

# Getting Started With OpenAM

Mark Craig, Gene Hirayama

# Table of Contents

Preface .....	2
Who Should Use this Guide .....	2
Formatting Conventions .....	2
Accessing Documentation Online .....	2
Joining the Open Identity Platform Community .....	3
Getting Support and the Contacting Open Identity Platform Community .....	3
Protecting a Web Site With OpenAM .....	4
About OpenAM .....	4
Software Requirements To Try Out OpenAM .....	4
Setting Up the Software .....	5

*Quick introduction to OpenAM for new users and readers evaluating the product. OpenAM provides open source Authentication, Authorization, Entitlement, and Federation software.*

# Preface

This guide shows you how to install and get started with OpenAM.

## Who Should Use this Guide

This guide is written for access management designers and administrators who build, deploy, and maintain OpenAM services for their organizations. This guide covers the tasks you need to quickly get OpenAM running on your system.

You do not need to be an OpenAM wizard to learn something from this guide, though a background in access management and maintaining web application software can help. You do need some background in managing services on your operating systems and in your application servers. You can nevertheless get started with this guide, and then learn more as you go along.

## Formatting Conventions

Most examples in the documentation are created in GNU/Linux or Mac OS X operating environments. If distinctions are necessary between operating environments, examples are labeled with the operating environment name in parentheses. To avoid repetition file system directory names are often given only in UNIX format as in `/path/to/server`, even if the text applies to `C:\path\to\server` as well. Absolute path names usually begin with the placeholder `/path/to/`. This path might translate to `/opt/`, `C:\Program Files\`, or somewhere else on your system. Command-line, terminal sessions are formatted as follows:

```
$ echo $JAVA_HOME
/path/to/jdk
```

Command output is sometimes formatted for narrower, more readable output even though formatting parameters are not shown in the command. Program listings are formatted as follows:

```
class Test {
    public static void main(String [] args) {
        System.out.println("This is a program listing.");
    }
}
```

## Accessing Documentation Online

Open Identity Platform Community publishes comprehensive documentation online:

- The Open Identity Platform Community [Documentation](#) offers a large and increasing number of up-to-date, practical articles that help you deploy and manage Open Identity Platform software.
- Open Identity Platform product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and

covers all product features and examples of how to use them.

## Joining the Open Identity Platform Community

Visit the [community resource center](#) where you can find information about each project, download nightly builds, browse the resource catalog, ask and answer questions on the forums, find community events near you, and of course get the source code as well.

## Getting Support and the Contacting Open Identity Platform Community

Open Identity Platform Community [Approved Vendors](#) provide support services, professional services, trainings, and partner services to assist you in setting up and maintaining your deployments.

# Protecting a Web Site With OpenAM

This guide shows you how to quickly set up OpenAM and get started with access management. In reading and following the instructions in this guide, you will learn how to protect a Web page using OpenAM and a Web policy agent.

## About OpenAM

OpenAM provides a service called *access management*, which manages access to resources, such as a web page, an application, or a web service, available over the network. Once it is set up, OpenAM provides an infrastructure for managing users, roles, and access to resources. In this chapter, you manage access to a single web page.

OpenAM centralizes access control by handling both *authentication* and *authorization*. Authentication is the process of identifying an individual, for example, by confirming a successful login. Authorization is the process of granting access to resources to authenticated individuals.

OpenAM centralizes authentication by using a variety of authentication modules that connect to identity repositories that store identities and provide authentication services. The identity repositories can be implemented as LDAP directories, relational databases, RADIUS, Windows authentication, one-time password services, and other standards-based access management systems.

OpenAM lets you chain together the authentication services used. Authentication chains let you configure stronger authentication for more sensitive resources for example. They also let you set up modules that remember a device when the user logs in successfully. Or that evaluate the risk given the login circumstances and therefore can require more credentials when a user is logging in from an unusual location. This chapter uses OpenAM's built-in identity repository and authentication modules to make it easier to get started.

OpenAM centralizes authorization by letting you use OpenAM to manage access policies separate from applications and resources. Instead of building access policy into a web application, you install a policy agent with the web application to request policy decisions from OpenAM. This way you can avoid issues that could arise when developers must embed policy decisions into their applications. With OpenAM, if policy changes or an issue is found after the application is deployed, you have only to change the policy definition in OpenAM, not deploy a new version of the application. OpenAM makes the authorization decisions, and policy agents enforce the decisions on OpenAM's behalf.

The rest of this chapter has you demonstrate OpenAM access management by installing OpenAM, creating a policy, and installing a policy agent on a web server to enforce the policy for a web page.

## Software Requirements To Try Out OpenAM

The only software you need to install is Docker. If you don't have Docker on your computer follow the instructions from the following link: <https://docs.docker.com/get-started/get-docker/#supported-platforms>.

You will learn how to run OpenAM docker container, and how to install OpenAM Apache Policy Agent for the Apache HTTP Server.

## Setting Up the Software

This section includes the following procedures that detail how to set up OpenAM to protect a web page:

- ["To Prepare Your Hosts File"](#)
- ["To Run OpenAM Docker Image"](#)
- [To Configure Cookie Domain in OpenAM](#)
- ["To Configure a Policy in OpenAM"](#)
- ["To Create a Web Policy Agent Profile"](#)
- ["To Install OpenAM Web Policy Agent"](#)

The procedures in this section are written for use on a Linux system. If you are running Microsoft Windows, adapt the examples accordingly.

### *To Prepare Your Hosts File*

OpenAM requires that you use fully qualified domain names when protecting web resources. This is because OpenAM uses [HTTP cookies](#) to keep track of sessions for single sign-on (SSO), and setting and reading cookies depends on the server name and domain.

You can get started with OpenAM without setting up separate systems for each fully qualified domain name. Give your system [openam.example.com](#) and [www.example.com](#) aliases by editing your [hosts file](#).

Alternatively, if you already have a DNS set up, you can use that instead of your hosts file.

- Add the aliases to your hosts file using your preferred text editor.

```
$ sudo vi /etc/hosts
Password:

### Edit /etc/hosts ###

$ cat /etc/hosts | grep openam
127.0.0.1    localhost openam.example.com www.example.com
```

### *To Install OpenAM*

Create a Docker network for OpenAM.

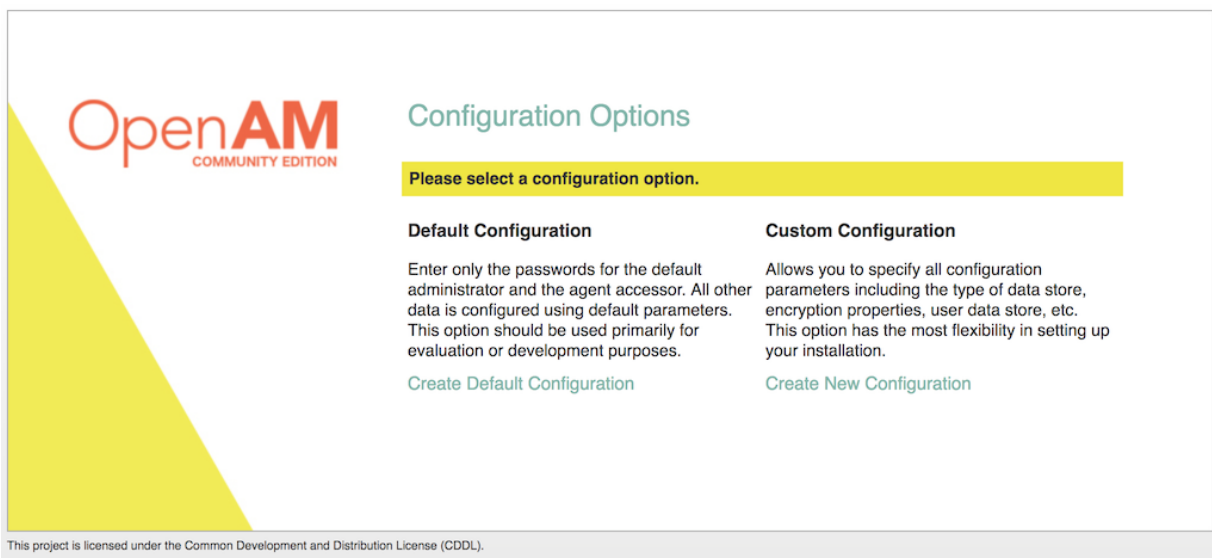
```
$ docker network create openam-quickstart
```

Run the OpenAM Docker image.

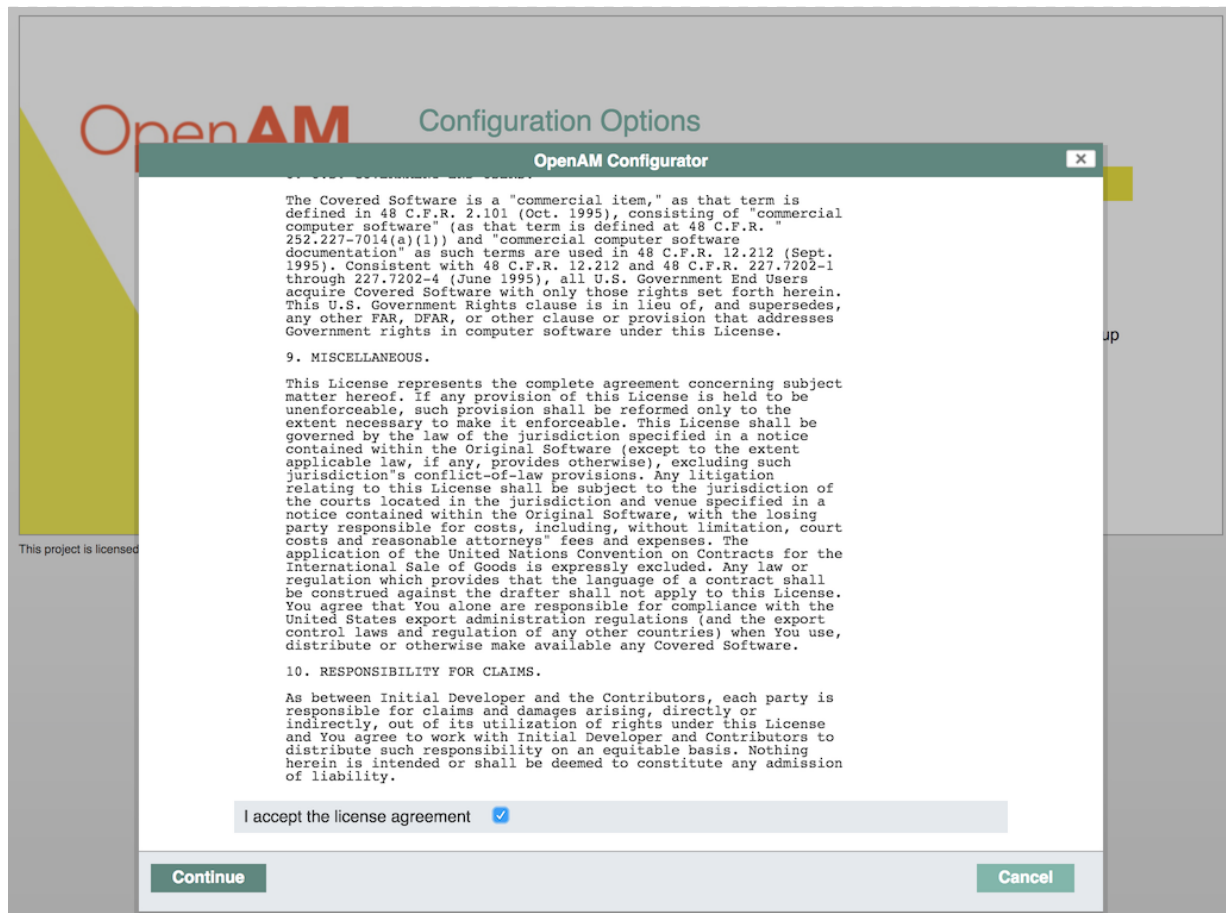
```
$ docker run -h openam.example.com -p 8080:8080 --network openam-quickstart --name openam openidentityplatform/openam
```

You can access the web application in a browser at <http://openam.example.com:8080/openam/>.

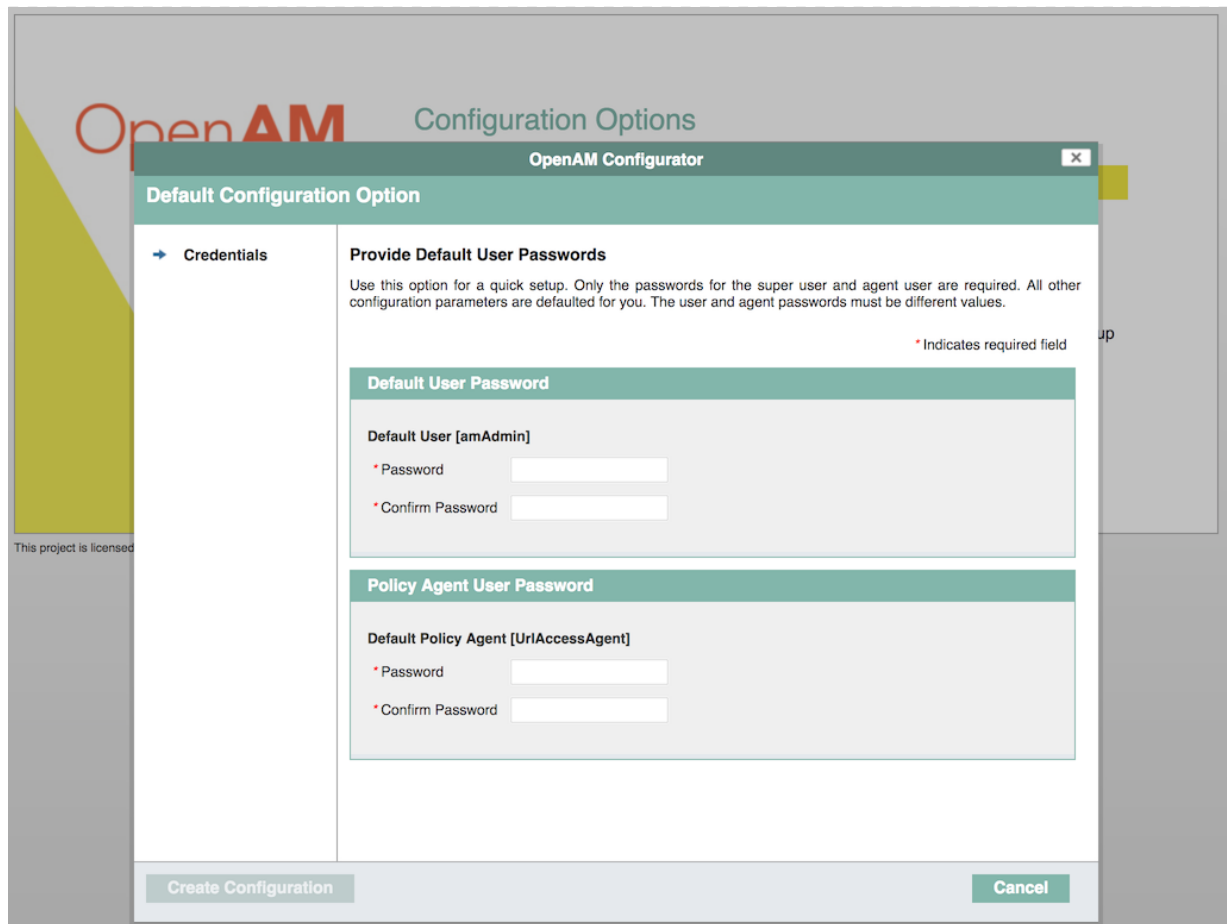
1. Browse to OpenAM in this example, <http://openam.example.com:8080/openam/>, to configure the application.
2. On the OpenAM home page, click Create Default Configuration.



3. Review the software license agreement. If you agree to the license, click "I accept the license agreement", and then click Continue.



4. Set the Default User [amAdmin] password to **changeit** and the Default Policy Agent [UrlAccessAgent] password to **secret12**, and then click Create Configuration to configure OpenAM.



**OpenAM Configuration Options**

**OpenAM Configurator**

**Default Configuration Option**

→ **Credentials**

**Provide Default User Passwords**

Use this option for a quick setup. Only the passwords for the super user and agent user are required. All other configuration parameters are defaulted for you. The user and agent passwords must be different values.

\* Indicates required field

**Default User Password**

Default User [amAdmin]

\* Password

\* Confirm Password

**Policy Agent User Password**

Default Policy Agent [UrlAccessAgent]

\* Password

\* Confirm Password

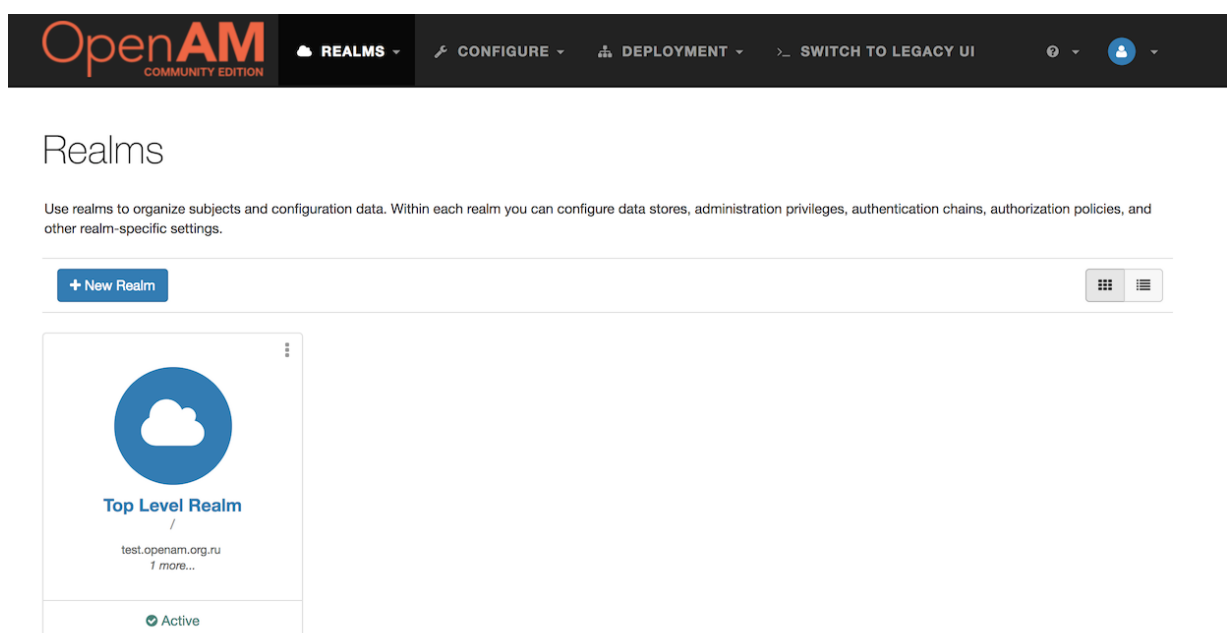
Create Configuration Cancel

## NOTE

If you were configuring OpenAM for real-world use, you would not use either of those passwords, but this is only to get started with OpenAM. The **amadmin** user is the OpenAM administrator, who is like a superuser in that **amadmin** has full control over the OpenAM configuration.

- Click the Proceed to Login link, then log in as **amadmin** with the password specified in the previous step, **changeit**.

After login, OpenAM should direct you to the Realms page.



OpenAM stores its configuration, including the embedded OpenDJ directory server in the folder named `~/openam/` in your home directory. The folder shares the same name as your server instance. It also has a hidden folder, `~/.openamcfg/`, with a file used by OpenAM when it starts up. If you ruin your configuration of OpenAM somehow, the quickest way to start over is to stop Tomcat, delete these two folders, and configure OpenAM again.

The OpenAM core server and OpenAM Console are now configured. Make sure you have successfully logged in to OpenAM Console before you proceed.

### To Configure Cookie Domain in OpenAM

Navigate to Configure → Global Services → Platform → Cookie Domain.

Set the cookie domain to `.example.com`, and save your settings.

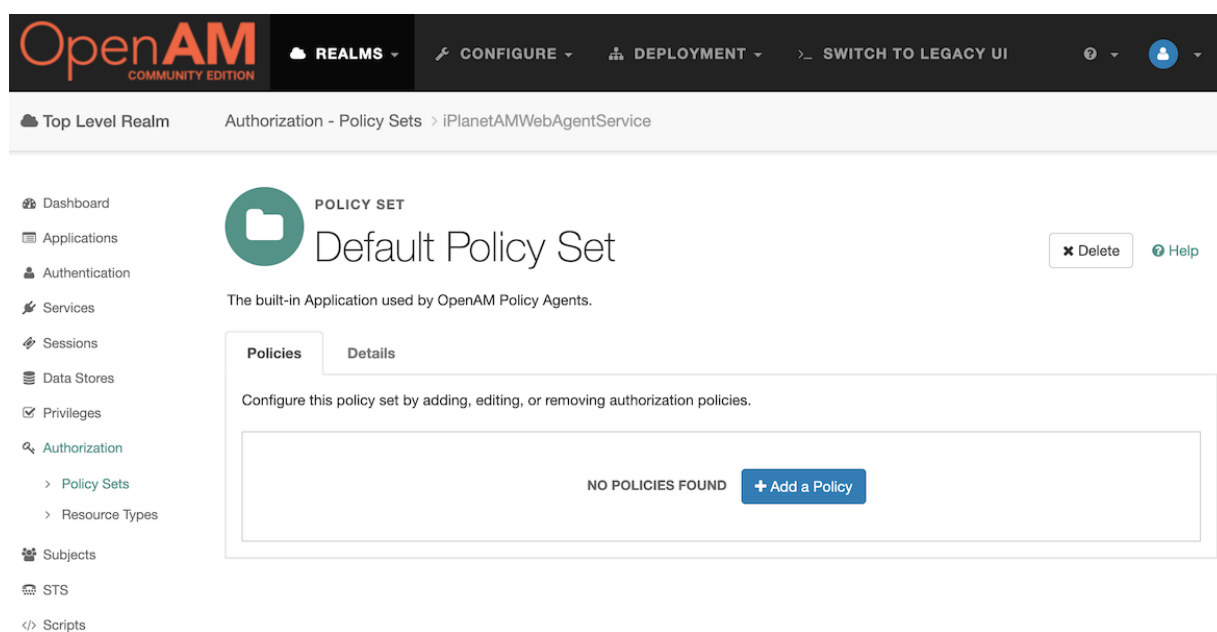
### To Configure a Policy in OpenAM

OpenAM authenticates users and then makes authorization decisions based on access policies that indicate user entitlements. Follow these steps to create a policy that allows all authenticated users to perform an HTTP GET (for example, to browse) the Apache HTTP home page that you set up earlier.

1. In the OpenAM console, select the Top Level Realm on the Realms page.

OpenAM allows you to organize identities, policies, and policy agent profiles into realms as described in "[Configuring Realms](#)" in the *Administration Guide*. For now, use the default Top Level Realm.

2. On the Realm Overview page, navigate to Authorization > Policy Sets > **Default Policy Set** > Add a Policy.



For more information on the relationship between realms, policy sets, and policies, see

"About Authorization in OpenAM" in the *Administration Guide*.

3. On the New Policy page, enter the following data:

- In the Name field, give your new policy the name **Authenticated users can get Apache HTTP home page**.
- On the Resource Type drop-down list, select **URL**.
- On the Resources drop-down list, select the URL pattern for your policy. In this example, select **::/\*/\***, then enter the resource URL: **http://www.example.com:8000/**, and then click Add.

OpenAM COMMUNITY EDITION

REALMS CONFIGURE DEPLOYMENT SWITCH TO LEGACY UI

Top Level Realm Authorization - Policy Sets iPlanetAMWebAgentService policies > new

Dashboard Applications Authentication Services Sessions Data Stores Privileges Authorization > Policy Sets > Resource Types Subjects STS Scripts

## New Policy

Help

**Name** Authenticated users can get Apache HTTP home page

**Resource Type** URL

Select the type of resource for which this policy will manage access.

**Resources** ::/\*/\*

http :// www.example.com : 8000 / Cancel Add

Cancel Create

d. Click Create to save your settings.

OpenAM COMMUNITY EDITION

REALMS CONFIGURE DEPLOYMENT SWITCH TO LEGACY UI

Top Level Realm Authorization - Policy Sets iPlanetAMWebAgentService policies > new

Dashboard Applications Authentication Services Sessions Data Stores Privileges Authorization > Policy Sets > Resource Types Subjects STS Scripts

## New Policy

Help

**Name** Authenticated users can get Apache HTTP home page

**Resource Type** URL

Select the type of resource for which this policy will manage access.

**Resources** http://www.example.com:8000/ ✕

+ Add Resource

Cancel Create

4. On your policy page, select the Actions tab, and then enter the following information:

- On the Add an action drop-down list, select **GET**.
- On the Add an action drop-down list, select **POST**.
- Save your changes.

The screenshot shows the OpenAM Community Edition interface. The top navigation bar includes 'REALMS', 'CONFIGURE', 'DEPLOYMENT', and 'SWITCH TO LEGACY UI'. The breadcrumb trail is: Top Level Realm > Authorization - Policy Sets > iPlanetAMWebAgentService > policies > Authenticated users can get Apache HTTP home page. The left sidebar lists various menu items, with 'Authorization' expanded to show 'Policy Sets' and 'Resource Types'. The main content area shows the 'Actions' tab for the policy. It includes a table with columns 'ACTION' and 'DEFAULT STATE'. The table has two rows: 'GET' and 'POST', both with 'Allow' selected under 'DEFAULT STATE'. There are 'Delete' and 'Help' buttons in the top right, and a 'Save Changes' button at the bottom right.

ACTION	DEFAULT STATE
GET	<input checked="" type="radio"/> Allow <input type="radio"/> Deny
POST	<input checked="" type="radio"/> Allow <input type="radio"/> Deny

5. On your policy page, navigate to Subjects and enter the following data:
  - a. On the All of drop-down list, review the list and select **All of...**.
  - b. On the Type section, click the Edit icon. On the Type drop-down list, select **Authenticated Users**, and then click the checkmark.
  - c. Save your changes.

The screenshot shows the OpenAM Community Edition interface with the 'Subjects' tab selected. The breadcrumb trail is the same as the previous screenshot. The main content area shows the 'Subjects' tab for the policy. It includes a section 'Specify the subject conditions to which the policy applies.' with a dropdown menu set to 'All of...'. Below this is a section 'Type' with 'Authenticated Users' selected. There are 'Add a Subject Condition' and 'Add a Logical Operator' buttons at the bottom left, and a 'Save Changes' button at the bottom right.

6. Review your configuration. To make changes to the configuration, click the relevant tab and amend the configuration.

Next, you must create a web policy agent profile before installing the agent in Apache HTTP Server

to enforce your new policy.

### To Create a Web Policy Agent Profile

OpenAM stores profile information about policy agents centrally by default. You can manage the policy agent profile through OpenAM Console. The policy agent retrieves its configuration from its OpenAM profile at installation and start up, and OpenAM notifies the policy agent of changes to its configuration. Follow these steps before installing the policy agent itself.

1. In OpenAM Console, browse to Realms > / Top Level Realm > Applications > Web Agents, and then click New in the Agents table.
2. In the page to configure your new web policy agent, set the following values.

#### Name

WebAgent

#### Password

password

#### Configuration

Keep the default, Centralized

#### Server URL

http://openam.example.com:8080/openam

#### Agent URL

http://www.example.com:8000

8000 is the port number you set previously for Apache HTTP Server.



#### New Web

Create Cancel

\* Indicates required field

\* Name:

\* Password:

\* Re-Enter Password:

Configuration: ☐ Local ☒ Centralized  
Where agent properties are stored. Local is the server on which the agent is running. Centralized is the OpenAM Server

\* Server URL:   
protocol://host:port/deploymentUri e.g. http://opensso.sample.com:58080/opensso

\* Agent URL:   
protocol://host:port e.g. http://agent1.sample.com:1234

3. Click Create to save the new web policy agent profile in OpenAM.

Next, install a policy agent in Apache HTTP Server to enforce your new policy.

Create a Dockerfile on your machine folder with the following contents:

```
FROM httpd:2.4.34

ENV PA_PASSWORD secret12

#Install pre-requisite packages
RUN echo "deb [trusted=yes] http://archive.kernel.org/debian-archive/debian/
jessie main" >> /etc/apt/sources.list

RUN apt-get update || true

RUN apt-get install -y curl unzip

#Install OpenAM Apache Agent
RUN curl -L -o /tmp/Apache_v24_Linux_64bit_4.1.1.zip
https://github.com/OpenIdentityPlatform/OpenAM-Web-
Agents/releases/download/4.1.1/Apache_v24_Linux_64bit_4.1.1.zip

RUN unzip /tmp/Apache_v24_Linux_64bit_4.1.1.zip -d /usr/

RUN rm /tmp/Apache_v24_Linux_64bit_4.1.1.zip

RUN echo $PA_PASSWORD > /tmp/pwd.txt

RUN cat /tmp/pwd.txt

RUN cat /etc/issue

#Configure OpenAM Apache Agent
RUN /usr/web_agents/apache24_agent/bin/agentadmin --s
"/usr/local/apache2/conf/httpd.conf" "http://openam.example.com:8080/openam"
"http://example.com:80" "/" "WebAgent" "/tmp/pwd.txt" --acceptLicence
--changeOwner
```

Build Apache Docker image with the preconfigured OpenAM Apache Policy Agent.

```
docker build --network=host -t apache_agent -f Dockerfile .
```

Run the image:

```
docker run -it --name apache_agent -p 8000:80 -h www.example.com --shm-size 2G
--network openam-quickstart apache_agent
```

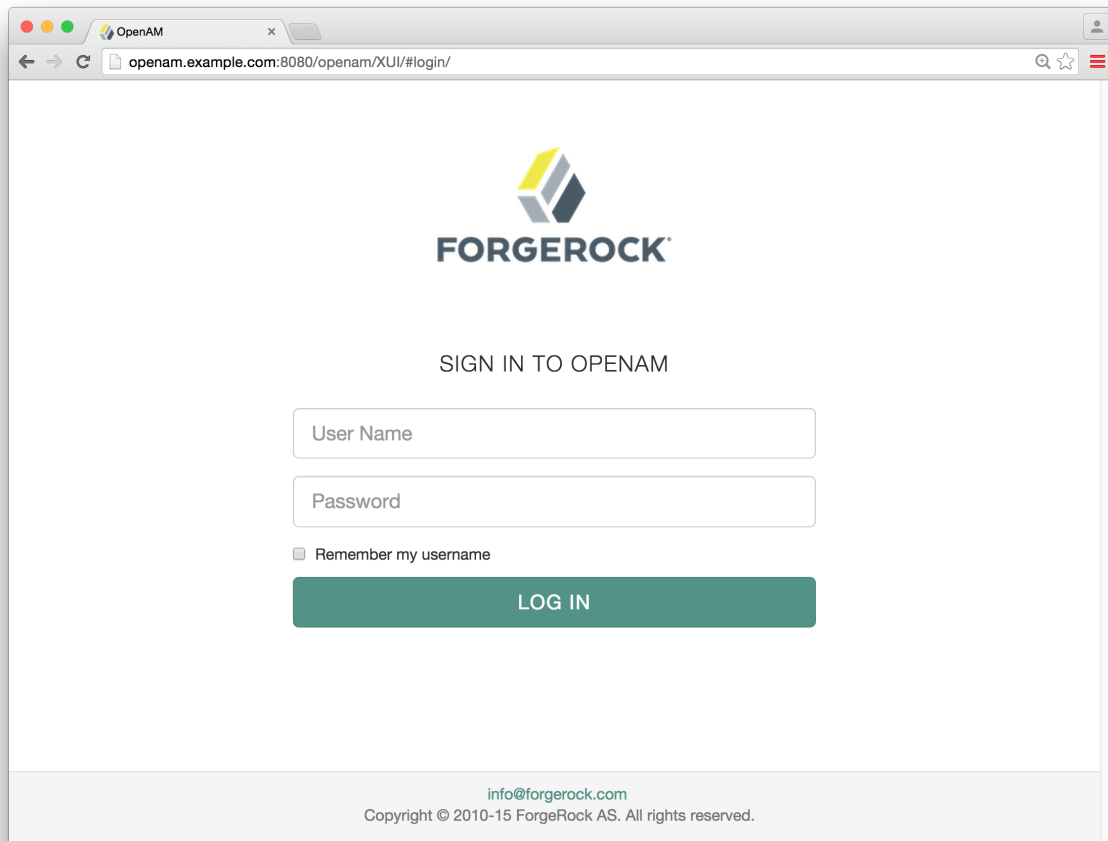
=== Trying It Out

Now that you have completed the steps above, you can access the protected web page to see OpenAM at work.

1. Log out of OpenAM Console.
2. Browse to <http://www.example.com:8080> to attempt to access the Apache "It works!" page.

At this point, the policy agent intercepts your request for the page. Your browser does not return a cookie indicating an OpenAM session, so the policy agent redirects you to OpenAM to authenticate.

3. Log in as the built-in default OpenAM demonstration user **demo** with password **changeit**.



On successful login, OpenAM sets a session cookie named **iPlanetDirectoryPro** in your browser for the domain **.example.com**. The cookie is then returned to servers in the **example.com** domain, such as **openam.example.com** and **www.example.com**.

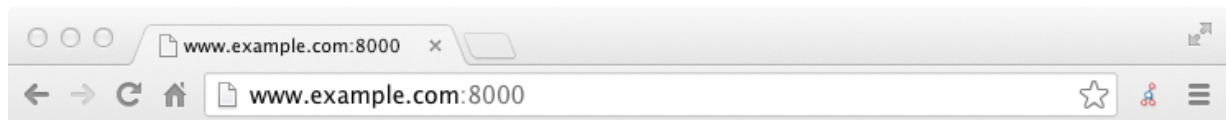
If you examine this cookie in your browser, you see that it has a value, such as **AQIC5wM2LY4SfcwciyfvJcQDUIB7kIWEH187Df\_txqLdAVc.AAJTSQACMDEAA1NLABMxMDYwNzY1MjQ0NTE0ODI2NTkx**. This is the SSO Token value. The value is in fact an encrypted reference to the session that is stored only by OpenAM. So, only OpenAM can determine whether you are actually logged in, or instead, that the session is no longer valid and you need to authenticate again.

The OpenAM session is used for SSO. When the browser presents the cookie to a server in the domain, the agent on the server can check with OpenAM using the SSO Token as a reference to the session. This lets OpenAM make policy decisions based on who is

authenticated, or prompt for additional authentication, if necessary.

Your SSO session can end in a few ways. For example, when examining the cookie in your browser, you should notice that it expires when the browser session ends (when you shut down your browser). Alternatively, you can log out of OpenAM explicitly. Sessions can also expire. OpenAM sets two limits, one that causes your session to expire if it remains inactive for a configurable period of time (default: 30 minutes), and another that caps the session lifetime (default: 2 hours).

4. After successful login, you are redirected to the Apache "It works!" page.



## It works!

In the background, OpenAM redirected your browser again to the page you tried to access originally, <http://www.example.com:8000>. This time, the web policy agent intercepted the request and found the SSO Token so it could request a policy decision from OpenAM regarding whether the user with the SSO Token has access to get <http://www.example.com:8000/>. OpenAM replied to the policy agent that it could allow access, and the policy agent allowed Apache HTTP Server to send back the web page.

Congratulations on protecting your first web site with OpenAM! Notice that you had only to install software and to configure OpenAM. You did not have to change your web site at all in order to add SSO and to set up access policies.

OpenAM can do much more than protect web pages. Read the next chapter to learn more.

### === Trying Out Stateless Sessions

In the ["Trying It Out"](#) section, you successfully configured OpenAM and viewed the [iPlanetDirectoryPro](#) session cookie. The session cookie contains information for OpenAM or a policy agent to locate the session data object on the server from which the session originated. Sessions that are stored in a server's memory are called *stateful*, which is the default configuration at the realm level.

OpenAM also supports *stateless* sessions, in which the authenticated user's session is stored on the client-side (for example, in a browser), not in memory. The session cookie cannot be updated until the session ends, when the user logs out or the session expires.

To try out stateless sessions, see ["Configuring Session State"](#) in the *Administration Guide*.

### == Where To Go From Here

OpenAM can do much more than protect web pages. In addition to being the right foundation for building highly available, Internet-scale access management services, OpenAM has a rich set of features that make it a strong choice for a variety of different deployments. This chapter presents the key features of OpenAM and indicates where in the documentation you can find

out more about them.

### === User Self-Service Features

OpenAM provides user self-registration and password reset services that allow users access to applications without the need to call your help desk.

OpenAM has access to the identity repositories where user profiles are stored. OpenAM is therefore well placed to help you manage self-service features that involve user profiles.

- **User Self-Registration.** OpenAM provides user self-registration as a feature of OpenAM's REST APIs. New users can easily self-register in OpenAM without assistance from administrators or help desk staff.

For information on configuring self-registration, see ["Configuring User Self-Registration"](#) in the *Administration Guide*.

For details on building your own self-registration application using the REST API, see ["Registering Users"](#) in the *Developer's Guide*.

- **Password Reset.** With OpenAM's self-service password reset, users can help reset passwords, as well as update their existing passwords. OpenAM handles both the case where a user knows their password and wants to change it, and also the case where the user has forgotten their password and needs to reset it, possibly after answering security questions.

For details on setting up password reset capabilities, see ["Configuring the Forgotten Password Reset Feature"](#) in the *Administration Guide*.

For details on building your own application to handle password reset using the REST API, see ["Retrieving Forgotten Usernames"](#) in the *Developer's Guide*.

- **Dashboard Service.** Users often have a number of applications assigned to them, especially if your organization has standardized SaaS, for example for email, document sharing, support ticketing, customer relationship management, web conferencing, and so forth. You can create an interface for users to access these web-based and internal applications using OpenAM's dashboard service.

The OpenAM cloud dashboard service makes this relatively easy to set up. For basic information on using the service, see ["Configuring the Dashboard Service"](#) in the *Administration Guide*.

OpenAM's user-facing pages are fully customizable and easy to skin for your organization. The [Installation Guide](#) has details on how to customize user-facing pages.

### === Single Sign-On

Single sign-on (SSO) is a core feature of OpenAM. Once you have set up OpenAM, you protect as many applications in the network domain as you want. Simply install policy agents for the additional servers, and add policies for the resources served by the applications. Users can authenticate to start a session on any site in the domain and stay authenticated for all sites in

the domain without needing to log in again (unless the session ends, or unless a policy requires stronger authentication. For details, see ["Configuring Single Sign-On Within One Domain"](#) in the *Administration Guide*.

Many organizations manage more than one domain. When you have multiple distinct domains in a single organization, cookies set in one domain are not returned to servers in another domain. In many organizations, sub-domains are controlled independently. These domains need to be protected from surreptitious takeovers like session cookie hijacking. OpenAM's cross-domain single sign-on (CDSSO) provides a safe mechanism for your OpenAM servers in one domain to work with policy agents from other domains, while allowing users to sign-on once across many domains without needing to reauthenticate. CDSSO allows users to sign on in one of your domains and not have to sign on again when they visit another of your domains.

CDSSO works through cooperation between policy agents and the `CDCServlet` in OpenAM. Together, the policy agents and OpenAM use federation capabilities to translate from one domain to another. For details on how to configure policy agents for CDSSO, see ["Configuring Cross-Domain Single Sign-On"](#) in the *Administration Guide*.

CDSSO only works with *stateful* sessions. CDSSO does not work with *stateless* sessions.

### === Standards-Based Federation

When you need to federate identities across different domains and different organizations with separate access management solutions, then you need interoperable federation technologies. Perhaps your organization acts as an identity provider for other organizations providing services. Perhaps you provide the services and allow users to use their identity from another organization to access your services. Either way, OpenAM has the capability to integrate well in federated access management scenarios. OpenAM supports standards-based federation technologies.

- Security Assertion Markup Language (SAML) 2.0 grew out of earlier work on SAML v1.x and the Liberty Alliance. SAML defines XML-based, standard formats and profiles for federating identities. SAML v2.0 is supported by a wide range of applications including major software as a service (SaaS) offerings. OpenAM supports SAML v2.0 and earlier standards, and can function as a hub in deployments where different standards are used. For details on OpenAM's SAML v2.0 capabilities, see ["Managing SAML v2.0 Federation"](#) in the *Administration Guide*.

When your deployment serves as an identity provider for a SAML federation, OpenAM makes it easy to develop applications called Fedlets that your service providers can easily deploy to participate in the federation. For details see ["Building SAML v2.0 Service Providers With Fedlets"](#) in the *Developer's Guide*.

- OAuth 2.0 and OpenID Connect 1.0 are open standards for authorization using REST APIs to allow users to authorize third-party access to their resources. These standards make it easier to federate modern web applications. OAuth for example is widely used in social applications.

OpenAM offers support for both OAuth 2.0 and OpenID Connect 1.0. OpenAM can serve as an authorization server and as a client of OAuth 2.0, while managing the profiles of the resource owners. When acting as a client, OpenAM policy agents can be used on resource servers to enforce authorization. For details, see "[Managing OAuth 2.0 Authorization](#)" in the *Administration Guide*.

OpenAM can serve as the OpenID Connect 1.0 provider with support for Basic and Implicit client profiles as well as discovery, dynamic registration, and session management. For details, see "[Managing OpenID Connect 1.0 Authorization](#)" in the *Administration Guide*.

### === Access Policies

In the first chapter of this guide you created an OpenAM access policy and saw how it worked. OpenAM can handle large numbers of access policies, each of which gives you control over user provisioning and user entitlements. For details, see "[Defining Authorization Policies](#)" in the *Administration Guide*.

OpenAM also supports standards-based access policies defined using the eXtensible Access Control Markup Language (XACML). XACML defines an XML Attribute-Based Access Control (ABAC) language with Role-Based Access Control (RBAC) features as well. For details on using XACML policies with OpenAM, see "[Importing and Exporting Policies](#)" in the *Administration Guide*.

### === Protect Any Web Application

In the first chapter of the guide you installed a web policy agent to enforce OpenAM's authorization decisions on Apache HTTP Server. That web policy agent is only one of many policy agents that work with OpenAM. "[Configuring Policy Agent Profiles](#)" in the *Administration Guide* describes policy agents for different web servers, for a variety of Java EE web application containers, for protecting SOAP-based web services, and for OAuth 2.0 clients.

For details about web policy agents also see the [Web Policy Agent User's Guide](#).

For details about Java EE policy agents also see the [Java EE Policy Agent User's Guide](#).

Furthermore [OpenIG Identity Gateway](#) works with applications where you want to protect access, but you cannot install a policy agent. For example, you might have a web application running in a server for which no policy agent has been developed. Or you might be protecting an application where you simply cannot install a policy agent. In that case, OpenIG functions as a flexible reverse proxy with standard SAML v2.0 capabilities. For details see the [OpenIG documentation](#).

### === Modern APIs For Developers

For client application developers, OpenAM offers REST, Java, and C APIs.

- OpenAM REST APIs make the common CRUD (create, read, update, delete) easy to use in modern web applications. They also offer extended actions and query capabilities for access management functionality.

To get started, see "[Using the REST API](#)" in the *Developer's Guide*.

- OpenAM Java APIs provided through the OpenAM Java SDK let your Java and Java EE applications call on OpenAM for authentication and authorization in both OpenAM and federated environments.

To get started, see ["Using the OpenAM Java SDK"](#) in the *Developer's Guide*.

- The OpenAM C SDK provides APIs for native applications, such as new web server policy agents. The C SDK is built for Linux, Solaris, and Windows platforms.

To get started, see ["Using the OpenAM C SDK"](#) in the *Developer's Guide*.

OpenAM provides built-in support for many identity repositories, web servers and web application containers, access management standards, and all the flexible, configurable capabilities mentioned in this chapter. Yet, for some deployments you might still need to extend what OpenAM's capabilities. For such cases, OpenAM defines Service Provider Interfaces (SPIs) where you can integrate your own plugins. For a list of extension points, and some examples, see ["OpenAM SPIs"](#) in the *Developer's Guide*.

=== Getting Help With Your Project

You can purchase OpenAM support subscriptions and training courses from an [Approved Vendor](#).