Deployment Planning Guide

/ ForgeRock Access Management 7.1

Latest update: 7.1.4

ForgeRock AS.
201 Mission St., Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Guide to planning ForgeRock® Access Management (AM) deployments. ForgeRock Access Management provides intelligent authentication, authorization, federation, and single sign-on functionality.

Table of Contents

Overview

The Deployment Planning Guide helps you to plan the deployment of ForgeRock Access Management, including implementing training teams and partners, customization and hardening, and development of a proof-of-concept implementation.

This guide is written for access management designers, developers, and administrators who build, deploy, and maintain Access Management services and features for their organizations.

Quick Start

About Identity Access Management

Discover how AM provides solutions to secure your identity data and your organization's resources.

Deployment Architecture

Create a good, concrete deployment plan tailored to your requirements.

Deployment Topology

View example topology of a multi-city multi-data-center deployment across a wide area network (WAN).

Size Hardware and Services

Size servers, network, storage, and service levels required by your Access Management deployment.

Deployment Requirements

Learn about the deployment requirements for Access Management sizing, including storage requirements and memory requirements.

Quick Start

Learn how to get started with an Access Management deployment.

About ForgeRock Identity Platform™ Software

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

Chapter 1

Understanding Identity Access Management

The proliferation of cloud-based technologies, mobile devices, social networks, Big Data, enterprise applications, and business-to-business (B2B) services has spurred the exponential growth of identity information, which is often stored in varied and widely-distributed identity environments.

The challenges of securing such identity data and the environments that depend on the identity data are daunting. Organizations that expand their services through internal development or acquisitions must manage identities across a wide spectrum of identity infrastructures. This expansion requires a careful integration of disparate access management systems, platform-dependent architectures with limited scalability, and ad-hoc security components.

ForgeRock, a leader in the IAM market, provides proven solutions to securing your identity data.

Identity Management is the automated provisioning, updating, and de-provisioning of identities over their lifecycles. Access Management is the authentication and authorization of identities who desire privileged access to an organization's resources. Access management encompasses the central auditing of operations performed on the system by customers, employees, and partners. Access management also provides the means to share identity data across different access management systems, legacy implementations, and networks.

Continue reading to learn more about AM and what it can do for your environment:

More than Just Single Sign-On

AM is an all-in-one, centralized access management solution, securing protected resources across the network and providing authentication, authorization, Web security, and Federation Services in a single, integrated solution. AM is deployed as a simple `.war` file and provides production-proven platform independence, flexible and extensible components, as well as a high availability and a highly scalable infrastructure. Using open standards, AM is fully extensible, and can expand its capabilities through its SDKs and numerous REST endpoints.

AM is part of the ForgeRock Identity Platform, and provides identity and access management of mobile-ready, cloud, enterprise, social, and partner services. The ForgeRock Identity Platform provides global consumer services across any platform for any connected device or any Internet-connected entity.

The ForgeRock Identity Platform features the following products:

- **ForgeRock Access Management**. Context-based access management system. Access Management is an all-in-one industry-leading access management solution, providing authentication, authorization, federation, Web services security, adaptive risk, and entitlements services among many other features. AM is deployed as a simple `.war` file, featuring an architecture that is platform independent, flexible, and extensible, and highly available and scalable.

- **ForgeRock Identity Management**. Cloud-focused identity administration. Identity Management is a lightweight provisioning system, built on resource-oriented principles. IDM is a self-contained system, providing workflow, compliance, synchronization, password management, and connectors. IDM features a next-generation modular architecture that is self-contained and highly extensible.

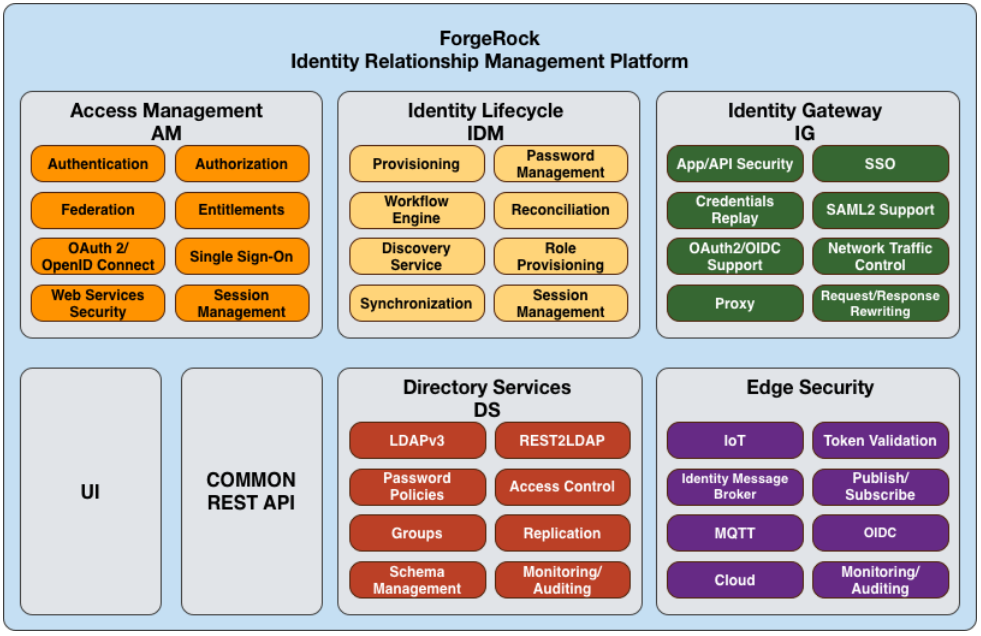
- **ForgeRock Directory Services**. Internet scale directory server. Directory Services provides full LDAP protocol support, multi-protocol access, cross-domain replication, ForgeRock® Common REST framework, SCIM support, and many other features.

- **ForgeRock Identity Gateway**. No touch single sign-on (SSO) to enterprise, legacy, and custom applications. Identity Gateway is a reverse proxy server with specialized session management and credential replay functionality. IG works together with AM to integrate Web applications without needing to modify the target application or the container that it runs in.

- **OpenICF**. Enterprise and cloud identity infrastructure connectors. OpenICF provides identity provisioning connections offering a consistent layer between target resources and applications and exposing a set of programming functions for the full lifecycle of an identity. OpenICF connectors are compatible with OpenIDM, Sun Identity Manager, Oracle® Waveset, Brinqa® GRC Platform, and so forth.

"ForgeRock Identity Platform" illustrates these components:

ForgeRock Identity Platform

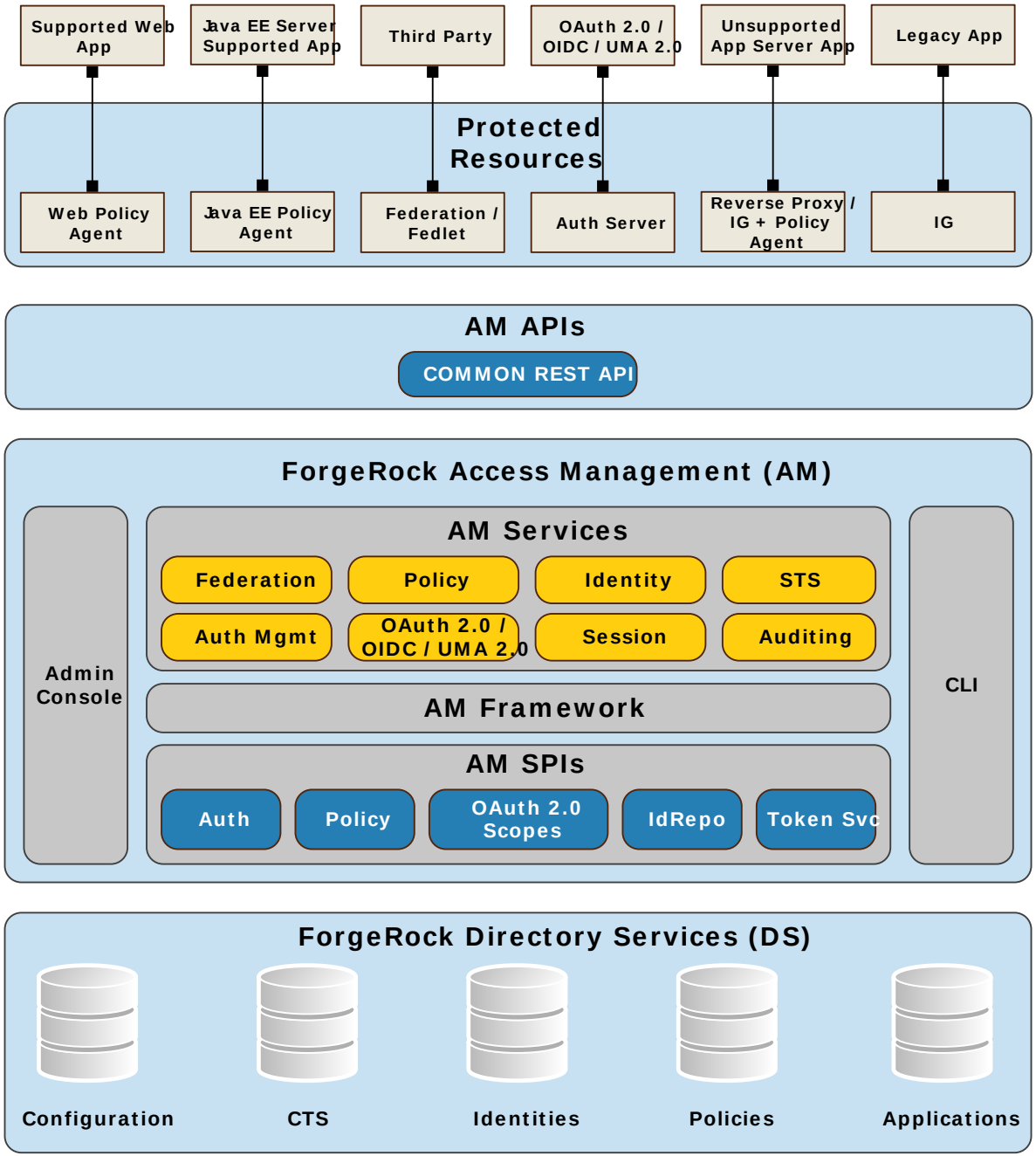


Server Overview

AM is a centralized access management server, securing protected resources across the network and providing authentication, authorization, Web security, and federation services in a single, integrated solution. AM manages access to the protected resources by controlling who has access, when, how long, and under what conditions by centralizing disparate hardware and software services for cloud, enterprise, mobile, and business-to-business (B2B) systems.

"Architecture" illustrates the AM architecture.

Architecture



AM features a highly modular and flexible architecture with multiple plugin points to meet any customer deployment. It leverages industry standard protocols, such as HTTP, XML, SOAP, REST, SAML v2.0, OAuth 2.0, OpenID Connect 1.0, and so forth to deliver a high performance, highly scalable, and highly available access management solution over the network. AM services are 100% Java-based, proven across multiple platforms and containers in many production deployments.

AM core server can be deployed and integrated within existing network infrastructures. AM provides the following distribution files:

Distribution Files

File	Description
`AM-7.1.4.war`	The distribution `.war` file includes the core server code with an embedded DS server. The distribution includes an administrative graphical user interface (GUI) Web console. During installation, the `.war` file accesses properties to obtain the fully qualified domain name, port, context path, and the location of the configuration folder. These properties can be obtained from the `boot.json` file in the AM installation directory, from environment variables, or from a combination of the two. This file is also available to download individually.
`AM-crypto-tool-7.1.4.war`	AM provides a utility with some cryptographic functionality used for creating Docker images. This utility is strictly for future use, and is not currently supported.
`AM-Soap-STS-Server-7.1.4.war`	AM provides a SOAP-based security token service (STS) server that issues tokens based on the WS-Security protocol[a].
`AM-SSOAdminTools-5.1.3.19.zip`	AM provides an `ssoadm` command-line tool that allows administrators to configure and maintain AM as well as create their own configuration scripts. The `zip` distribution file contains binaries, properties file, script templates, and setup scripts for UNIX and windows servers.
`AM-SSOConfiguratorTools-5.1.3.19.zip`	AM provides configuration and upgrade tools for installing and maintaining your server. The `zip` distribution file contains libraries, legal notices, and supported binaries for these configuration tools. Also, you can view example configuration and upgrade properties files that can be used as a template for your deployments.
`Config-Upgrader-7.1.4.zip`	AM provides a configuration file upgrade tool. For more information on converting configuration files for import into AM, see the `README.md` file in the `Config-Upgrader-7.1.4.zip` file.
`Fedlet-7.1.4.zip`	AM provides an AM Fedlet, a light-weight SAML v2.0 service provider. The Fedlet lets you set up a federated deployment without the need of a fully-featured service provider.
`IDPDiscovery-7.1.4.war`	AM provides an IDP Discovery Profile (SAMLv2 binding profile) for its IDP Discovery service. The profile keeps track of the identity providers for each user.

File	Description
`sample-trees-7.1.4.zip`	Clean installs of AM with an embedded data store provide ready-made sample authentication trees to demonstrate how they can be put together. These sample trees are not installed by default on installs of AM with an external configuration store, or if you are upgrading an existing instance of AM. The `sample-trees-7.1.4.zip` file contains the sample trees in JSON files, ready for import by *Amster* command-line interface. For information on importing files by using Amster, see Importing Configuration Data in the *Amster 7.1 User Guide*.
`Truststore-Utility-7.1.4.zip`	AM provides a utility to help with creating a trust store for use with web authentication. See the `readme.md` in the ZIP file for instructions, and "*MFA: Web Authentication (WebAuthn)*" in the *Authentication and Single Sign-On Guide* for more information.

[a]AM also provides REST-based STS service endpoints, which you can directly utilize on the AM server.

The *ForgeRock BackStage download site* hosts downloadable versions of AM, including a `.zip` file with all of the AM components, the `.war` file, AM tools, the configurator, web and Java agents, and documentation. Verify that you review the Software License and Subscription Agreement presented before you download AM files.

ForgeRock offers the services you need to deploy AM commercial builds into production, including training, consulting, and support.

Key Benefits

The goal of AM is to provide secure, low friction access to valued resources while presenting the user with a consistent experience. AM provides excellent security, which is totally transparent to the user.

AM provides the following key benefits to your organization:

- **Enables Solutions for Additional Revenue Streams**. AM provides the tools and components to quickly deploy services to meet customer demand. For example, AM's Federation Services supports quick and easy deployment with existing SAML v2.0, OAuth2, and OpenID Connect systems. For systems that do not support a full SAMLv2 deployment, AM provides a *Fedlet*, a small SAML v2.0 application, which lets service providers quickly add SAML v2.0 support to their Java applications. These solutions open up new possibilities for additional revenue streams.

- **Reduces Operational Cost and Complexity**. AM can function as a hub, leveraging existing identity infrastructures and providing multiple integration paths using its authentication, SSO, and policies to your applications without the complexity of sharing Web access tools and passwords for data exchange. AM decreases the total cost of ownership (TCO) through its operational efficiencies, rapid time-to-market, and high scalability to meet the demands of our market.

- **Improves User Experience**. AM enables users to experience more services using SSO without the need of multiple passwords.

- **Easier Configuration and Management**. AM centralizes the configuration and management of your access management system, allowing easier administration through its console and command-line tools. AM also features a flexible deployment architecture that unifies services through its modular and embeddable components. AM provides a common REST framework and common user interface (UI) model, providing scalable solutions as your customer base increases to the hundreds of millions. AM also allows enterprises to outsource IAM services to system integrators and partners.
- **Increased Compliance**. AM provides an extensive entitlements service, featuring attribute-based access control (ABAC) policies as its main policy framework with features like import/export support to XACML, a policy editor, and REST endpoints for policy management. AM also includes an extensive auditing service to monitor access according to regulatory compliance standards.

History

AM's timeline is summarized as follows:

- In 2001, Sun Microsystems releases iPlanet Directory Server, Access Management Edition.
- In 2003, Sun renames iPlanet Directory Server, Access Management Edition to Sun ONE Identity Server.
- Later in 2003, Sun acquires Waveset.
- In 2004, Sun releases Sun Java Enterprise System. Waveset Lighthouse is renamed to Sun Java System Identity Manager and Sun ONE Identity Server is renamed to Sun Java System Access Manager. Both products are included as components of Sun Java Enterprise System.
- In 2005, Sun announces an open-source project, OpenSSO, based on Sun Java System Access Manager.
- In 2008, Sun releases OpenSSO build 6, a community open-source version, and OpenSSO Enterprise 8.0, a commercial enterprise version.
- In 2009, Sun releases OpenSSO build 7 and 8.
- In January 2010, Sun was acquired by Oracle and development for the OpenSSO products were suspended as Oracle no longer planned to support the product.

In February 2010, a small group of former Sun employees founded ForgeRock to continue OpenSSO support, which was renamed to OpenAM. ForgeRock continued OpenAM's development with the following releases:

- 2010: OpenAM 9.0
- 2011: OpenAM 9.5
- 2012: OpenAM 10 and 10.1

- 2013: OpenAM 11.0
- 2014: OpenAM 11.1, 12.0, and 12.0.1
- 2015: OpenAM 11.0.3 and 12.0.2
- 2016: OpenAM 12.0.3, 12.0.4, 13.0.0, and 13.5.0
- 2017: Access Management 5 and 5.5
- 2018: Access Management 6 and 6.5
- 2020: Access Management 7

ForgeRock continues to develop, enhance, and support the industry-leading AM product to meet the changing and growing demands of the market.

Chapter 2

Planning the Deployment Architecture

Deployment planning is critical to ensuring your AM system is properly implemented within the time frame determined by your requirements. The more thoroughly you plan your deployment, the more solid your configuration will be, and you will meet timelines and milestones while staying within budget.

A deployment plan defines the goals, scope, roles, and responsibilities of key stakeholders, architecture, implementation, and testing of your AM deployment. A good plan ensures that a smooth transition to a new product or service is configured and all possible contingencies are addressed to quickly troubleshoot and solve any issue that may occur during the deployment process. The deployment plan also defines a training schedule for your employees, procedural maintenance plans, and a service plan to support your AM system.

Deployment Planning Considerations

When planning a deployment, you must consider some important questions regarding your system:

+ *What are you protecting?*

  You must determine which applications, resources, and levels of access to protect? Are there plans for additional services, either developed in-house or through future acquisitions that also require protected access?

+ *How many users are supported?*

  It is important to determine the number of users supported in your deployment based on system usage. Once you have determined the number of users, it is important to project future growth.

+ *What are your product service-level agreements?*

  In addition to planning for the growth of your user base, it is important to determine the production service-level agreements (SLAs) that help determine the current load requirements on your system and for future loads. The SLAs help define your scaling and high-availability requirements.

For example, suppose you have 100,000 active users today, and each user has an average of two devices (laptop, phone) that get a session each day. Suppose that you also have 20 protected applications, with each device hitting an average of seven protected resources an average of 1.4 times daily. Let's say that works out to about 200,000 sessions per day with 7 x 1.4 = ~10 updates to each session object. This can result in 200K session creations, 200K session deletions, and 2M session updates.

Now, imagine next year you still have the same number of active users, 100K, but each has an average of three devices (laptop, phone, tablet), and you have added another 20 protected applications. Assume the same average usage per application per device, or even a little less per device. You can see that although the number of users is unchanged, the whole system needs to scale up considerably.

You can scale your deployment using vertical or horizontal scaling. Vertical scaling involves increasing components to a single host server, such as increasing the number of CPUs or increasing heap memory to accommodate a larger session cache or more policies. Horizontal scaling involves adding additional host servers, possibly behind a load balancer, so that the servers can function as a single unit.

+ *What are your high availability requirements?*

High availability refers to your system's ability to operate continuously for a specified length of time. It is important to design your system to prevent single points of failure and for continuous availability. Based on the size of your deployment, you can create an architecture using a single-site configuration. For larger deployments, consider implementing a multi-site configuration with replication.

+ *Which type of clients will be supported?*

The type of client determines the components required for the deployment. For example, applications deployed on a web server require a web agent. Applications deployed in Java containers require a Java agent. An AJAX application can use AM's RESTful API. Legacy or custom applications can use the ForgeRock Identity Gateway. Applications in an unsupported application server can use a reverse proxy with a web or Java agent. Third party applications can use federation or a fedlet, or an OpenID Connect or an OAuth 2.0 component.

+ *What are your SSL/TLS requirements?*

There are two common approaches to handling SSL. First, using SSL through to the application servers themselves, for example, using SSL on the containers. Or second, using SSL offloading via a network device and running HTTP clear internally. You must determine the appropriate approach as each method requires different configurations. Determining SSL use early in the

planning process is vitally *important*, as adding SSL later in the process is more complicated and could result in delays in your deployment.

+ *What are your other security requirements?*

The use of firewalls provides an additional layer of security for your deployment. If you are planning to deploy the AM server behind a firewall, you can deploy a reverse proxy, such as Identity Gateway. For another level of security, consider using multiple DNS infrastructures using zones; one zone for internal clients, another zone for external clients. To provide additional performance, you can deploy the DNS zones behind a load balancer.

Ensure all stakeholders are engaged during the planning phase. This effort includes but is not limited to delivery resources, such as project managers, architects, designers, implementers, testers, and service resources, such as service managers, production transition managers, security, support, and sustaining personnel. Input from all stakeholders ensures all viewpoints are considered at project inception, rather than downstream, when it may be too late.

Deployment Planning Steps

The general deployment planning steps can be summarized as follows:

Project Initiation

The Project Initiation phase begins by defining the overall scope and requirements of the deployment.

+ *Items to Plan*

- Determine the scope, roles and responsibilities of key stakeholders and resources required for the deployment.
- Determine critical path planning including any dependencies and their assigned expectations.
- Run a pilot to test the functionality and features of AM and uncover any possible issues early in the process.
- Determine training for administrators of the environment and training for developers, if needed.

Architecting

The Architecting phase involves designing the deployment.

+ *Items to Plan*

- Determine the use of products, map requirements to features, and ensure the architecture meets the functional requirements.
- Ensure that the architecture is designed for ease of management and scale. TCO is directly proportional to the complexity of the deployment.
- Determine how the Identity, Configuration, and Core Token Service (CTS) data stores are to be configured.
- Determine the sites configuration.
- Determine where SSL is used in the configuration and how to maintain and update the certificate keystore and truststore for AM's components, such as the agent installer, **ssoadm** tool, agent server, and other AM servers. Planning for SSL at this point can avoid more difficulty later in the process.
- Determine if AM will be deployed behind a load balancer with SSL offloading. If this is the case, you must ensure that the load balancer rewrites the protocol during redirection. If you have a web or Java agent behind a load balancer with SSL offloading, ensure that you set the web or Java agent's override request URL properties.
- For multiple AM deployments, there is a requirement to deploy a layer 7 cookie-based load balancer and intelligent keep-alives (for example, `/openam/isAlive.jsp`). The network teams should design the appropriate solution in the architecting phase.
- Determine requirements for vertical scaling, which involves increasing the Java heap based on anticipated session cache, policy cache, federation session, and restricted token usage. Note that vertical scaling could come with performance cost, so this must be planned accordingly.
- Determine requirements for horizontal scaling, which involves adding additional AM servers and load balancers for scalability and availability purposes.
- Determine whether to configure AM to store authentication sessions in the CTS token store, on the client, or in AM's memory:
  - Client-based authentication sessions provide authentication high availability and are easier to deploy in global authentication environments, but the authentication session is held by the client.
  - CTS-based authentication sessions provide high availability and keep the authentication session in your environment, but require consistent and fast replication across the CTS token store deployment.
  - In-memory authentication sessions do not provide authentication high availability, but keep authentication session in your environment.

- Determine whether to configure AM to store sessions in the CTS token store or on the client. Client-based sessions allow for easier horizontal scaling but do not provide equivalent functionality to CTS-based sessions.
- Determine if any coding is required including extensions and plugins. Unless it is absolutely necessary, leverage the product features instead of implementing custom code. AM provides numerous plugin points and REST endpoints.

Implementation

The Implementation phase involves deploying your AM system.

+ *Items to Consider*

- Install and configure the AM server, datastores, and components. For information on installing AM, see the Installation Guide.
- Maintain a record and history of the deployment to maintain consistency across the project.
- Tune AM's JVM, caches, LDAP connection pools, container thread pools, and other items. For information on tuning AM, see "*Tuning Instances*" in the *Maintenance Guide*.
- Tune the DS server. Consider tuning the database back end, replication purge delays, garbage collection, JVM memory, and disk space considerations. For more information, see the DS server documentation.
- Consider implementing separate file systems for both AM and DS, so that you can keep log files on a different disk, separate from data or operational files, to prevent device contention should the log files fill up the file system.

Automation and Continuous Integration

The Automation and Continuous Integration phase involves using tools for testing:

- Set up a continuous integration server, such as Jenkins, to ensure that builds are consistent by running unit tests and publishing Maven artifacts. Perform continuous integration unless your deployment includes no customization.
- Ensure your custom code has unit tests to ensure nothing is broken.

Functional Testing

The Functional Testing phase should test all functionality to deliver the solution without any failures. You must ensure that your customizations and configurations are covered in the test plan.

Non-Functional Testing

The Non-Functional Testing phase tests failover and disaster recovery procedures. Run load testing to determine the demand of the system and measure its responses. You can anticipate peak load conditions during the phase.

Supportability

The supportability phase involves creating the runbook for system administrators including procedures for backup and restores, debugging, change control, and other processes. If you have a ForgeRock Support contract, it ensures everything is in place prior to your deployment.

Preparing Deployment Plans

When you create a good concrete deployment plan, it ensures that a change request process is in place and utilized, which is essential for a successful deployment. This section looks at planning the full deployment process. When you have addressed everything in this section, then you should have a concrete plan for deployment.

Planning Training

Training provides common understanding, vocabulary, and basic skills for those working together on the project. Depending on previous experience with access management and with AM, both internal teams and project partners might need training.

+ *What Types of Training Do Team Members Need?*

- All team members should take at least some training that provides an overview of AM. This helps to ensure a common understanding and vocabulary for those working on the project.
- Team members planning the deployment should take an AM deployment training before finalizing your plans, and ideally before starting to plan your deployment.

  AM not only offers a broad set of features with many choices, but the access management it provides tends to be business critical. AM deployment training pays for itself as it helps you to make the right initial choices to deploy more quickly and successfully.
- Team members involved in designing and developing AM client applications or custom extensions should take training in AM development in order to help them make the right choices. This includes developers customizing the AM UI for your organization.

- Team members who have already had been trained in the past might need to refresh their knowledge if your project deploys newer or significantly changed features, or if they have not worked with AM for some time.

ForgeRock University regularly offers training courses for AM topics, including AM development and deployment. For a current list of available courses, see http://forgerock.com/services/university/.

When you have determined who needs training and the timing of the training during the project, prepare a training schedule based on team member and course availability. Include the scheduled training plans in your deployment project plan.

ForgeRock also offers an accreditation program for partners, offering an in-depth assessment of business and technical skills for each ForgeRock product. This program is open to the partner community and ensures that best practices are followed during the design and deployment phases.

Planning Customization

When you customize AM, you can improve how the software fits your organization. AM customizations can also add complexity to your system as you increase your test load and potentially change components that could affect future upgrades. Therefore, a best practice is to deploy AM with a minimum of customizations.

Most deployments require at least some customization, like skinning end user interfaces for your organization, rather than using the AM defaults. If your deployment is expected to include additional client applications, or custom extensions (authentication modules, policy conditions, and so forth), then have a team member involved in the development help you plan the work. The Getting Started with REST can be useful when scoping a development project.

Although some customizations involve little development work, it can require additional scheduling and coordination with others in your organization. An example is adding support for profile attributes in the identity repository.

The more you customize, the more important it is to test your deployment thoroughly before going into production. Consider each customization as sub-project with its own acceptance criteria, and consider plans for unit testing, automation, and continuous integration. See "Planning Tests" for details.

When you have prepared plans for each customization sub-project, you must account for those plans in your overall deployment project plan. Functional customizations, such as custom authentication modules or policy conditions might need to reach the pilot stage before you can finish an overall pilot implementation.

Planning a Pilot Implementation

Unless you are planning a maintenance upgrade, consider starting with a pilot implementation, which is a long term project that is aligned with customer-specific requirements.

A pilot shows that you can achieve your goals with AM plus whatever customizations and companion software you expect to use. The idea is to demonstrate feasibility by focusing on solving key use cases with minimal expense, but without ignoring real-world constraints. The aim is to fail fast before you have too much invested so that you can resolve any issues that threaten the deployment.

Do not expect the pilot to become the first version of your deployment. Instead, build the pilot as something you can afford to change easily, and to throw away and start over if necessary.

The cost of a pilot should remain low compared to overall project cost. Unless your concern is primarily the scalability of your deployment, you run the pilot on a much smaller scale than the full deployment. Scale back on anything not necessary to validating a key use case.

Smaller scale does not necessarily mean a single-server deployment, though. If you expect your deployment to be highly available, for example, one of your key use cases should be continued smooth operation when part of your deployment becomes unavailable.

The pilot is a chance to try and test features and services before finalizing your plans for deployment. The pilot should come early in your deployment plan, leaving appropriate time to adapt your plans based on the pilot results. Before you can schedule the pilot, team members might need training and you might require prototype versions of functional customizations.

Plan the pilot around the key use cases that you must validate. Make sure to plan the pilot review with stakeholders. You might need to iteratively review pilot results as some stakeholders refine their key use cases based on observations.

Planning Security Hardening

When you first configure AM, there are many options to evaluate, plus a number of ways to further increase levels of security. You must, therefore, plan to secure the deployment as described in Security Guide.

Planning With Providers

AM delegates authentication and profile storage to other services. AM can store configuration, policies, session, and other tokens in an external directory service. AM can also participate in a circle of trust with other SAML entities. In each of these cases, a successful deployment depends on coordination with service providers, potentially outside of your organization.

The infrastructure you need to run AM services might be managed outside your own organization. Hardware, operating systems, network, and software installation might be the responsibility of providers with which you must coordinate.

When working with providers, take the following points into consideration:

- Shared authentication and profile services might have been sized prior to or independently from your access management deployment.

  An overall outcome of your access management deployment might be to decrease the load on shared authentication services (and replace some authentication load with single-sign on that is

managed by AM), or it might be to increase the load (if, for example, your deployment enables many new applications or devices, or enables controlled access to resources that were previously unavailable).

Identity repositories are typically backed by shared directory services. Directory services might need to provision additional attributes for AM. This could affect not only directory schema and access for AM, but also sizing for the directory services that your deployment uses.

- If your deployment uses an external directory service for AM configuration data and AM policies, then the directory administrator must include attributes in the schema and provide access rights to AM. The number of policies depends on the deployment. For deployments with thousands or millions of policies to store, AM's use of the directory could affect sizing.
- If your deployment uses an external directory service as a backing store for the AM Core Token Service (CTS), then the directory administrator must include attributes in the schema and provide access rights to AM.

  CTS load tends to involve more write operations than configuration and policy load, as CTS data tend to be more volatile, especially if most tokens concern short-lived sessions. This can affect directory service sizing.

  CTS enables cross-site session high availability by allowing a remote AM server to retrieve a user session from the directory service backing the CTS. For this feature to work quickly in the event of a failure or network partition, CTS data must be replicated rapidly including across WAN links. This can affect network sizing for the directory service.

  When configured to store sessions in the client, AM does *not* write the sessions to the CTS token store. Instead, AM uses the CTS token store for session blacklists. Session blacklisting is an optional AM feature that provides logout integrity.
- SAML federation circles of trust require organizational and legal coordination before you can determine what the configuration looks like. Organizations must agree on which security data they share and how, and you must be involved to ensure that their expectations map to the security data that is actually available.

  There also needs to be coordination between all SAML parties, (that is, agreed-upon SLAs, patch windows, points of contact and escalation paths). Often, the technical implementation is considered, but not the *business requirements*. For example, a common scenario occurs when a service provider takes down their service for patching without informing the identity provider or vice-versa.
- When working with infrastructure providers, realize that you are likely to have better sizing estimates after you have tried a test deployment under load. Even though you can expect to revise your estimates, take into account the lead time necessary to provide infrastructure services.

  Estimate your infrastructure needs not only for the final deployment, but also for the development, pilot, and testing stages.

For each provider you work with, add the necessary coordinated activities to your overall plan, as well as periodic checks to make sure that parallel work is proceeding according to plan.

Planning Integration With Client Applications

When planning integration with AM client applications, the applications that are most relevant are those that register with AM; therefore, you should make note of the following types of client applications registering with AM:

- + *AM web and Java Agents Reside with the Applications They Protect*

  By default, web and Java agents store their configuration profiles in AM's configuration store. If notifications are enabled, AM sends web and Java agents notifications about configuration changes.

  To delegate administration of multiple web or Java agents, AM lets you create a group profile for each realm to register the agent profiles.

  While the AM administrator manages web or Java agent configuration, application administrators are often the ones who install the agents. You must coordinate installation and upgrades with them.

- + *OAuth 2.0/OpenID Connect 1.0 Clients Register Profiles With AM*

  AM optionally allows registration of such applications without prior authentication. By default, however, registration requires an access token granted to an OAuth 2.0 client with access to register profiles.

  If you expect to allow dynamic registration, or if you have many clients registering with your deployment, then consider clearly documenting how to register the clients, and building a client to register clients.

- + *You Must Configure Circles of Trust for SAML v2.0 Federation*

  Registration happens at configuration time, rather than at runtime.

  Address the necessary configuration as described in "Planning With Providers".

  If your deployment functions as a SAML v2.0 Identity Provider (IDP) and shares Fedlets with Service Providers (SP), the SP administrators must install the Fedlets, and must update their Fedlets for changes in your IDP configuration. Consider at least clearly documenting how to do so, and if necessary, build installation and upgrade capabilities.

- If you have custom client applications, consider how they are configured and how they must register with AM.

- REST API client applications authenticate based on a user profile.

  REST client applications can therefore authenticate using whatever authentication mechanisms you configure in AM, and therefore do not require additional registration.

For each client application whose integration with AM requires coordination, add the relevant tasks to your overall plan.

Planning Integration With Audit Tools

AM and the web or Java agents can log audit information to different formats, such as flat files and relational databases. Log volumes depend on usage and on logging levels. By default, AM generates both access and error messages for each service, providing the raw material for auditing the deployment. For more information about supported audit log formats and the information logged, see "*Setting Up Audit Logging*" in the *Security Guide* and Reference.

In order to analyze the raw material, however, you must use other software, such as Splunk, which indexes machine-generated data for analysis.

If you require integration with an audit tool, plan the tasks of setting up logging to work with the tool, and analyzing and monitoring the data once it has been indexed. Consider how you must retain and rotate log data once it has been consumed, as a high volume service can produce large volumes of log data.

Include these plans in the overall plan.

Planning Tests

In addition to planning tests for each customized component, test the functionality of each service you deploy, such as authentication, policy decisions, and federation. You should also perform non-functional testing to validate that the services hold up under load in realistic conditions. Perform penetration testing to check for security issues. Include acceptance tests for the actual deployment. The data from the acceptance tests help you to make an informed decision about whether to go ahead with the deployment or to roll back.

+ *Planning Functional Testing*

  Functional testing validates that specified test cases work with the software considered as a black box.

  As ForgeRock already tests AM and the web and Java agents functionally, focus your functional testing on customizations and service-level functions. For each key service, devise automated functional tests. Automated tests make it easier to integrate new deliveries to take advantage of recent bug fixes and to check that fixes and new features do not cause regressions.

  Tools for running functional testing include Apache JMeter and Selenium. Apache JMeter is a load testing tool for Web applications. Selenium is a test framework for Web applications, particularly for UIs.

As part of the overall plan, include not only tasks to develop and maintain your functional tests, but also to provision and to maintain a test environment in which you run the functional tests before you significantly change anything in your deployment. For example, run functional tests whenever you upgrade AM, AM web and Java agents, or any custom components, and analyze the output to understand the effect on your deployment.

+ *Planning Service Performance Testing*

For written service-level agreements and objectives, even if your first version consists of guesses, you turn performance plans from an open-ended project to a clear set of measurable goals for a manageable project with a definite outcome. Therefore, start your testing with clear definitions of success.

Also, start your testing with a system for load generation that can reproduce the traffic you expect in production, and provider services that behave as you expect in production. To run your tests, you must therefore generate representative load data and test clients based on what you expect in production. You can then use the load generation system to perform iterative performance testing.

Iterative performance testing consists in identifying underperformance and the bottlenecks that cause it, and discovering ways to eliminate or work around those bottlenecks. Underperformance means that the system under load does not meet service level objectives. Sometimes re-sizing and/or tuning the system or provider services can help remove bottlenecks that cause underperformance.

Based on service level objectives and availability requirements, define acceptance criteria for performance testing, and iterate until you have eliminated underperformance.

Tools for running performance testing include Apache JMeter, for which your loads should mimic what you expect in production, and Gatling, which records load using a domain specific language for load testing. To mimic the production load, examine both the access patterns and also the data that AM stores. The representative load should reflect the expected random distribution of client access, so that sessions are affected as in production. Consider authentication, authorization, logout, and session timeout events, and the lifecycle you expect to see in production.

Although you cannot use actual production data for testing, you can generate similar test data using tools, such as the DS **makeldif** command, which generates user profile data for directory services. AM REST APIs can help with test provisioning for policies, users, and groups.

As part of the overall plan, include not only tasks to develop and maintain performance tests, but also to provision and to maintain a pre-production test environment that mimics your production environment. Security measures in your test environment must also mimic your production environment, as changes to secure AM as described in "Planning Security Hardening", such as using HTTPS rather than HTTP, can impact performance.

Once you are satisfied that the baseline performance is acceptable, run performance tests again when something in your deployment changes significantly with respect to performance. For example, if the load or number of clients changes significantly, it could cause the system to

underperform. Also, consider the thresholds that you can monitor in the production system to estimate when your system might start to underperform.

+ *Planning Penetration Testing*

Penetration testing involves attacking a system to expose security issues before they show up in production.

When planning penetration testing, consider both white box and black box scenarios. Attackers can know something about how AM works internally, and not only how it works from the outside. Also, consider both internal attacks from within your organization, and external attacks from outside your organization.

As for other testing, take time to define acceptance criteria. Know that ForgeRock has performed penetration testing on the software for each enterprise release. Any customization, however, could be the source of security weaknesses, as could configuration to secure AM.

You can also plan to perform penetration tests against the same hardened, pre-production test environment also used for performance testing.

+ *Planning Deployment Testing*

Deployment testing is used as a description, and not a term in the context of this guide. It refers to the testing implemented within the deployment window after the system is deployed to the production environment, but before client applications and users access the system.

Plan for minimal changes between the pre-production test environment and the actual production environment. Then test that those changes have not cause any issues, and that the system generally behaves as expected.

Take the time to agree upfront with stakeholders regarding the acceptance criteria for deployment tests. When the production deployment window is small, and you have only a short time to deploy and test the deployment, you must trade off thorough testing for adequate testing. Make sure to plan enough time in the deployment window for performing the necessary tests and checks.

Include preparation for this exercise in your overall plan, as well as time to check the plans close to the deployment date.

Planning Documentation and Tracking Changes

The AM product documentation is written for readers like you, who are architects and solution developers, as well as for AM developers and for administrators who have had AM training. The people operating your production environment need concrete documentation specific to your deployed solution, with an emphasis on operational policies and procedures.

Procedural documentation can take the form of a runbook with procedures that emphasize maintenance operations, such as backup, restore, monitoring and log maintenance, collecting data pertaining to an issue in production, replacing a broken server or web or Java agent, responding to a monitoring alert, and so forth. Make sure in particular that you document procedures for taking remedial action in the event of a production issue.

Furthermore, to ensure that everyone understands your deployment and to speed problem resolution in the event of an issue, changes in production must be documented and tracked as a matter of course. When you make changes, always prepare to roll back to the previous state if the change does not perform as expected.

Include documentation tasks in your overall plan. Also, include the tasks necessary to put in place and to maintain change control for updates to the configuration.

Planning Maintenance and Support in Production

If you own the architecture and planning, but others own the service in production, or even in the labs, then you must plan coordination with those who own the service.

Start by considering the service owners' acceptance criteria. If they have defined support readiness acceptance criteria, you can start with their acceptance criteria. You can also ask yourself the following questions:

- What do they require in terms of training in AM?
- What additional training do they require to support your solution?
- Do your plans for documentation and change control, as described in "Planning Documentation and Tracking Changes", match their requirements?
- Do they have any additional acceptance criteria for deployment tests, as described in Planning Deployment Testing?

Also, plan back line support with ForgeRock or a qualified partner. The aim is to define clearly who handles production issues, and how production issues are escalated to a product specialist if necessary.

Include a task in the overall plan to define the hand off to production, making sure there is clarity on who handles monitoring and issues.

Planning Rollout Into Production

In addition to planning for the hand off of the production system, also prepare plans to roll-out the system into production. Rollout into production calls for a well-choreographed operation, so these are likely the most detailed plans.

Take at least the following items into account when planning the rollout:

- Availability of all infrastructure that AM depends upon the following elements:
  - Server hosts and operating systems
  - Web application containers
  - Network links and configurations
  - Load balancers
  - Reverse proxy services to protect AM
  - Data stores, such as directory services
  - Authentication providers
- Installation for all AM services.
- Installation of AM client applications:
  - Web and Java agents
  - Fedlets
  - SDK applications
  - OAuth 2.0 applications
  - OpenID Connect 1.0 applications
- Final tests and checks.
- Availability of the personnel involved in the rollout.

In your overall plan, leave time and resources to finalize rollout plans toward the end of the project.

Planning for Growth

Before rolling out into production, plan how to monitor the system to know when you must grow, and plan the actions to take when you must add capacity.

Unless your deployment is very constrained, after your successful rollout of access management services, you can expect to add capacity at some point in the future. Therefore, you should plan to monitor system growth.

You can grow many parts of the system by adding servers or adding clients. The parts of the system that you cannot expand so simply are those parts that depend on writing to the directory service.

The directory service eventually replicates each write to all other servers. Therefore, adding servers simply adds the number of writes to perform. One simple way of getting around this limitation is to

split a monolithic directory service into several directory services. That said, directory services often are not a bottleneck for growth.

When should you expand the deployed system? The time to expand the deployed system is when growth in usage causes the system to approach performance threshold levels that cause the service to underperform. For that reason, devise thresholds that can be monitored in production, and plan to monitor the deployment with respect to the thresholds. In addition to programming appropriate alerts to react to thresholds, also plan periodic reviews of system performance to uncover anything missing from regular monitoring results.

Planning for Upgrades

In this section, "upgrade" means moving to a more recent release, whether it is a patch, maintenance release, minor release, or major release. For definitions of the types of release, refer to Interface stability.

Upgrades generally bring fixes, or new features, or both. For each upgrade, you must build a new plan. Depending on the scope of the upgrade, that plan might include almost all of the original overall plan, or it might be abbreviated, for example, for a patch that fixes a single issue. In any case, adapt deployment plans, as each upgrade is a new deployment.

When planning an upgrade, pay particular attention to testing and to any changes necessary in your customizations. For testing, consider compatibility issues when not all agents and services are upgraded simultaneously. Choreography is particularly important, as upgrades are likely to happen in constrained low usage windows, and as users already have expectations about how the service should behave.

When preparing your overall plan, include a regular review task to determine whether to upgrade, not only for patches or regular maintenance releases, but also to consider whether to upgrade to new minor and major releases.

Planning for Disaster Recovery

Disaster recovery planning and a robust backup strategy is essential when server hardware fails, network connections go down, a site fails, and so on. Your team must determine the disaster recovery procedures to recover from such events.

Chapter 3

Deployment Configuration Locations

Every AM deployment has associated *configuration* data. Configuration data consists of properties and settings used by the AM instance to function.

Configuration data is often referred to as "*static*", because after your instance is configured to your requirements, it does not need to be changed.

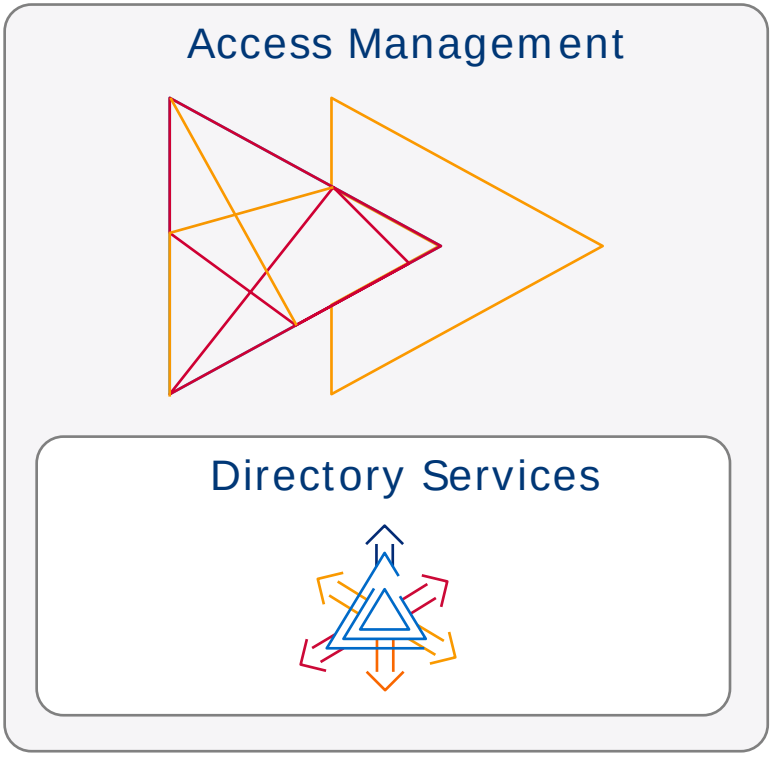
Configuration data includes properties and settings for the following:

- Global services
- Realms
- Authentication trees

Configuration data can be stored is the following places, each tailored to particular deployment requirements:

Embedded DS Instance

For evaluation deployments, you can store all your configuration data in the single, embedded DS server that is included in AM.



Important

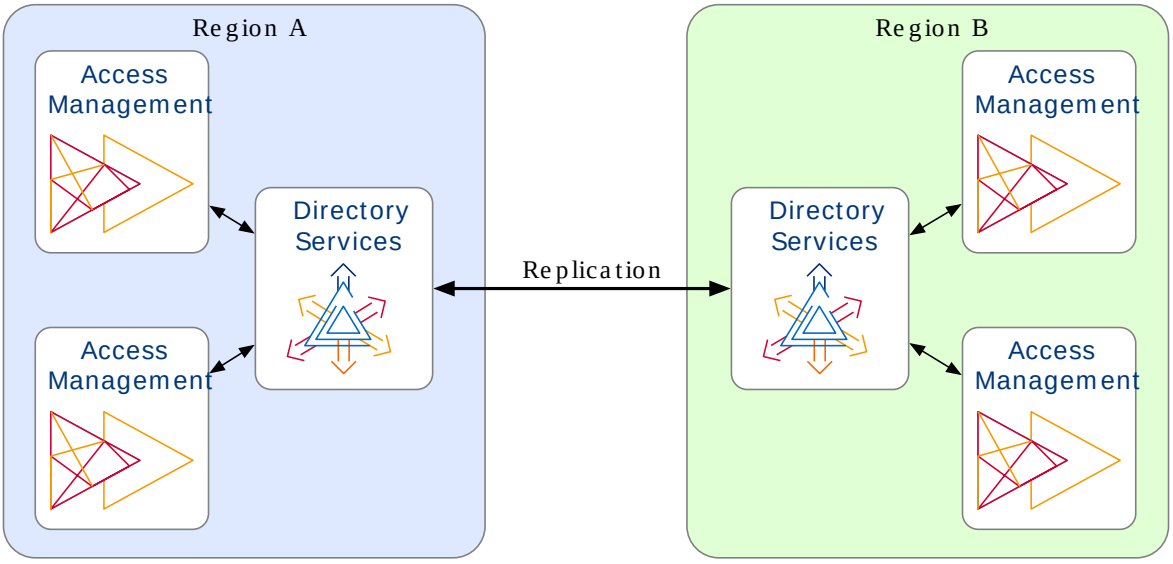
Use of embedded DS servers for production deployments is not supported in AM 7.

For information on installing an AM instance for evaluation, see the Evaluation Guide.

External DS Instances

Storing configuration data in external DS data stores offers deployment flexibility and provides instance high availability.

Configuration data in the DS instances is shared between the AM instances in your deployment. The configuration data can be replicated between multiple DS instances in a cluster, and made available to AM instances in different regions, improving availability, and data integrity.



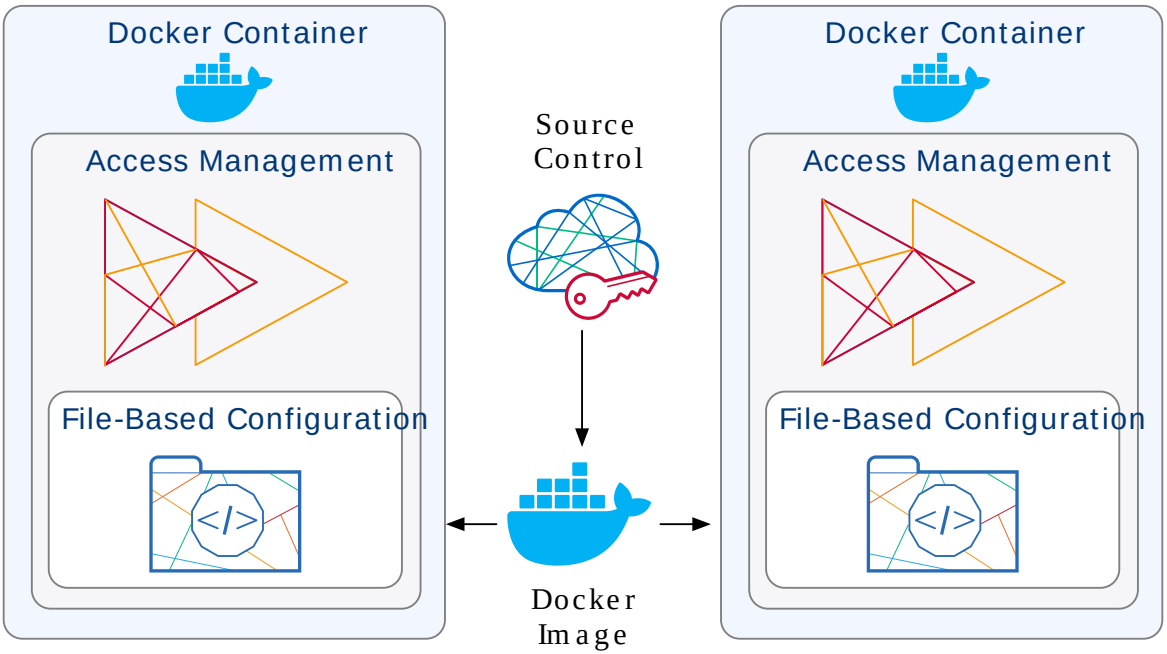
For information on installing AM instances with external data stores, see the Installation Guide.

Files

File-based configuration is used specifically for automated cloud deployments, and only available and supported when used within Docker images.

AM's static configuration data is written to files in the file system, and checked into a source control system, such as Git.

AM instances are created as Docker images, with the file-based configuration incorporated in the image.



You can insert variables into these configuration files before checking them into source control. These variables are substituted with the appropriate values at runtime, when starting up an image, so that the same base configuration files can be reused for multiple instances, and different staging; for example, development, QA, or pre-production, and then promoted to production.

For information on installing AM instances with Kubernetes, see the *Forge*Rock Dev*Ops* (ForgeOps) documentation.

AM instances also create dynamic, run-time data. This data can change and grow often, even in a production instance, as business logic changes.

Dynamic data includes properties, settings, and values for the following:

- Policies, policy sets, and resource types
- OAuth 2.0 client profiles
- Federation entities
- Core Token Service (CTS) tokens
- UMA resources, labels, audit messages, and pending requests

Dynamic data is stored in one or more DS instances. You can choose to store dynamic data alongside the configuration data, or separate it into different data stores.

How you separate dynamic data into data stores will depend on the amount dynamic data you expect to handle. For example, CTS data is often highly volatile and short lived, so it warrants its own set of tuned DS instances. The other dynamic data types may not be so volatile, and could potentially all share a set of differently tuned DS instances.

For information on setting up external stores for use with AM, see "*Preparing External Stores*" in the *Installation Guide*

Chapter 4

Example Deployment Topology

You can configure AM in a wide variety of deployments depending on your security requirements and network infrastructure. This chapter presents an example enterprise deployment, featuring a highly available and scalable architecture across multiple data centers.

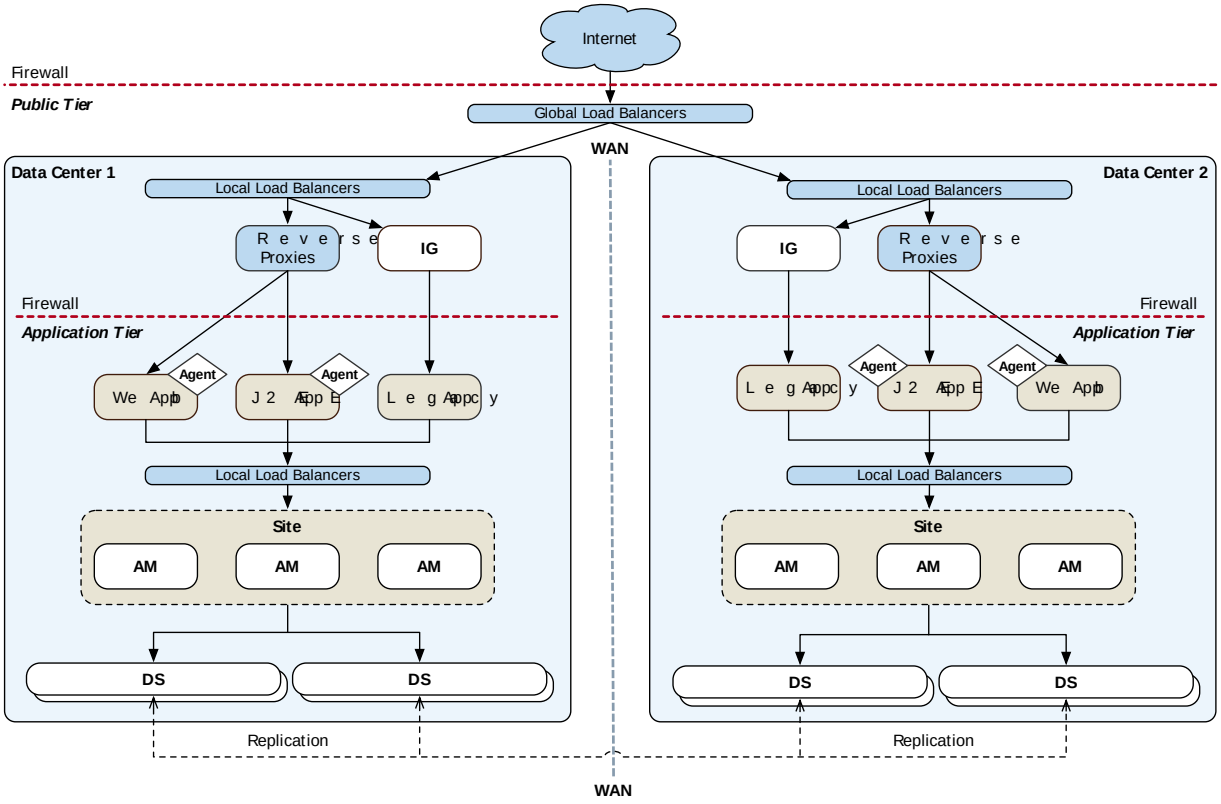
"Deployment Example" presents an example topology of a multi-city multi-data-center deployment across a wide area network (WAN). The example deployment is partitioned into a two-tier architecture. The top tier is a DMZ with the initial firewall securing public traffic into the network. The second firewall limits traffic from the DMZ into the application tier where the protected resources are housed.

The example components in this chapter are presented for illustrative purposes. ForgeRock does not recommend specific products, such as reverse proxies, load balancers, switches, firewalls, and so forth, as AM can be deployed within your existing networking infrastructure.

Tip

For an example showing a ForgeRock Identity Platform deployment on physical hardware, see the ForgeRock Identity Platform Setup Guide.

Deployment Example



See the following sections for more information about the different actors in the example:

The Public Tier

The public tier provides an extra layer of security with a DMZ consisting of load balancers and reverse proxies. This section presents the DMZ elements:

+ *The Global Load Balancer*

The example deployment uses a global load balancer (GLB) to route DNS requests efficiently to multiple data centers. The GLB reduces application latency by spreading the traffic workload among data centers and maintains high availability during planned or unplanned down time, during which it quickly re-routes requests to another data center to ensure online business activity continues successfully.

You can install a cloud-based or a hardware-based version of the GLB. The leading GLB vendors offer solutions with extensive health-checking, site affinity capabilities, and other features for most systems. Detailed deployment discussions about global load balancers are beyond the scope of this guide.

+ *Front End Local Load Balancers*

Each data center has local front end load balancers to route incoming traffic to multiple reverse proxy servers, thereby distributing the load based on a scheduling algorithm. Many load balancer solutions provide server affinity or stickiness to efficiently route a client's inbound requests to the same server. Other features include health checking to determine the state of its connected servers, and SSL offloading to secure communication with the client.

You can cluster the load balancers themselves or configure load balancing in a clustered server environment, which provides data and session high availability across multiple nodes. Clustering also allows horizontal scaling for future growth. Many vendors offer hardware and software solutions for this requirement. In most cases, you must determine how you want to configure your load balancers, for example, in an active-passive configuration that supports high availability, or in an active-active configuration that supports session high availability.

There are many load balancer solutions available in the market. You can set up an external network hardware load balancer, or a software solution like HAProxy (L4 or L7 load balancing) or Linux Virtual Server (LVS) (L4 load balancing), and many others.

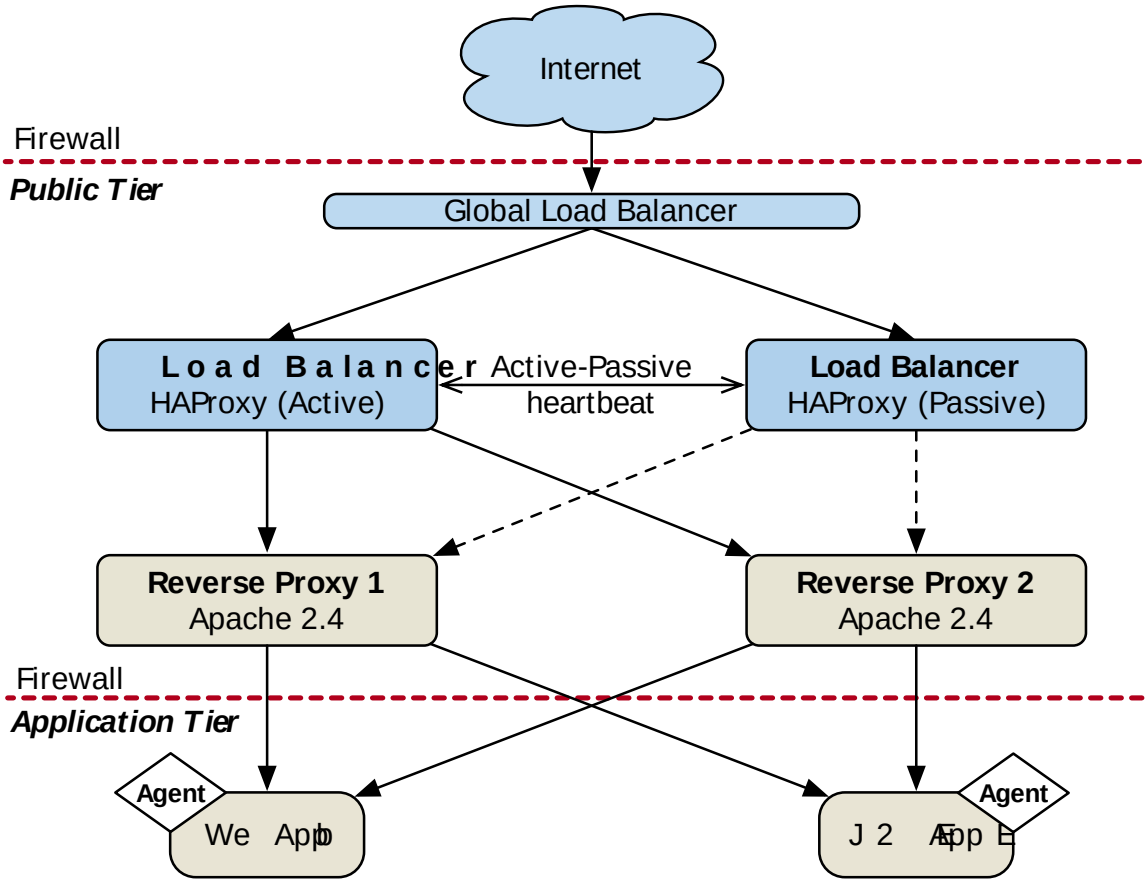
+ *Reverse Proxies*

The reverse proxies work in concert with the load balancers to route the client requests to the back end Web or application servers, providing an extra level of security for your network. The reverse proxies also provide additional features, like caching to reduce the load on the Web servers, HTTP compression for faster transmission, URL filtering to deny access to certain sites, SSL acceleration to offload public key encryption in SSL handshakes to a hardware accelerator, or SSL termination to reduce the SSL encryption overhead on the load-balanced servers.

The use of reverse proxies has several key advantages. First, the reverse proxies serve as an highly scalable SSL layer that can be deployed inexpensively using freely available products, like Apache HTTP server or nginx. This layer provides SSL termination and offloads SSL processing to the reverse proxies instead of the load balancer, which could otherwise become a bottleneck if the load balancer is required to handle increasing SSL traffic.

"Frontend Load Balancer Reverse Proxy Layer" illustrates one possible deployment using HAProxy in an active-passive configuration for high availability. The HAProxy load balancers forward the requests to Apache 2.2 reverse proxy servers. For this example, we assume SSL is configured everywhere within the network.

Frontend Load Balancer Reverse Proxy Layer

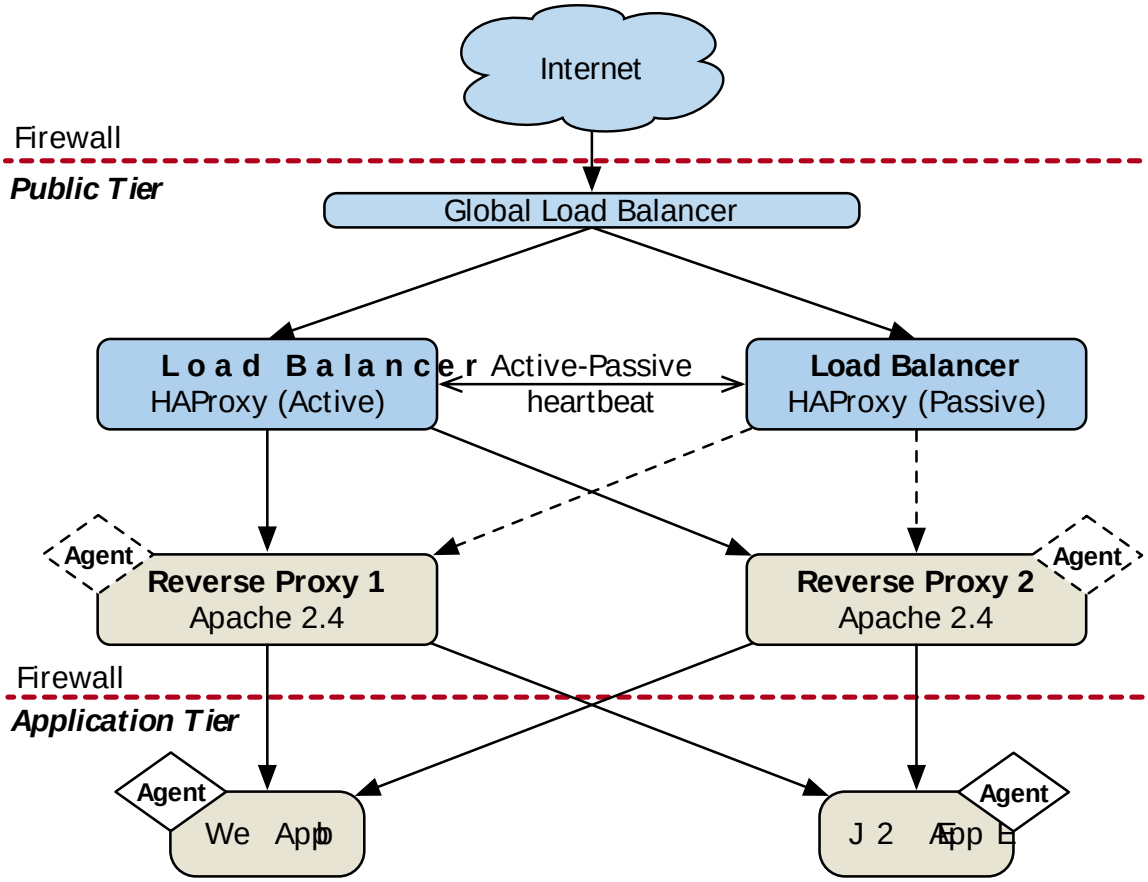


Another advantage to reverse proxies is that they allow access to only those endpoints required for your applications. If you need the authentication user interface and OAuth2/OpenID Connect endpoints, then you can expose only those endpoints and no others. A good rule of thumb is to check which functionality is required for your public interface and then use the reverse proxy to expose only those endpoints.

A third advantage to reverse proxies is when you have applications that sit on non-standard containers for which ForgeRock does not provide a native agent. In this case, you can implement a reverse proxy in your web tier, and deploy a web or Java agent on the reverse proxy to filter any requests.

"Frontend Load Balancers and Reverse Proxies with Agent" shows a simple topology diagram of your Web tier with agents deployed on your reverse proxies. The dotted agents indicate that they can be optionally deployed in your network depending on your configuration, container type, and application.

Frontend Load Balancers and Reverse Proxies with Agent



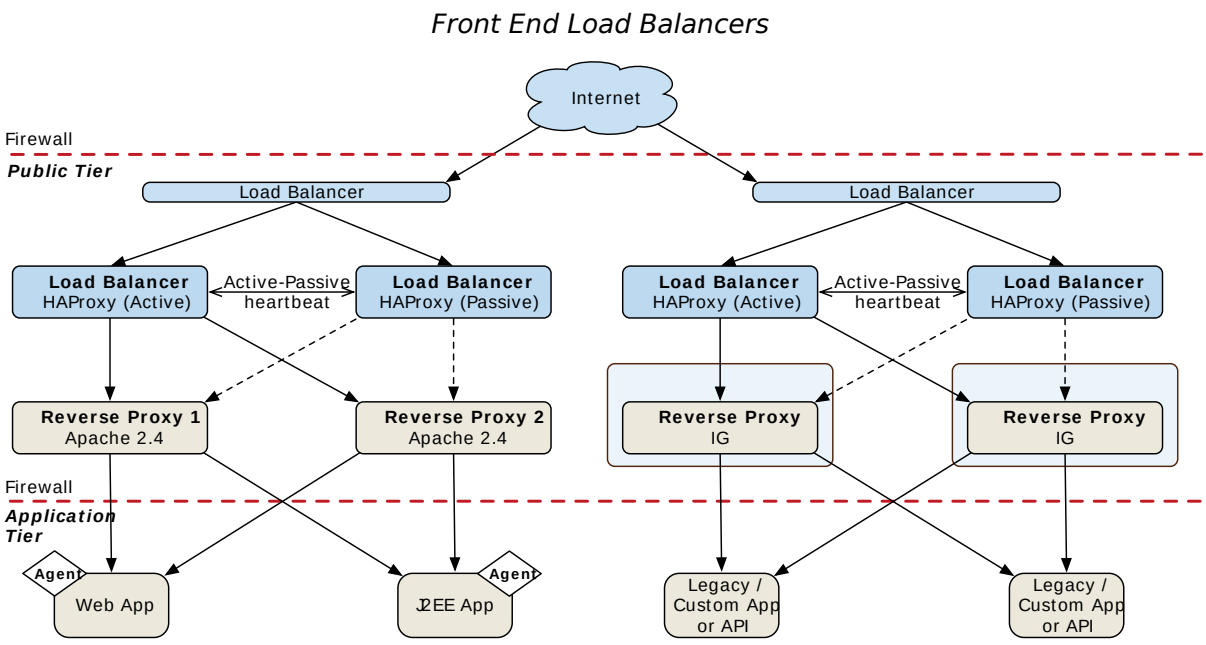
+ *Identity Gateway*

ForgeRock Identity Gateway is a specialized reverse proxy that allows you to secure web applications and APIs, and integrate your applications with identity and access management.

IG extends AM's authentication and authorization services to provide SSO and API security across mobile applications, social applications, partner applications, and web applications. When used in conjunction with AM, IG intercepts HTTP requests and responses, enforces authentication and

authorization, and provides throttling, auditing, password replay, and redaction or enrichment of messages.

IG runs as a standalone Java application, and can be deployed inside a Docker container.



Note

Some authentication modules may require additional user information to authenticate, such as the IP address where the request originated. When AM is accessed through a load balancer or proxy layer, you can configure AM to consume and forward this information with the request headers.

+ *SSL Termination*

One important security decision ia whether to terminate SSL or offload your SSL connections at the load balancer. Offloading SSL effectively decrypts your SSL traffic before passing it on as HTTP or at the reverse proxy. Another option is to run SSL pass-through where the load balancer does not decrypt the traffic but passes it on to the reverse proxy servers, which are

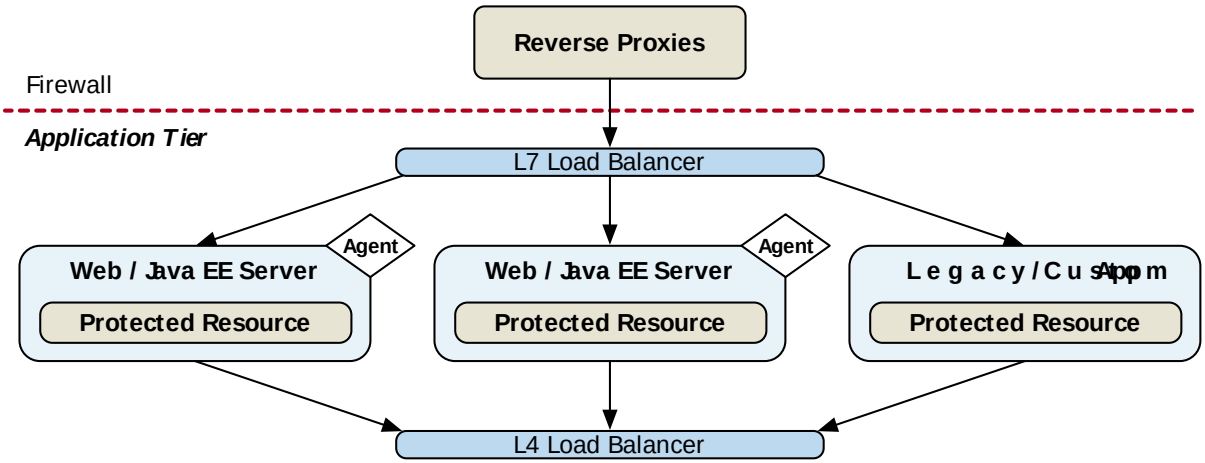
responsible for the decryption. The other option is to deploy a more secure environment using SSL everywhere within your deployment.

The Application Tier

The application tier is where the protected resources reside on Web containers, application servers, or legacy servers. AM web and Java agents intercept all access requests to protected resources on the web servers and grant access to the user based on AM policy decisions. For a list of supported web servers, refer to Application containers.

Because AM is Java-based, you can install the server on a variety of platforms, such as Linux, Solaris, and Windows. For a list of platforms that AM has been tested on, refer to Operating systems.

App Server Deployment



+ *How Does It Work?*

When the client sends an access request to a resource, the web or Java agent redirects the client to an authentication login page. Upon successful authentication, the web or Java agent forwards the request via the load balancer to one of the AM servers.

In AM deployments storing sessions in the CTS token store, the AM server that satisfies the request maintains the session in its in-memory cache to improve performance. If a request for the same user is sent to another AM server, that server must retrieve the session from the CTS token store, incurring a performance overhead.

Client-based sessions are held by the client and passed to AM on each request. Client-based sessions should be signed and encrypted for security reasons, but decrypting the session may an expensive operation for AM to perform on each request depending on the signing and/or encryption algorithms. To improve performance, the decrypt sequence is cached in AM's memory. If a request for the same user is sent to another AM server, that server must decrypt the session again, incurring a performance overhead.

Therefore, even if sticky load balancing is not a requirement when deploying AM, it is recommended for performance.

CTS-based and client-based authentication sessions and client-based authentication sessions share the same characteristics as their CTS-based and client-based session counterparts. Therefore, their performance also benefits from sticky load balancing.

In-memory authentication sessions, however, require sticky load balancing to ensure the same AM server handles the authentication flow for a user. If a request is sent to a different AM server, the authentication flow will start anew.

AM provides a cookie (default: `amlbcoookie`) for sticky load balancing to ensure that the load balancer optimally routes requests to the AM servers. The load balancer inspects the cookie to determine which AM server should receive the request. This ensures that all subsequent requests for a session or authentication session are routed to the same server.

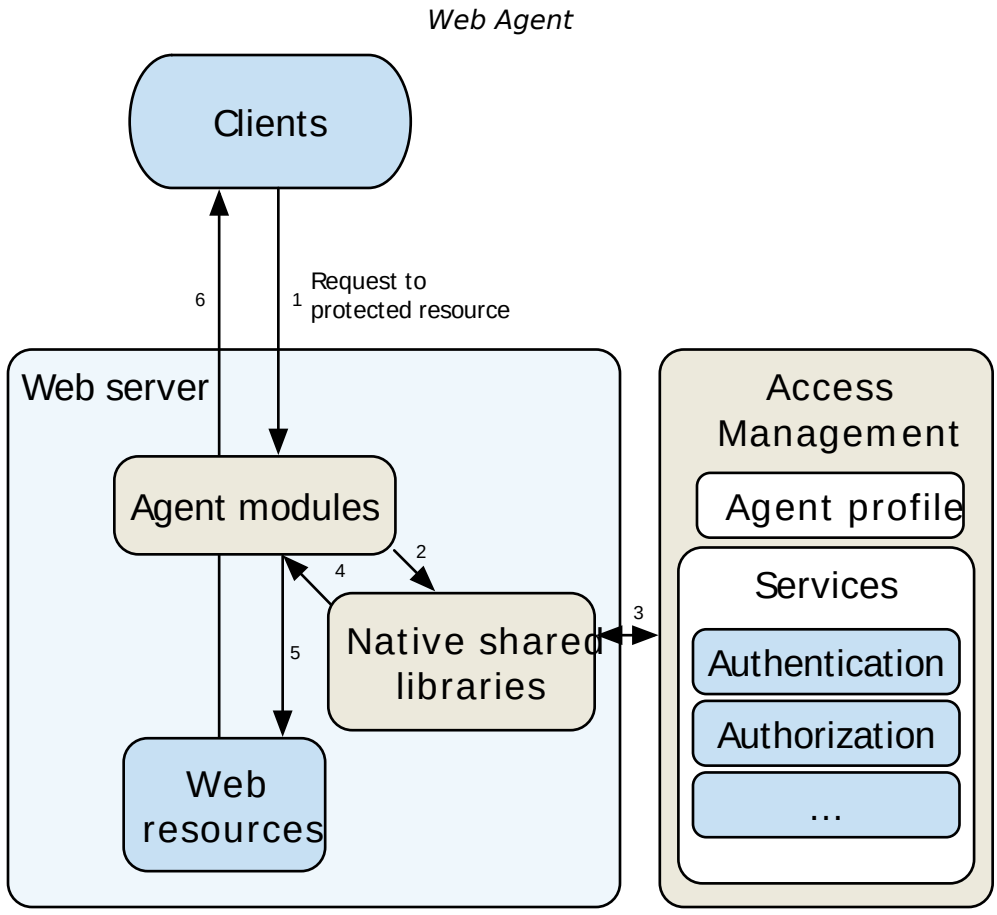
Access Management Agents

ForgeRock Access Management agents are components installed on web servers or Java containers that protect resources, such as websites and applications. Interacting with AM, web and Java agents ensure that inbound requests to protected resources are authenticated and authorized.

AM provides two agents:

- **Web Agent**. Comprised of agent modules tailored to each web server and several native shared libraries. Configure the web agent in the web server's main configuration file.

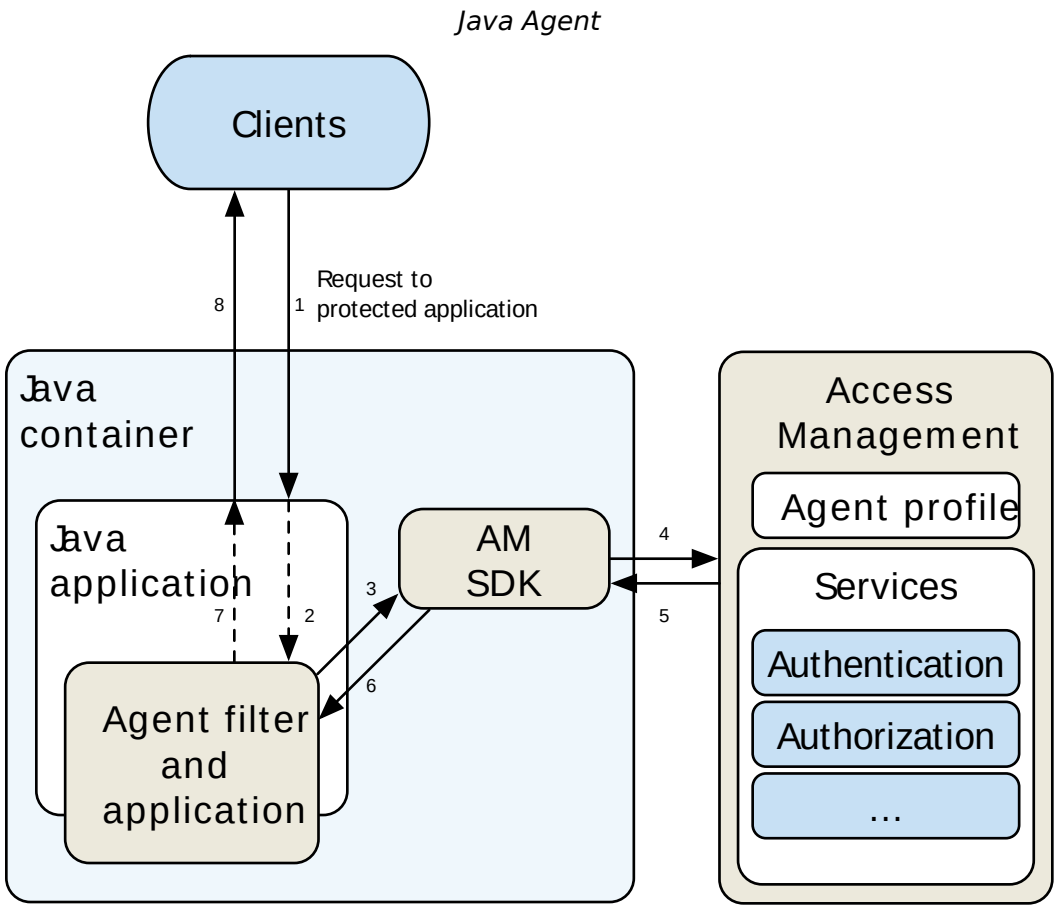
  + *Web Agents Simplified Flow*



For a detailed explanation of the flow, refer to the *Web Agents documentation*.

- **Java Agent**. Comprised of an agent filter, an agent application, and the AM SDK libraries.

  + *Java Agents Simplified Flow*



The figure represents the agent filter (configured in the protected Java application) and the agent application (deployed in the Java container) in the same box as a simplification.

For a detailed explanation of the flow, refer to the *Java Agents documentation*.

Web and Java agents provide the following capabilities, among others:

- **Cookie Reset**. Web and Java agents can reset any number of cookies in the session before the client is redirected for authentication. Reset cookies when the agent is deployed with a parallel authentication mechanism and when cookies need to be reset between mechanisms.

- **Authentication-Only Mode**. Instead of enforcing both authentication and resource-based policy evaluation, web and Java agents can enforce authentication only. Use the authentication-only mode

when there is no need for fine-grain authorization to particular resources, or when you can provide authorization by different means.

- **Not-Enforced Lists**. Web and Java agents can bypass authentication and authorization and grant immediate access to specific resources or client IP addresses. Use not-enforced lists to configure URL and URI lists of resources that does not require protection, or client IP lists for IPs that do not require authentication or authorization to access specific resources.

- **URL Checking and Correction**. Web and Java agents require that clients accessing protected resources use valid URLs with fully qualified domain names (FQDNs). If invalid URLs are referenced, policy evaluation can fail as the FQDN will not match the requested URL, leading to blocked access to the resource. Misconfigured URLs can also result in incorrect policy evaluation for subsequent access requests. Use FQDN checking and correction when clients may specify a resource URL that differs from the FQDN configured in AM's policies, for example, in environments with load balancers and virtual hosts.

- **Attribute Injection**. Web and Java agents can inject user profile attributes into cookies, requests, and HTTP headers. Use attribute injection, for example, with websites that address the user by the name retrieved from the user profile.

- **Notifications**. AM can notify web and Java agents about configuration and session state changes. Notifications affect the validity of the web or Java agent caches, for example, requesting the agent to drop the policy and session cache after a change to policy configuration.

- **Cross-Domain Single Sign-On (CDSSO)**. Web and Java agents can be configured to provide cross-domain single sign-on capabilities. Configure CDSSO when the web or Java agents and the AM instances are in different DNS domains.

- **POST Data Preservation**. Web and Java agents can preserve HTML form data posted to a protected resource by an unauthenticated client. Upon successful authentication, the agent recovers the data stored in the cache and auto-submits it to the protected resource. Use POST data preservation, for example, when users or clients submit large amounts of data, such as blog posts and wiki pages, and their sessions are short-lived.

- **Continuous Security**. Web and Java agents can collect inbound login requests' cookie and header information which an AM server-side authorization script can then process. Use continuous security to configure AM to act upon specific headers or cookies during the authorization process.

- **Conditional Redirection**. Web and Java agents can redirect users to specific AM instances, AM sites, or websites other than AM based on the incoming request URL. Configure conditional redirection login and logout URLs when you want to have fine-grained control over the login or logout process for specific inbound requests.

For more information about the capabilities, see the *ForgeRock Web Agents User Guide* and the *ForgeRock Java Agents User Guide*.

Sites

AM provides the capability to logically group two or more redundant AM servers into a *site*, allowing the servers to function as a single unit identified by a site ID across a LAN or WAN. When you set up a single site, you place the AM servers behind a load balancer to spread the load and provide system failover should one of the servers go down for any reason. You can use round-robin or load average for your load balancing algorithms.

Note

Round-robin load balancing should only be used for a first time access of AM or if the `amlbcookie` is not set; otherwise, cookie-based load balancing should be used.

In AM deployments with CTS-based sessions, the set of servers comprising a site provides uninterrupted service. CTS-based sessions are shared among all servers in a site. If one of the AM servers goes down, other servers in the site read the user session data from the CTS token store, allowing the user to run new transactions or requests without re-authenticating to the system. The same is true for CTS-based authentication sessions; if one of the AM servers becomes unavailable while authenticating a user, any other server in the site can read the authentication session data from the CTS token store and continue with the authentication flow.

AM provides uninterrupted session availability if all servers in a site use the same CTS token store, which is replicated across all servers. For more information, see Core Token Service Guide (CTS).

AM deployments configured for client-based sessions do not use the CTS token store for session storage and retrieval to make sessions highly available. Instead, sessions are stored in HTTP cookies on clients. The same is true for client-based authentication sessions.

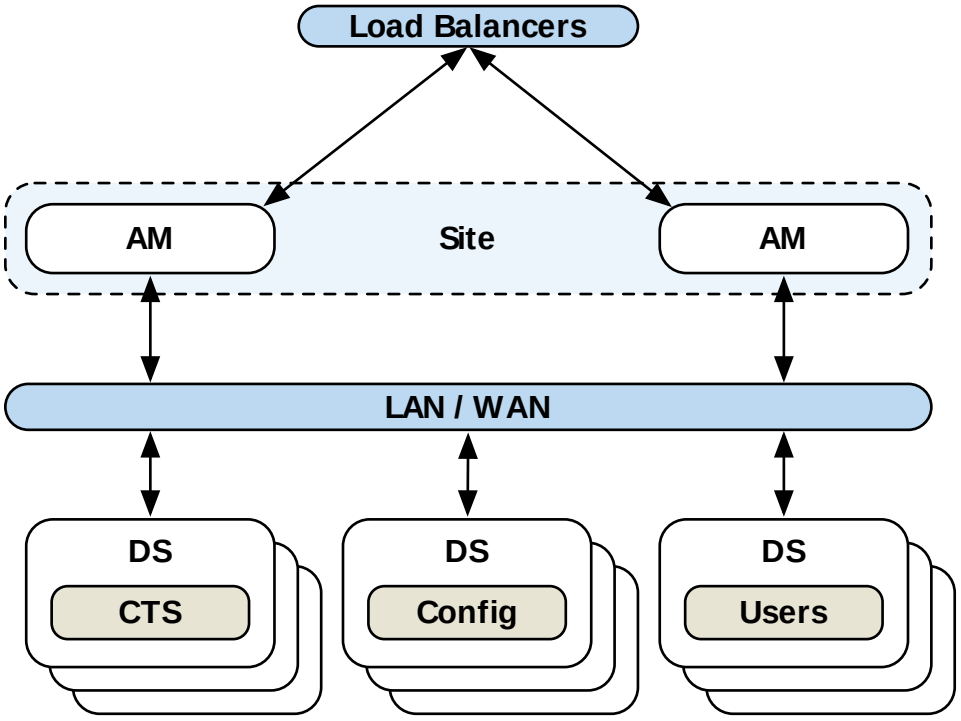
Authentication sessions stored in AM's memory are not highly available. If the AM server authenticating a user becomes unavailable during the authentication flow, the user needs to start the authentication flow again on a different server.

Site deployment examples:

Routing and Load Balancing on the AM Servers

The following picture shows a possible implementation using Linux servers with AM and routing software, like Keepalived, installed on each server. If you require L7 load balancing, you can consider many other software and hardware solutions. AM relies on DS's SDK for load balancing, failover, and heartbeat capabilities to spread the load across the directory servers or to throttle performance.

Application Tier Deployment



Note

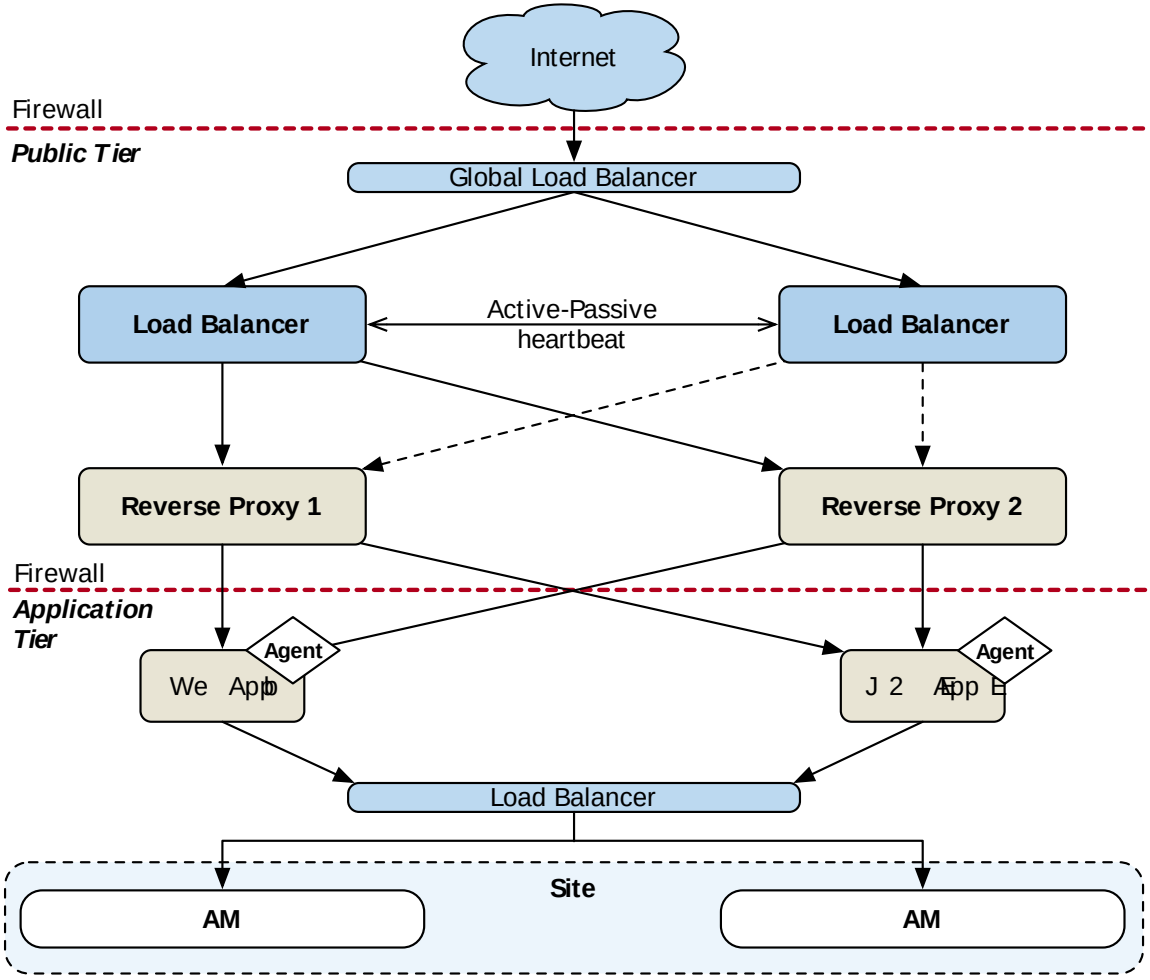
When protecting AM with a load balancer or proxy service, configure your container so that AM can trust the load balancer or proxy service.

Single Load Balancer Deployment

You can also set up a load balancer with multiple AM servers. You configure the load balancer to be sticky using the value of the AM cookie, `amlbcookie`, which routes client requests to that primary server. If the primary AM server goes down for any reason, it fails over to another AM server. Session data also continues uninterrupted if a server goes down as it is shared between AM servers. You must also ensure that the container trusts the load balancer.

You must determine if SSL should be terminated on the load balancer or communication be encrypted from the load balancer to the AM servers.

Site Deployment With a Single Load Balancer

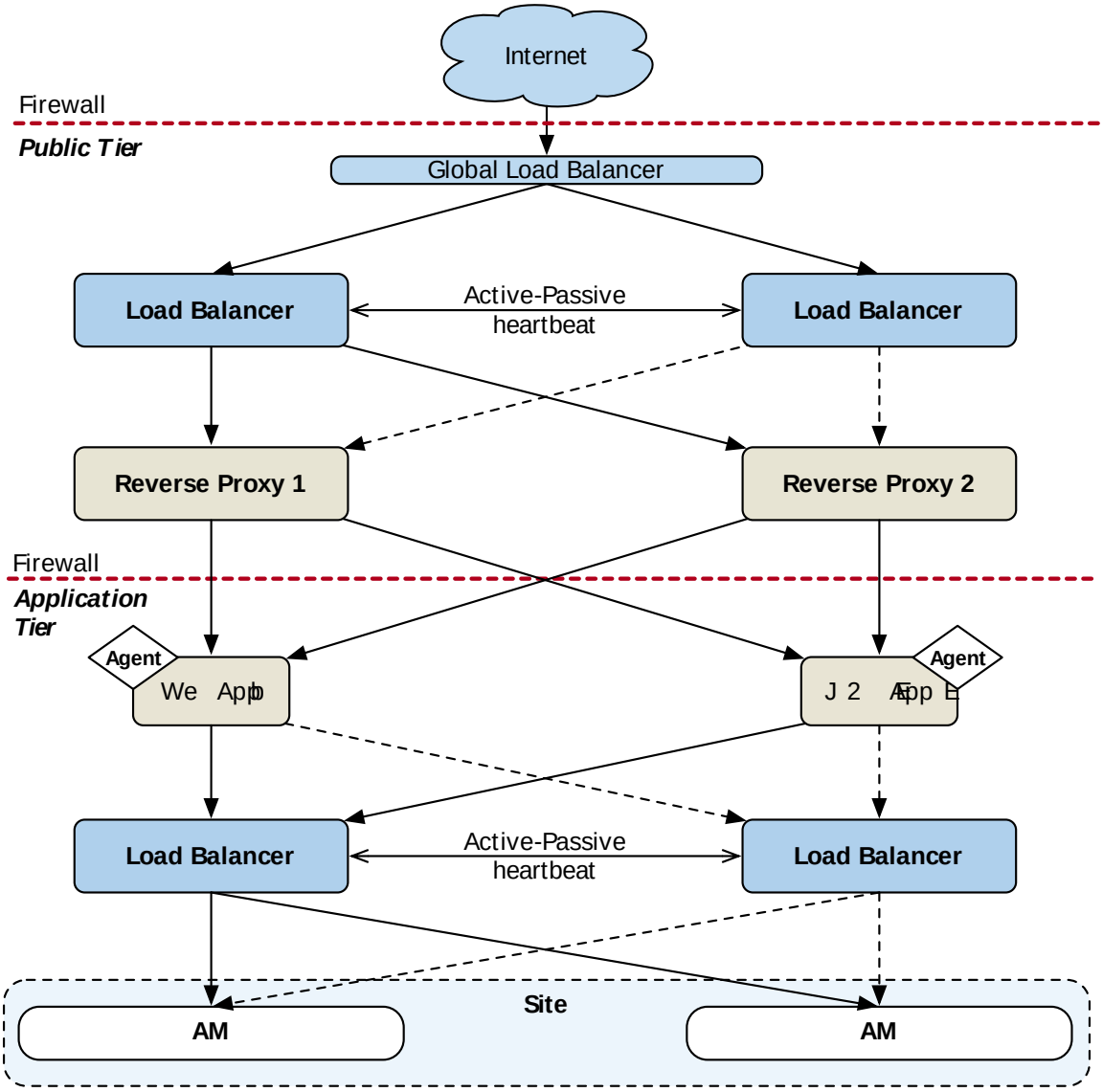


The load balancer is a single point of failure. If the load balancer goes down, then the system becomes inoperable.

Multiple Load Balancer Deployment

To make the deployment highly available, you add more than one load balancer to the set of AM servers in an active/passive configuration that provides high availability should one load balancer go down for an outage.

Site Deployment With Multiple Load Balancers



Back End Directory Servers

Each AM server requires at least an external DS instance to store policies, configuration data, and CTS tokens. You can have as many stores as your environment requires, depending on your performance needs. For large environments supporting many users, you should configure an external CTS store at the least.

Note

AM includes an embedded DS instance that you can deploy at installation time for test and demo purposes only. When AM is configured to use the embedded DS, it cannot be part of a site.

+ *About Identity Stores*

For identity repositories, AM provides built-in support for LDAP repositories. You can implement a number of different directory server vendors for storing your identity data, allowing you to configure your directory servers in a number of deployment typologies. For a list of supported data stores, refer to Directory servers.

When configuring external LDAP identity stores, you must manually carry out additional installation tasks that could require a bit more time for the configuration process. For example, you must manually add schema definitions, access control instructions (ACIs), privileges for reading and updating the schema, and resetting user passwords. For more information, see "Preparing Identity Repositories" in the *Installation Guide*.

If AM does not support your particular identity store type, you can develop your own customized plugin to allow AM to run method calls to fetch, read, create, delete, edit, or authenticate to your identity store data. For more information, see "Customizing Identity Stores" in the *Setup Guide*.

You can configure the Data Store authentication module to require the user to authenticate against a particular identity store for a specific realm. AM associates a realm with at least one identity repository and authentication process. When you initially configure AM, you define the identity repository for authenticating at the top level realm (/), which is used to administer AM. From there, you can define additional realms with different authentication and authorization services as well as different identity repositories if you have enough identity data. For more information, see "*Realms*" in the *Setup Guide*.

+ *About Configuration Data Stores*

Configuration data includes authentication information that defines how users and groups authenticate, identity store information, service information, policy information for evaluation, and partner server information that can send trusted SAML assertions. For a list of supported data stores, see Directory servers.

A combined configuration, policy, and application store may be sufficient for your environment, but you may want to deploy external policy and/or application stores if required for large-scale systems with many policies, realms, or applications (OAuth 2.0 clients, SAML entities, etc).

For more information about external stores and the type of data they contain, see "*Preparing External Stores*" in the *Installation Guide*.

+ *About CTS Data Stores*

The CTS provides persistent and highly available token storage for AM CTS-based sessions and authentication sessions, OAuth 2.0, UMA, client-based session blacklist (if enabled), client-based authentication session whitelist (if enabled), SAML v2.0 tokens for Security Token Service token validation and cancellation (if enabled), push notification for authentication, and cluster-wide notification.

CTS traffic is volatile compared to configuration data, so deploying CTS as a dedicated external data store is advantageous for systems with many users and many sessions. For more information, see Core Token Service Guide (CTS).

For high availability, configure AM to use multiple external directory servers. When you configure AM to use multiple external directory servers for a data store, AM employs internal mechanisms using DS's SDK for load balancing. Let AM perform load balancing between AM and directory servers in one of the following ways:

- AM uses failover (active/passive) load balancing for connections to configuration data stores. When AM uses multiple configuration data stores, AM connects to the primary if it is available. AM only fails over to non-primary servers when the primary is not available.
- AM uses *affinity* load balancing for its pools of connections to CTS token stores and identity repositories. Affinity load balancing routes LDAP requests with the same target DN to the same directory server. AM only fails over to another directory server when that directory server becomes unavailable. Affinity load balancing is advantageous because it:
  - Allows AM to use all available directory servers in read-write mode.
  - Combats replication delay between directory servers.
  - Removes the need for end-to-end stickiness to AM for session activities.

Note

The connection strings to the data or identity stores are static and not hot-swappable. This means that, if you expand or contract your DS affinity deployment, AM will not detect the change.

To work around this, either:

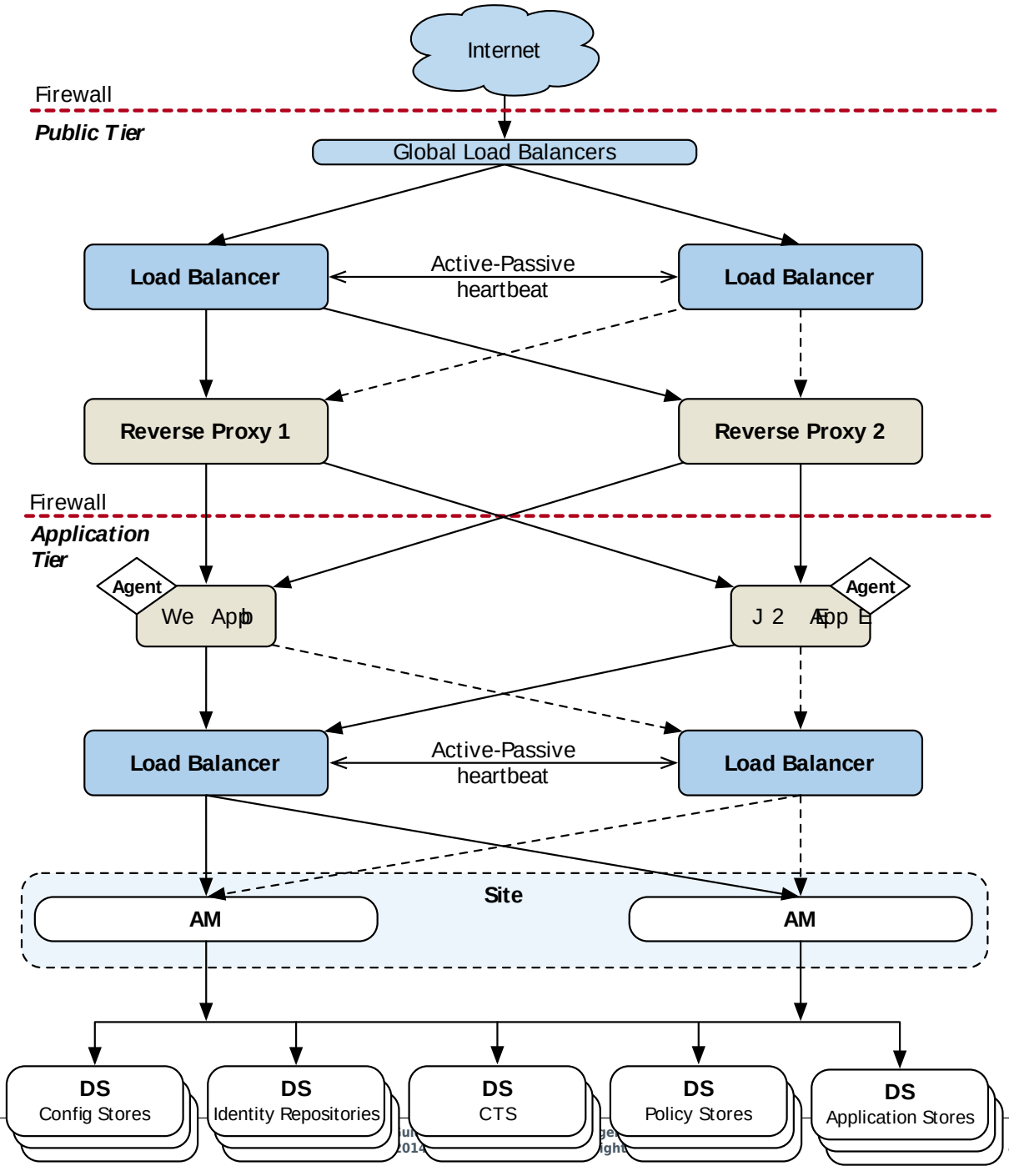
- Manually add or remove the instances from the connection string and restart AM or the container where it runs.

- Configure a DS proxy in front of the DS instances to distribute data across multiple DS *shards*, and configure the proxy's URL in the connection string.

For details regarding load balancing and directory services, see *On Load Balancers* in the *Directory Services 7.1 Configuration Guide*.

"Site Deployment With External Datastores" shows a back end deployment with replicated external DS instances for configuration, identities, CTS, policy, and application data. Although not shown in the diagram, you can also set up a directory tier, separating the application tier from the repositories with another firewall. This tier provides added security for your identity repository or policy data.

Site Deployment With External Datastores



Realms

The previous sections in this chapter present the logical and physical topologies of an example highly available AM deployment, including the clustering of servers using *sites*. One important configuration feature of AM is its ability to run multiple client entities to secure and manage applications through a single AM instance.

AM supports its multiple clients through its use of *realms*. You configure realms within AM to handle different sets of users to whom you can set up different configuration options, storage requirements, delegated administrators, and customization options per realm.

Typically, you can configure realms for customers, partners, or employees within your AM instance, for different departments, or for subsidiaries. In such cases, you create a global administrator who can delegate privileges to realm administrators, each specifically responsible for managing their respective realms.

Chapter 5

Sizing Hardware and Services For Deployment

Determine the correct size of your deployment by understanding what is that you need out of it. Availability over anything? Fast response times? Can you find a compromise?

Sizing for Availability

Any part of a system that can fail eventually will fail. Keeping your service available means tolerating failure in any part of the system without interrupting the service. You make AM services highly available through good maintenance practices and by removing single points of failure from your architectures.

Removing single points of failure involves replicating each system component, so that when one component fails, another can take its place. Replicating components comes with costs not only for the deployment and maintenance of more individual components, but also for the synchronization of anything those components share. Due to necessary synchronization between components, what you spend on availability is never fully recovered as gains in capacity. (Two servers cannot do quite twice the work of one server.) Instead you must determine the right trade offs for the deployment.

To start thinking about the trade offs, answer the following questions.

+ *What is the impact of the AM service becoming unavailable?*

  In an online system this could be a severe problem, interrupting all access to protected resources. Most deployments fall into this category.

  In an embedded system protecting local resources, it might be acceptable to restart the service.

  Deployments that require always-on service availability require some sort of load balancing solution at minimum between AM and AM client applications. The load balancer itself must be redundant, too, so that it does not become a single point of failure. Illustrations in "*Example Deployment Topology*", show multiple levels of load balancing for availability and firewalls for security.

+ *Is the service critical enough to warrant deployment across multiple sites?*

  AM allows you to deploy replicated configurations in different physical locations, so that if the service experiences complete failure at one site, you can redirect client traffic to another

site and continue operation. The question is whether the benefit in reducing the likelihood of failure outweighs the costs of maintaining multiple sites.

When you need failover across sites, one of the costs is (redundant) WAN links scaled for inter-site traffic. AM synchronizes configuration and policy data across sites, and by default also synchronizes session data as well. AM also expects profiles in identity stores to remain in sync. As shown in "*Example Deployment Topology*", the deployment involves directory service replication between sites.

+ *What is the impact of losing individual session information?*

AM offers different ways of storing session information to suit your environment needs. If your environment requires session high availability, AM can store sessions in the CTS token store. Even if a server fails or the client browser crashes, the session remains available as long as the CTS is available.

In deployments where an interruption in access to a protected resource could cause users to lose valuable information, configuring the CTS correctly can prevent the loss. The underlying directory service should replicate sessions stored in CTS. Because session information can be quite volatile—more volatile than configuration and policy data—replication of the CTS across sites can therefore call for more WAN bandwidth, as more information is shared across sites.

Once you have the answers to these questions for the deployment, you can draw a diagram of the deployment, checking for single points of failure to avoid. In the end, you should have a count of the number of load balancers, network links, and servers that you need, with the types of clients and an estimated numbers of clients that access the AM service.

Also, you must consider the requirements for non-functional testing, covered in "Planning Tests". While you might be able to perform functional testing by using a single AM server, other tests require a more complete environment with multiple servers, secure connections, and so forth. Performance testing should reveal any scalability issues. Performance testing should also run through scenarios where components fail, and check that critical functionality remains available and continues to provide acceptable levels of service.

Sizing For Service Levels

Beyond service availability, your aim is to provide some level of service. You can express service levels in terms of throughput and response times. For example, the service level goal could be to handle an average of 10,000 authentications per hour with peaks of 25,000 authentications per hour, and no more than 1 second wait for 95% of users authenticating, with an average of 100,000 live SSO sessions at any given time. Another service level goal could be to handle an average of 500 policy requests per minute per web or Java agent with an average response time of 0.5 seconds.

When you have established your service level goals, you can create load tests for each key service as described in Planning Service Performance Testing. Use the load tests to check your sizing assumptions.

To estimate sizing based on service levels, take some initial measurements and extrapolate from those measurements.

+ *For a service that handles policy decision (authorization) requests, what is the average policy size? What is the total size of all expected policies?*

  To answer these questions, you can measure the current disk space and memory occupied by the configuration directory data. Next, create a representative sample of the policies that you expect to see in the deployment, and measure the difference. Then, derive the average policy size, and use it to estimate total size.

  To measure rates of policy evaluations, you can monitor policy evaluation counts over SNMP. For details, see "SNMP Monitoring for Policy Evaluation" in the *Maintenance Guide*.

+ *How Big is CTS Data?*

  The average total size depends on the number of live CTS entries, which in turn depends on session and token lifetimes. The lifetimes are configurable and depend also on user actions like logout, that are specific to the deployment.

  For example, suppose that the deployment only handles CTS-based SSO sessions, session entries occupy on average 20 KB of memory, and that you anticipate on average 100,000 active sessions. In that case, you would estimate the need for 2 GB (100,000 x 20,000) RAM on average if you wanted to cache all the session data in memory. If you expect that sometimes the number of active sessions could rise to 200,000, then you would plan for 4 GB RAM for the session cache. Keep in mind that this is extra memory needed in addition to memory needed for everything else that the system does including running AM server.

  Session data is relatively volatile, as the CTS creates session entries when sessions are created. CTS deletes session entries when sessions are destroyed due to logout or timeout. Sessions are also modified regularly to update the idle timeout. Suppose the rate of session creation is about 5 per second, and the rate of session destruction is also about 5 per second. Then you know that the underlying directory service must handle on average 5 adds and 5 deletes per second. The added entries generate on the order of 100 KB replication traffic per second (5/s x 20 KB plus some overhead). The deleted entries generate less replication traffic, as the directory service only needs to know the distinguished name (DN) of the entry to delete, not its content.

  You can also gather statistics about CTS operations over using AM monitoring services. For more information, see "*Monitoring Instances*" in the *Maintenance Guide*.

+ *What level of network traffic do you expect for session change notifications?*

  When sizing the network, you must account for change notifications from the Core Token Server token store to the AM server.

In AM deployments, there is no direct network traffic between AM servers.

+ *What increase in user and group profile size should you expect?*

AM stores data in user profile attributes. AM can use or provision many profile attributes, as described in "To Configure an Identity Store" in the *Setup Guide*.

When you know which attributes are used, you can estimate the average increase in size by measuring the identity store as you did for configuration and CTS-related data. If you do not manage the identity store as part of the deployment, you can communicate this information with the maintainers. For a large deployment, the increase in profile size can affect sizing for the underlying directory service.

+ *How does the number of realms affect the configuration data size?*

In a centrally managed deployment with only a few realms, the size of realm configuration data might not be consequential. Also, you might have already estimated the size of policy data. For example, each new realm might add about 1 MB of configuration data to the configuration directory, not counting the policies added to the realm.

In a multi-tenant deployment or any deployment where you expect to set up many new realms, the realm configuration data and the additional policies for the realm can add significantly to the size of the configuration data overall. You can measure the configuration directory data as you did previously, but specifically for realm creation and policy configuration, so that you can estimate an average for a new realm with policies and the overall size of realm configuration data for the deployment.

See "Sizing Systems" for about CPU, memory, network, and disk-specific sizing information.

Sizing Systems

Given availability requirements and estimates on sizing for services, estimate the required capacity for individual systems, networks, and storage. This section considers the AM server systems, not the load balancers, firewalls, independent directory services, and client applications.

Although you can start with a rule of thumb, you see from the previous sections that the memory and storage footprints for the deployment depend in large part on the services you plan to provide. With that in mind, to performance test a basic deployment providing SSO, you can start with AM systems having at least 4 GB free RAM, 4 CPU cores (not throughput computing cores, but normal modern cores), plenty of local storage for configuration, policy, and CTS data, and LAN connections to other AM servers. This rule of thumb assumes the identity stores are sized separately, and that the service is housed on a single local site. Notice that this rule of thumb does not take into account anything particular to the service levels you expect to provide. Consider it a starting point when you lack more specific information.

Sizing System CPU and Memory

AM services use CPU resources to process requests and responses, and essentially to make policy decisions. Encryption, decryption, signing, and checking signatures can absorb CPU resources when processing requests and responses. Policy decision evaluation depends both on the number of policies configured and on their complexity.

Memory depends on space for AM code, on the number of live connections AM maintains, on caching of configuration data, user profile data, and CTS-based session data. The AM code in memory probably never changes while the server is running, as JSPs deployed are unlikely ever to change in production.

The number of connections and data caching depending on server tuning, as described in "*Tuning Instances*" in the *Maintenance Guide*.

If AM uses the embedded DS server, then the memory needed depends on what you store in the embedded directory and what you calculated as described in Sizing For Service Levels. The embedded DS server shares memory with the AM server process. By default, the directory server takes half of the available heap as database cache for directory data. That setting is configurable as described in the DS server documentation.

Sizing Network Connections

When sizing network connections, you must account for the requests and notifications from other servers and clients, as well as the responses. This depends on the service levels that the deployment provides, as described in Sizing For Service Levels. Responses for browser-based authentication can be quite large if each time a new user visits the authentication UI pages, AM must respond with the UI page, plus all images and JavaScript logic and libraries included to complete the authentication process. Inter-server synchronization and replication can also require significant bandwidth.

For deployments with sites in multiple locations, be sure to account for configuration, CTS, and identity directory data over WAN links, as this is much more likely to be an issue than replication traffic over LAN links.

Make sure to size enough bandwidth for peak throughput, and do not forget redundancy for availability.

Sizing Disk I/O and Storage

As described in "*Deployment Requirements*", the largest disk I/O loads for AM servers arise from logging and from the embedded DS server writing to disk. You can estimate your storage requirements as described in that section.

I/O rates depend on the service levels that the deployment provides, as described in Sizing For Service Levels. When you size disk I/O and disk space, you must account for peak rates and leave a safety margin when you must briefly enable debug logging to troubleshoot any issues that arise.

Also, keep in mind the possible sudden I/O increases that can arise in a highly available service when one server fails and other servers must take over for the failed server temporarily.

Another option is to consider placing log, configuration, and database files on a different file system to maximize throughput and minimize service disruption due to a file system full or failure scenarios.

Chapter 6

Deployment Requirements

This page lists requirements for deploying AM servers:

Server Disk Storage Requirements

Disk storage requirements for AM servers depend partly on AM itself and partly on your deployment. Disk storage requirements also depend on the space needed for binaries and configuration data, space for log files and rate of writes for logs, space for directory data and file system requirements when using an embedded DS server.

For initial installation, a few hundred MB is sufficient, not including the downloaded files.

- The AM `.war` file size varies from release to release, but if your container holds one `.war` file and one directory with the contents of the `.war` file, the disk space required is on the order of 300 MB.

  This space requirement remains stable as you use AM.

- The installation of AM with an embedded DS server requires free disk space equal to or greater than 5 GB, plus 5% of the total size of the filesystem on AM's configuration directory.

- This space requirement grows as you use AM.

By default, AM servers write audit logs to flat files under *`config-dir`*`/openam/logs/`. Alternatively, AM servers can write audit logs to `syslog`, or to a relational database.

When using flat-file audit logging, AM lets you configure rotation and purging for logs under`openam/logs/`, so you can effectively cap the maximum disk space used for logs. Make sure, however, that you

retain the information you need before logs are purged. Also make sure that your disk can keep pace with the volume of logging, which can be significant in high volume deployments, as AM logs not only errors, but also access messages.

For details about audit logging configuration, see "*Setting Up Audit Logging*" in the *Security Guide*.

By default, AM servers write debug logs to flat files under `config-dir/openam/debug/`. AM lets you configure rotation for debug logs. As you can change debug log levels at runtime when investigating issues, debug log volume is not as predictable as for regular logs. Leave a margin in production environments, so that you can turn up debug log levels to diagnose problems.

For details about debug logging configuration, see "*Debug Logging*" in the *Maintenance Guide*.

When using the embedded DS server, take the following into account:

- DS is designed to work with local storage for the database, but not for network file system (NFS) nor network-attached storage (NAS) due to some file system locking functions that DS needs. High performance storage, like solid state drives (SSD), is essential if you need to handle high write throughput.

  By default, AM's configuration directory resides under the `$HOME` directory of the user running the container. `$HOME` directories can be mounted over the network.

  This is not an issue if you are using DS mainly for configuration data. It can however be a serious problem when you use DS to back the CTS in a high-volume deployment.

- Embedded DS server log files are stored under `/path/to/openam/opends/logs/`. As for AM, you can configure DS server log rotation and purging. The default cap for access logs is 2 GB.

- AM stores policy information in the configuration directory. The space this takes up depends on the policies you have.

- By default, AM stores CTS information in the configuration directory. The space this takes up depends on the volume of traffic to the server and whether AM is configured for client-based sessions.

- With the default database implementation, DS database files handling sustained writes can grow to about double their initial size on disk.

- For DS on Linux systems, enable file system write barriers and ensure the file system journaling mode is ordered to avoid directory database file corruption after crashes or power failures. For details on enabling write barriers and setting the journaling mode for data, see the options for your file system in the **mount** command manual page.

- DS uses file descriptors when handling connections.

  Defaults can be limited to 1024 file descriptors per user on Linux systems. Consider increasing this limit to at least 64K. For details, see "Setting Maximum File Descriptors and Processes Per User" in the *Installation Guide*.

Web and Java Agents Disk Storage Requirements

Web and Java agent binaries do not require more than a few MB of disk space, although they may require additional free space to store configuration files, POST data cache files, and others. Refer to the installation requirements of your web or Java agent for more information.

You should also consider the web or Java agent logging when provisioning disk storage:

- Web and Java agents can log audit messages locally to the agent installation or can send them to the AM instances. Refer to the configuration reference of your agent for more information.
- Debug messages are logged to files local to the agent installation, and their volume depends on the debug log level. In production environments, provision additional storage to ensure you can enable higher debug log levels for diagnostic purposes.

Identity Gateway Disk Storage Requirements

The IG Web application can vary in size from release to release. On disk, the `.war` file is under 50 MB. For containers that keep both the `.war` file and an unpacked version, the total size is under 100 MB.

By default, IG configuration resides under the `$HOME` directory of the user who runs the container.

If you use the default log sink, messages are sent to the container logs. Manage those as you would any container logs.

Both normal log messages and debug messages go to the log sink. As for other components, debug logging volume depends on log level. Leave a margin in production environments so that you can turn up debug log levels to diagnose problems.

IG does not run rotation or purging of the following logs, which you must manually manage:

- Logs generated using a `CaptureFilter`
- Log messages created by scriptable filters and handlers

Disk Storage Recommendations

The following are based on the preceding information in this section. When deciding on disk storage, keep the following recommendations in mind:

- Plan enough space and enough disk I/O to comfortably absorb the load for logs.

  Check your assumptions in testing. For example, make sure that logs are cleaned up so that they do not exceed your space threshold even in long-duration testing.
- When using local web or Java agent logs, make sure you have a mechanism in place to clean them up.

- For IG, make sure you turn off `CaptureFilter` logging, scriptable filter, and handler debug logging before moving to production.

RAM Requirements

AM core services require a minimum JVM heap size of 1 GB. If you are deploying with the embedded DS server, AM requires at least a 2 GB heap, as 50% of that space is allocated to DS.

Note

Ensure that the `Xms` and `Xmx` JVM parameters are set to the same value to prevent a large garbage collection as the memory profile increases from the default up to the `Xms` value. Also, setting `Xms` and `Xmx` to the same value ensures that small controlled garbage collection events minimize application unresponsiveness.

Software Requirements

For up-to-date information about the software requirements for this version, refer to Requirements.

Chapter 7

Getting Started for Architects and Deployers

- **Learn about AM**. You can access online information, meet with your ForgeRock Sales representative, go to a seminar, or call ForgeRock about AM's capabilities.

  The following are some general questions that you may want to have answered:

Initial Questions

Initial Tasks	Done ?	
Understand the access management problems that AM helps to solve	Y	N
Learn how to protect a Web site with AM	Y	N
Get to know the AM software deliverables	Y	N
Get to know the tools for administering AM	Y	N
Get to know the APIs for AM client applications	Y	N
Find out how to get help and support from ForgeRock and partners	Y	N
Find out how to get training from ForgeRock and partners	Y	N
Find out how to keep up to date on new development and new releases	Y	N
Find out how to report problems	Y	N

- **Set up a Demo or Pilot**. View an AM demo or set up a pilot to determine how you want to use AM to protect your site(s). ForgeRock Sales representatives can assist you with a demo or pilot.
- **Attend a Training Class**. ForgeRock presents effective training classes to deploy AM in your environment. See ForgeRock University for more information.
- **Complete the Accreditation Program**. Complete the product-specific ForgeRock Accreditation Program to gain in-depth design and deployment expertise or seek partners who are ForgeRock Accredited Partners.
- **Determine Your Service Level Agreements**. ForgeRock provides a set of standard service level agreements that you can sign up for. ForgeRock also provides custom service level agreements if the standard set does not meet your needs.

Standard SLAs

Priority	Gold	Silver	Bronze
Urgent (P1)	2 Hour	4 Hour	Next Business Day
High (P2)	4 Hour	8 Hour	2 Business Days
Normal (P3)	6 Hour	Next Business Day	3 Business Days
Low (P4)	Next Business Day	2 Business Days	4 Business Days

- **Determine Your Services**. ForgeRock provides a full, proven-production Identity Management stack to meet your requirements.

Services

Services Task	Done ?	
Understand the services AM software provides	Y	N
Determine which services to deploy	Y	N
Determine which services the deployment consumes (load balancing, application container, authentication services, configuration storage, profile storage, token/session storage, policy storage, log storage)	Y	N
Determine which services the deployment provides (SSO, CDSSO, SAML Federation IDP/SP, XACML PDP, REST STS, OAuth 2.0/OpenID Connect 1.0, and so forth)	Y	N
Determine which resources AM protects (who consumes AM services)	Y	N

- **Determine Your Deployment Objectives**. AM provides proven performance and security in many production deployments. You should determine your overall deployment objectives.

Deployment Objectives

Deployment Objectives	Done ?	
Define deployment objectives in terms of service levels (expectations for authentication rates, active sessions maintained, session life cycles, policies managed, authorization decision rates, response times, throughput, and so forth)	Y	N
Define deployment objectives in terms of service availability (AM service availability, authentication availability, authorization decision availability, session availability, elasticity)	Y	N
Understand how AM services scale for high availability	Y	N
Understand the restrictions in an AM deployment that uses client-based sessions	Y	N
Plan for availability (number of sites and servers, load balancing and AM software configuration)	Y	N
Define the domains managed and domains involved in the deployment	Y	N
Define deployment objectives for delegated administration	Y	N

Deployment Objectives	Done ?	
Agree with partners for federated deployments on circles of trust and terms	Y	N

- **Plan Sizing**. At this stage, you should determine the sizing estimates for your deployment. ForgeRock Sales Engineers can assist you in this task.

Sizing

Sizing	Done ?	
Derive sizing estimates from service levels and availability	Y	N
Understand how to test sizing estimates (load generation tools?)	Y	N
Size servers for AM deployment: CPU	Y	N
Size servers for AM deployment: Memory	Y	N
Size servers for AM deployment: Network	Y	N
Size servers for AM deployment: I/O	Y	N
Size servers for AM deployment: Storage	Y	N
Quantify impact on external services consumed (LDAP, other auth services, load balancing, and so forth)	Y	N
Plan testing and acceptance criteria for sizing	Y	N

- **Plan the Topology**. Plan your logical and physical deployment.

Topology Planning

Topology	Done ?	
Specify the logical and physical deployment topology (show examples of each)	Y	N
Determine how many external stores you need (configuration, CTS, application, policy, UMA...)	Y	N
Plan installation of AM services (including external dependencies)	Y	N
Plan installation of AM web and Java agents, Fedlets, and IG (might be done by partner service providers)	Y	N
Plan integration with client applications	Y	N
Plan customization of AM (UI, user profile attributes, authentication modules, identity repositories, OAuth 2.0 scope handling, OAuth 2.0 response types, post-authentication actions, policy evaluation, session quota exhaustion actions, policy evaluation, identity data storage, AM service, custom logger, custom policy enforcement points or agents).	Y	N

- **Plan Security**. At this stage, you must plan how to secure your deployment.

Security

Security	Done ?	
Understand security guidelines, including legal requirements	Y	N
Change default settings and administrative user credentials	Y	N
Protect service ports (Firewall, Dist Auth UI, reverse proxy)	Y	N
Turn off unused service endpoints	Y	N
Separate administrative access from client access	Y	N
Secure communications (HTTPS, LDAPS, secure cookies, cookie hijacking protection, key management for signing and encryption)	Y	N
Determine if components handle SSL acceleration or termination	Y	N
Securing processes and files (e.g. with SELinux, dedicated non-privileged user and port forwarding, and so forth)	Y	N

- **Post-Deployment Tasks**. At this stage, you should plan your post-deployment tasks to sustain and monitor your system.

Post-Deployment Tasks

Post Deployment Tasks	Done ?	
Plan administration following AM deployment (services, agents/IG, delegated administration)	Y	N
Plan monitoring following deployment	Y	N
Plan how to expand the deployment	Y	N
Plan how to upgrade the deployment	Y	N

Glossary

Access control	Control to grant or to deny access to a resource.
Account lockout	The act of making an account temporarily or permanently inactive after successive authentication failures.
Actions	Defined as part of policies, these verbs indicate what authorized identities can do to resources.
Advice	In the context of a policy decision denying access, a hint to the policy enforcement point about remedial action to take that could result in a decision allowing access.
Agent administrator	User having privileges only to read and write agent profile configuration information, typically created to delegate agent profile creation to the user installing a web or Java agent.
Agent authenticator	Entity with read-only access to multiple agent profiles defined in the same realm; allows an agent to read web service profiles.
Application	In general terms, a service exposing protected resources. In the context of AM policies, the application is a template that constrains the policies that govern access to protected resources. An application can have zero or more policies.
Application type	Application types act as templates for creating policy applications. Application types define a preset list of actions and functional logic, such as policy lookup and resource comparator logic.

	Application types also define the internal normalization, indexing logic, and comparator logic for applications.
Attribute-based access control (ABAC)	Access control that is based on attributes of a user, such as how old a user is or whether the user is a paying customer.
Authentication	The act of confirming the identity of a principal.
Authentication chaining	A series of authentication modules configured together which a principal must negotiate as configured in order to authenticate successfully.
Authentication level	Positive integer associated with an authentication module, usually used to require success with more stringent authentication measures when requesting resources requiring special protection.
Authentication module	AM authentication unit that handles one way of obtaining and verifying credentials.
Authorization	The act of determining whether to grant or to deny a principal access to a resource.
Authorization Server	In OAuth 2.0, issues access tokens to the client after authenticating a resource owner and confirming that the owner authorizes the client to access the protected resource. AM can play this role in the OAuth 2.0 authorization framework.
Auto-federation	Arrangement to federate a principal's identity automatically based on a common attribute value shared across the principal's profiles at different providers.
Bulk federation	Batch job permanently federating user profiles between a service provider and an identity provider based on a list of matched user identifiers that exist on both providers.
Circle of trust	Group of providers, including at least one identity provider, who have agreed to trust each other to participate in a SAML v2.0 provider federation.
Client	In OAuth 2.0, requests protected web resources on behalf of the resource owner given the owner's authorization. AM can play this role in the OAuth 2.0 authorization framework.
Client-based OAuth 2.0 tokens	After a successful OAuth 2.0 grant flow, AM returns a token to the client. This differs from CTS-based OAuth 2.0 tokens, where AM returns a *reference* to token to the client.
Client-based sessions	AM sessions for which AM returns session state to the client after each request, and require it to be passed in with the subsequent

	request. For browser-based clients, AM sets a cookie in the browser that contains the session information. For browser-based clients, AM sets a cookie in the browser that contains the session state. When the browser transmits the cookie back to AM, AM decodes the session state from the cookie.
Conditions	Defined as part of policies, these determine the circumstances under which which a policy applies. Environmental conditions reflect circumstances like the client IP address, time of day, how the subject authenticated, or the authentication level achieved. Subject conditions reflect characteristics of the subject like whether the subject authenticated, the identity of the subject, or claims in the subject's JWT.
Configuration datastore	LDAP directory service holding AM configuration data.
Cross-domain single sign-on (CDSSO)	AM capability allowing single sign-on across different DNS domains.
CTS-based OAuth 2.0 tokens	After a successful OAuth 2.0 grant flow, AM returns a *reference* to the token to the client, rather than the token itself. This differs from client-based OAuth 2.0 tokens, where AM returns the entire token to the client.
CTS-based sessions	AM sessions that reside in the Core Token Service's token store. CTS-based sessions might also be cached in memory on one or more AM servers. AM tracks these sessions in order to handle events like logout and timeout, to permit session constraints, and to notify applications involved in SSO when a session ends.
Delegation	Granting users administrative privileges with AM.
Entitlement	Decision that defines which resource names can and cannot be accessed for a given identity in the context of a particular application, which actions are allowed and which are denied, and any related advice and attributes.
Extended metadata	Federation configuration information specific to AM.
Extensible Access Control Markup Language (XACML)	Standard, XML-based access control policy language, including a processing model for making authorization decisions based on policies.
Federation	Standardized means for aggregating identities, sharing authentication and authorization data information between trusted providers, and

	allowing principals to access services across different providers without authenticating repeatedly.
Fedlet	Service provider application capable of participating in a circle of trust and allowing federation without installing all of AM on the service provider side; AM lets you create Java Fedlets.
Hot swappable	Refers to configuration properties for which changes can take effect without restarting the container where AM runs.
Identity	Set of data that uniquely describes a person or a thing such as a device or an application.
Identity federation	Linking of a principal's identity across multiple providers.
Identity provider (IDP)	Entity that produces assertions about a principal (such as how and when a principal authenticated, or that the principal's profile has a specified attribute value).
Identity repository	Data store holding user profiles and group information; different identity repositories can be defined for different realms.
Java agent	Java web application installed in a web container that acts as a policy enforcement point, filtering requests to other applications in the container with policies based on application resource URLs.
Metadata	Federation configuration information for a provider.
Policy	Set of rules that define who is granted access to a protected resource when, how, and under what conditions.
Policy agent	Java, web, or custom agent that intercepts requests for resources, directs principals to AM for authentication, and enforces policy decisions from AM.
Policy Administration Point (PAP)	Entity that manages and stores policy definitions.
Policy Decision Point (PDP)	Entity that evaluates access rights and then issues authorization decisions.
Policy Enforcement Point (PEP)	Entity that intercepts a request for a resource and then enforces policy decisions from a PDP.
Policy Information Point (PIP)	Entity that provides extra information, such as user profile attributes that a PDP needs in order to make a decision.
Principal	Represents an entity that has been authenticated (such as a user, a device, or an application), and thus is distinguished from other entities.

	When a Subject successfully authenticates, AM associates the Subject with the Principal.
Privilege	In the context of delegated administration, a set of administrative tasks that can be performed by specified identities in a given realm.
Provider federation	Agreement among providers to participate in a circle of trust.
Realm	AM unit for organizing configuration and identity information. Realms can be used for example when different parts of an organization have different applications and identity stores, and when different organizations use the same AM deployment. Administrators can delegate realm administration. The administrator assigns administrative privileges to users, allowing them to perform administrative tasks within the realm.
Resource	Something a user can access over the network such as a web page. Defined as part of policies, these can include wildcards in order to match multiple actual resources.
Resource owner	In OAuth 2.0, entity who can authorize access to protected web resources, such as an end user.
Resource server	In OAuth 2.0, server hosting protected web resources, capable of handling access tokens to respond to requests for such resources.
Response attributes	Defined as part of policies, these allow AM to return additional information in the form of "attributes" with the response to a policy decision.
Role based access control (RBAC)	Access control that is based on whether a user has been granted a set of permissions (a role).
Security Assertion Markup Language (SAML)	Standard, XML-based language for exchanging authentication and authorization data between identity providers and service providers.
Service provider (SP)	Entity that consumes assertions about a principal (and provides a service that the principal is trying to access).
Authentication Session	The interval while the user or entity is authenticating to AM.
Session	The interval that starts after the user has authenticated and ends when the user logs out, or when their session is terminated. For browser-based clients, AM manages user sessions across one or more applications by setting a session cookie. See also CTS-based sessions and Client-based sessions.

Session high availability	Capability that lets any AM server in a clustered deployment access shared, persistent information about users' sessions from the CTS token store. The user does not need to log in again unless the entire deployment goes down.
Session token	Unique identifier issued by AM after successful authentication. For a CTS-based sessions, the session token is used to track a principal's session.
Single log out (SLO)	Capability allowing a principal to end a session once, thereby ending her session across multiple applications.
Single sign-on (SSO)	Capability allowing a principal to authenticate once and gain access to multiple applications without authenticating again.
Site	Group of AM servers configured the same way, accessed through a load balancer layer. The load balancer handles failover to provide service-level availability. The load balancer can also be used to protect AM services.
Standard metadata	Standard federation configuration information that you can share with other access management software.
Stateless Service	Stateless services do not store any data locally to the service. When the service requires data to perform any action, it requests it from a data store. For example, a stateless authentication service stores session state for logged-in users in a database. This way, any server in the deployment can recover the session from the database and service requests for any user. All AM services are stateless unless otherwise specified. See also Client-based sessions and CTS-based sessions.
Subject	Entity that requests access to a resource When an identity successfully authenticates, AM associates the identity with the Principal that distinguishes it from other identities. An identity can be associated with multiple principals.
Identity store	Data storage service holding principals' profiles; underlying storage can be an LDAP directory service or a custom `IdRepo` implementation.
Web Agent	Native library installed in a web server that acts as a policy enforcement point with policies based on web page URLs.