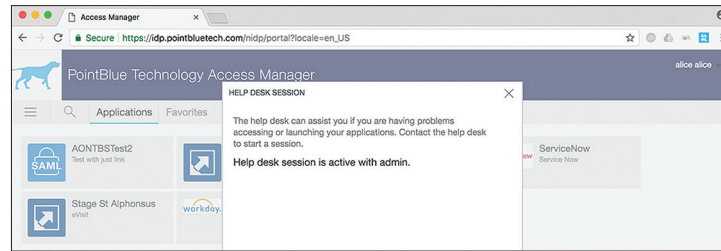


Figure 12. Access manager help desk

A notice is presented to the user for the duration of the impersonation session:

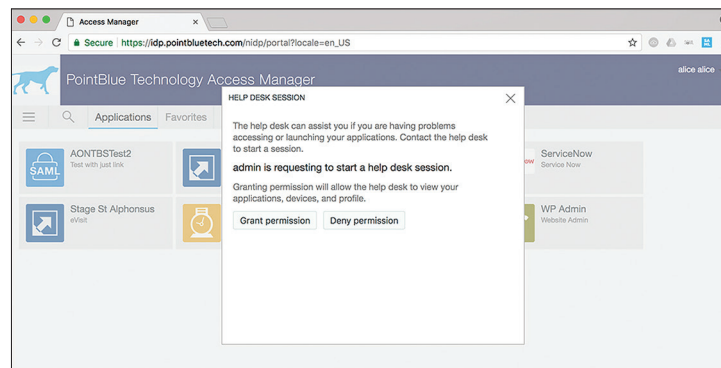


Figure 13. Access manager help desk notification

Specific audit events are generated throughout this process so that security requirements can be fully met.

Session-Based Logging

When dealing with a high-volume system, you need a way to isolate the log events for each session. Access Manager enables you to identify a specific session from which you can generate a log file containing only that session's events. This capability is invaluable because wading through gigabytes of log data to gain understanding of what's happening to a specific user session is inefficient and not a good use of an engineer's time.

System Tuning

For testing purposes, we created one hundred million user accounts in the directory. This resulted in a 390GB database. Dividing the size of the database by the number of users gives us an average record size of 39KB. Our user records contain only a few basic demographic attributes and the private key that represents their password. For planning purposes, we recommend that you assume 50KB per user account. Note that we used the normal user object class, which has some overhead that is not usually needed in B2C or G2C implementations. Using a custom class might allow you to reduce the storage requirements. Usernames are indexed for maximum performance.

We will select accounts randomly for each load test so that caching will not artificially improve the results. In real-world usage, caching will be beneficial because there is often a subset of users that access the system more frequently than others. We will also direct our tests to a single Identity Server node and a single Access Gateway node. This will allow us to establish the baseline performance for a single node. We can then extrapolate the performance provided as we add more nodes.

The default settings for many Access Manager and eDirectory tuning parameters are much too low for a high-volume system. The default values assume the minimum recommended hardware configurations. The values shown below are appropriate for the hardware configurations tested and sufficient to meet the performance goals of the reference implementation. When implementing a production system, you should use these values as a reference but adjust them based on your own performance testing.

eDirectory Tuning

We applied the following tuning to the eDirectory server:

- Set the database cache to 2GB
- Set the "n4u.server.max-threads" parameter to 512

Access Gateway Tuning

The Access Gateway is comprised of two major functional processes: an Apache-based reverse proxy that is responsible for delivering content and a Tomcat-based component where the federation service provider and other access control logic is implemented. Both components require tuning changes in a high-volume environment. We modified the Access Gateway settings as follows:

- Increased the maximum java heap size for Tomcat to 16GB
- Set the initial java heap size for Tomcat to 16GB
- Increased the java stack memory to 1024k
- Set the ESP_Busy_Threshold to 5000
- Increased the maximum number of threads for the AJP Tomcat connector to 2000
- Reduced the "Keep Alive Interval" on all reverse proxy instances to 15 seconds

We recommend Micro Focus StormRunner for testing. This cloud-based, load and performance testing platform is capable of generating the very large loads needed for testing B2C and G2C systems.

It is imperative that you do effective load testing of the system after it is integrated with your application. We often find that the performance of the protected application becomes the limiting factor.

We set the configuration parameters for the Apache Worker MPM module to the values shown in the table below:

Parameter	8 CPU Instance
ThreadLimit	1000
StartServers	9
MaxClients	9000
MinSpareThreads	9000
MaxSpareThreads	9000
ThreadsPerChild	1000
ServerLimit	10
MaxRequestsPerChild	0

Identity Provider Tuning

For the Identity Server, we modified the following Tomcat parameters and configured settings that allow for a greater number of LDAP connections:

- Increased the maximum java heap size for Tomcat to 16GB
- Set the initial java heap size for Tomcat to 16GB
- Increased the java stack memory to 1024k
- Increased the maximum number of Tomcat threads for the HTTPS connector to 2000
- Set the "IdapLoadThreshold" to 600 in web.xml
- Created multiple LDAP replica entries for the User Store and set each to the maximum of 50 connections; total of 100 connections per replica for each IDP
- Set authentication session inactivity timeout to 15 minutes

Web/Application Server Tuning

The tuning required for your application to match the performance provided by Access Manager is obviously entirely specific to your platform and application. For this reference platform, we used largely static HTML and configured sufficient server resources to ensure that the protected application was not a performance bottleneck.

Performance Testing

We will use Micro Focus StormRunner for testing. StormRunner is a cloud-based load and performance testing platform. It is capable of generating the very large loads needed for testing B2C and G2C systems.

To test authentication performance, we will isolate the authentication process on a single server to establish a baseline for capacity planning and complete these steps:

1. Load the B2C application public landing page.
2. Enter credentials and submit them for authentication.

The user is authenticated and then redirected to a private application landing page.

The results are very impressive. As the graph below shows, the single identity server was able to hit peaks above 1200/s and was able average around 1100/s.

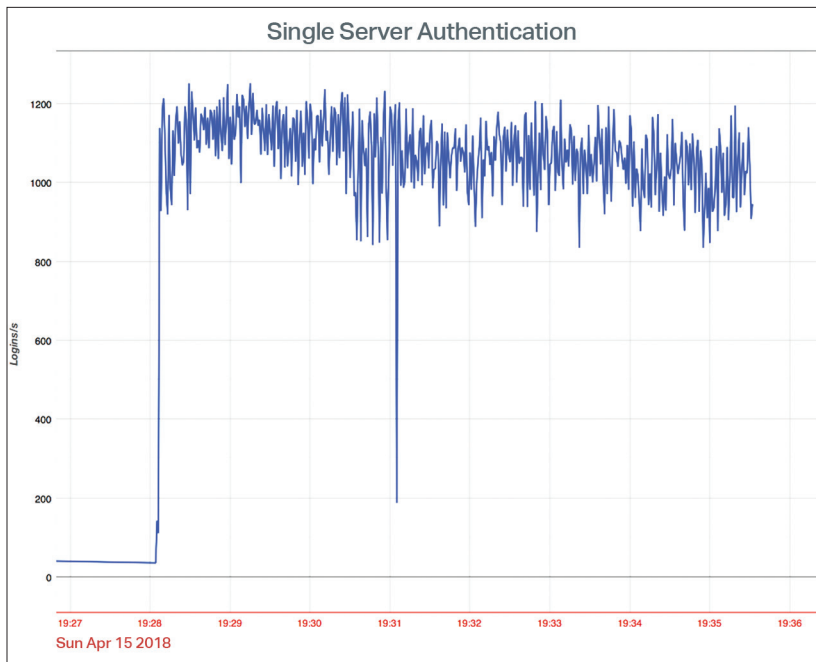


Figure 14. Single server authentication

This means a single server could support 66,000 authentications in a minute. To scale this number up to a full cluster, we need to include a factor that accounts for the impact of inter-node communication. With well- configured load balancing, we can assume that the impact would only be about 10%. Based on this, a cluster of 12 could support $792,000 \times 0.9 = 712,800$ authentications in a minute.

These numbers mean you can authenticate millions of user sessions per hour. Very few applications ever see this kind of load. But remember, this is a baseline. Your capacity planning should be based on 80-85% of this number. This is especially important in a cloud deployment, because you cannot assume that the infrastructure will perform exactly the same at all times. We saw these numbers vary slightly from day to day, with the weekend showing consistently better performance.

Determining the number of Access Gateway nodes required is more complex because the application being protected has an enormous impact on overall performance. The mix of public page requests versus those requiring authorization policy evaluations will change the performance profile.

The reference implementation provides redundancy, high availability, and tremendous scalability. Your implementation might not be this large, but the example provided should give you a good starting point for designing a system that matches your requirements.

It is also important to remember that this test was done using the absolute fastest authentication configuration. The login page was static and hosted on a separate web server, so the IDP did not need to generate a conventional JSP-based login page. The authentication method used was name/password-based LDAP authentication against eDirectory. This authentication method is extremely fast and efficient. Other authentication methods will not achieve this level of performance, so you must test and establish a baseline for each method used in your environment. Other directories might not match this level of performance, especially when loaded with 100,000,000 user accounts.

The authentication performance goal for our hypothetical application was 2000 authentications per second. This goal can be met with only two Identity Server nodes, but doing so will put us close to the maximum performance metric that we established. We will want at least one additional node per availability zone so that we have some excess capacity. That gives us three nodes. We now need to look at how many sessions each node can support, because we might find that we need additional nodes to support the concurrent session load.

During this test, the Identity Server reached 2,231,061 active sessions and we still had 33% of our 16GB of heap memory available. Based on this result, it's safe to use 225,000 sessions per node for our capacity planning calculations. Note that the actual maximum might be much higher, especially if we were to increase the heap memory allocation, but it's better to be conservative in our planning. The three nodes calculated above would support a total of 675,000 (3 x 225,000) sessions, so we don't need to increase the node count to meet our target of 600,000.

Determining the number of Access Gateway nodes required is more complex because the application being protected has an enormous impact on overall performance. The mix of public page requests versus those requiring authorization policy evaluations will change the performance profile. The percentage of resources served from cache and the response time for non-cached resources will have an impact.

Because these factors vary so much, based on the application, we are going to do some calculations using values that are conservative and have been seen in real-world deployments. It is imperative that you do effective load testing of the system after it is integrated with your application. You might find that you are able to scale down from these conservative recommendations.

Here are the Access Gateway metrics we will use to determine a baseline number of nodes:

- Concurrent Sessions per AG node: 150,000
- Request per second on each node: 10,000
- Concurrent active connections per node: 7500 (The Apache tuning we did enforces a hard limit of 9000)

Contact us at:
www.microfocus.com

To meet our target of 600,000 concurrent sessions, we will need 4 nodes ($600,000 / 150,000 = 4$). To meet the 40,000 requests per second target, we will also require 4 nodes ($40,000 / 10,000 = 4$). The math says 4 nodes and that is where you should start your testing. However, it is a good idea to have spare capacity available, so we will use 5 nodes in our reference deployment. Remember, one of the benefits of an AWS deployment is being able to rapidly scale up (or down) as needed.

Based on these calculations, our initial server instance inventory for a single availability zone is:

Component	Instances	Price
Admin Console	1 x c5xlarge	\$0.27 /hr
Directory Replica Server	2 x i3.2xlarge	\$1.45 /hr
Identity Server	3 x m5.2xlarge	\$1.14 /hr
Access Gateway Server	5 x m5.2xlarge	\$1.90 /hr
		Total: \$4.76 /hr

Conclusion

We have provided a concrete example of how Micro Focus Access Manager can be implemented within AWS to support a high-volume B2C or G2C application. The reference implementation provides redundancy, high availability, and tremendous scalability. Your implementation might not be this large, but the example provided should give you a good starting point for designing a system that matches your requirements.

Learn More At

Access Manager: www.netiq.com/products/access-manager/

Advanced Authentication: www.netiq.com/products/advanced-authentication/

eDirectory: www.netiq.com/products/edirectory/

Self-Service Password Reset: www.netiq.com/products/self-service-password-reset/

StormRunner Load: www.microfocus.com/srl