# SecureLogin 9

## Advanced Edition Installation and Configuration Guide

**June, 2021**

## Legal Notice

# Contents

# About This Book

This manual provides information about deploying SecureLogin Advanced Edition.

## Additional Documentation

For the latest version of this guide and other SecureLogin documentation resources, see the SecureLogin Documentation and keep up to date on patches and versions of both SecureLogin and the host operating system.

## Contact Information

We want to hear your comments and suggestions about this book and the other documentation included with this product. You can use the **comment on this topic** link at the bottom of each page of the online documentation, or send an email to Documentation-Feedback@microfocus.com.

For specific product issues, contact Micro Focus Customer Care at https://www.microfocus.com/support-and-services/.

# 1 Introduction

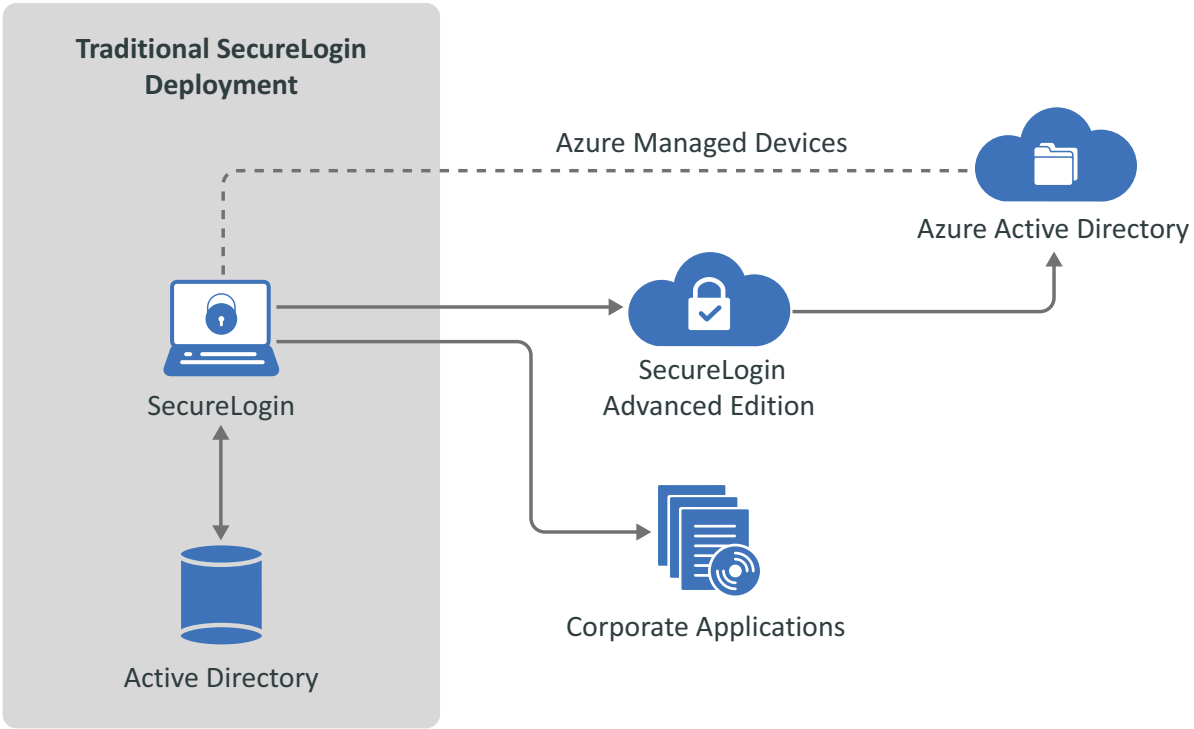
SecureLogin Advanced Edition, hereinafter called the Advanced Edition, is a new SecureLogin server introduced to use Azure Active Directory (Azure AD) as an identity provider. As organizations are moving towards cloud-based identity providers, Advanced Edition is developed to support Azure AD. Advanced Edition will also be capable of supporting other cloud-based identity providers in the future. Using Advanced Edition, you can seamlessly migrate from an Active Directory environment to an Azure AD environment.

Advanced Edition is also useful when you are deploying SecureLogin for the first time and want to use Azure AD as the identity provider.

The SecureLogin client communicates with Advanced Edition for single sign-on (SSO) data storage and retrieval. When a user creates or modifies the SSO data, SecureLogin stores the SSO data in Advanced Edition. Advanced Edition stores the SSO data in a SQL database (PostgreSQL) and manages it.

The following diagram shows the Advanced Edition architecture:

***Figure 1-1*** *SecureLogin Advanced Edition Architecture*




Advanced Edition does not support the following features:

- Windows Hello for Business authentication
- Integration with Privileged Account Management
- Integration with Advanced Authentication

- Desktop Automation Service
- Citrix environment
- PKI smart card support
- Active Directory Lightweight Directory Services (AD LDS) to Advanced Edition migration

# 2 Understanding Docker and Kubernetes



## 2.1 What Is Docker

Docker is a tool that uses OS-level virtualization to deliver software in packages called containers. Containers make it easy to create, deploy, and run applications. Containers are isolated from one another and bundle their software, libraries, and configuration files. They can communicate with each other through well-defined channels.

Docker benefits system administrators as it provides flexibility and reduces the number of systems needed because of its small footprint and lower overhead. The Advanced Edition containers can be deployed quickly on Kubernetes services provided by Azure, thereby reducing downtime. The Advanced Edition containers are also highly scalable to meet your requirements.

Kubernetes is a container orchestration engine for Docker. Advanced Edition is deployed on Kubernetes clusters to automate the manual processes of deploying and managing the Advanced Edition containers.

### Advantages of Docker

Enterprise software built as Docker containers is easier to assemble and maintain. Some of the parameters that Docker containers enable are:

- Isolation
- Portability
- Orchestration and Scaling

**Isolation:** Docker container isolates the applications from one another, and also from the primary system. This makes the software stack clean, and it becomes easier to monitor an application's usage of system resources, such as CPU, memory, and networking.

**Portability:** Docker containers can run on any system that supports the container’s runtime environment.

**Orchestration and Scaling:** You can deploy many Advanced Edition containers on a single system as the containers are lightweight. You can also have multiple systems with containers. Containers can also be used to scale SecureLogin components across clusters of systems when required.

## 2.2 Understanding the Basic Terminologies of Kubernetes

### 2.2.1 Kubernetes Keywords and Their Usage in SecureLogin

The following table includes the basic terminologies that are used in the proceeding sections.

| Term | Description |
|---|---|
| Container | An executable image that contains the Advanced Edition component and all of its dependencies. |
| Pod | A pod consists of a set of running Advanced Edition containers that share the same networking and storage resources. |
| Worker Node or Node | A worker node is a virtual or physical machine where the Advanced Edition pod runs. |
| Master Node | The master node controls and manages the worker node(s). |
| Kubernetes Cluster | A set of master and worker nodes to run the Advanced Edition pods. |
| Kubelet | An agent that runs on each node in the cluster to ensure that the containers are running in a pod. |
| Kube-scheduler | Scheduling means assigning a pod to a node. Kube-scheduler is the default Kubernetes scheduler that finds and assigns the optimal node for every newly created pod. It also assigns node for any other unscheduled pod. Kubelet runs the pods in a node. For more information, see Kubernetes Scheduler. |
| Namespace | A virtual Kubernetes cluster. For more information, see Namespaces. |
| Release | An instance of a chart that is running in a Kubernetes cluster. You can install the same chart multiple times to create many releases. |
| Ingress | Ingress manages the external access to the Advanced Edition service in a Kubernetes cluster. For more information, see Ingress. |
| Ingress Controller | Ingress controller makes the Ingress resources to work. For more information, see Ingress Controllers. |

### 2.2.2 Helm Charts

Helm is a package manager for Kubernetes. The Helm packaging format is called Charts or Helm Charts. Using Helm, you can deploy, configure, and upgrade Advanced Edition on Kubernetes clusters. Helm provides this functionality through a command-line tool called Kubectl.

The Advanced Edition Helm chart defines several Kubernetes resources as a set. The default Advanced Edition chart contains a minimum of a deployment template and a service template. This reduces the number of Kubernetes commands that you need to run to create and configure resources.

The following table describes the files and directories of Helm:

| File or Directory | Description |
|---|---|
| Helm Charts or Charts | A collection of YAML template files that describe the Kubernetes resources. |
| Chart.yaml | A YAML file that contains general information about the Advanced Edition chart such as chart name and version, version number, and search keywords. |

| File or Directory | Description |
| --- | --- |
| values.yaml | A YAML file that contains the default Advanced Edition configuration values for the chart. As per your requirement, modify this file before installing Advanced Edition. |
| charts/ | This directory contains the Advanced Edition charts |
| templates/ | This directory contains the following template files that are combined with configuration values and rendered into Kubernetes manifests:<br>◆ image-secret.yml<br>◆ ingress-http-rule.yml<br>◆ ingress-https-rule.yml<br>◆ NOTES.txt |
| templates/NOTES.txt | A text file which prints to a user's terminal when the user installs the chart. This file contains the following post-installation information:<br>◆ Command to check the status of the components. |

# 3 Planning SecureLogin Advanced Edition Deployment

Advanced Edition is deployed through a containerized mechanism. The Advanced Edition server is delivered as a Docker image that can be deployed on a Kubernetes cluster configured on Microsoft Azure. For more information about Docker and Kubernetes, see Understanding Docker and Kubernetes.

---

**IMPORTANT:** Before you start the Advanced Edition installation, ensure to have expert knowledge in the following areas:

- Docker
- Deployment and administration of Azure Kubernetes cluster
- Installation and administration of PostgreSQL

---

This section includes the following topics:



## 3.1 Requirements for Deploying Advanced Edition

You must meet the following requirements before deploying Advanced Edition:

- ❒ **Hardware Requirements:** Ensure that your setup meets the hardware requirements listed in NetIQ SecureLogin System Requirements.
- ❒ **Azure Kubernetes Environment Requirements:** See Azure Kubernetes Environment Requirements.
- ❒ The Windows device joined to Azure Active directory. For information about how to join a device to Azure Active Directory, see Microsoft Documentation.

  ---

  **NOTE:** Deploying Advanced Edition is supported only on an Azure AD joined Windows 10 device.

  ---

- ❒ PostgreSQL v10 or later installed.

  Run the following command to list the PostgreSQL pod. The status must be `Running`.

  ```
  kubectl get pods -n <Namespace>
  ```

```
dinesh@Azure:~$ kubectl get pods -n nsl-dinesh3
NAME                                                READY   STATUS    RESTARTS   AGE
cert-manager-69bf4b9b75-7ph2k                       1/1     Running   0          13d
cert-manager-cainjector-55db655cd8-8vhsw            1/1     Running   2          13d
cert-manager-webhook-78c5f4464c-5tx79               1/1     Running   0          13d
nginx-ingress-ingress-nginx-controller-85484674f4-5q7rr   1/1     Running   0          13d
nginx-ingress-ingress-nginx-controller-85484674f4-xjpfr   1/1     Running   0          13d
postgres-55fd48d58d-vk4bm                           1/1     Running   0          11d
slserver-6ccb57598b-jt4lh                           1/1     Running   0          4d22h
slserver-6ccb57598b-njnrg                           1/1     Running   0          4d22h
```

**IMPORTANT:** Ensure that PostgreSQL is installed and configured before installing Advanced Edition. It can be installed on the same AKS cluster on which you will deploy Advanced Edition or on a different AKS cluster.

Also, you must create the database name. Run the following command:

**To connect to the database:** `psql "host=<your_host> port=<your_port> dbname=<your_database> user=<your_username> password=<your_password> sslmode=require"`

**To create the database:** `CREATE DATABASE <DB-name>;`

The DB name must be in lowercase. For example, securelogin_top_db.

**NOTE:** Ensure that the PostgreSQL server supports the SSL connection.

❒ The users role assignment is configured. See Enabling Role-based Access and Token Validation.

## Azure Kubernetes Environment Requirements

Perform the following steps to set up an Azure Kubernetes environment to install Advanced Edition:

1 Create a Resource group.

For information about creating a Resource group, see Create Resource Groups.

2 Deploy an AKS cluster (v1.19 or later) and connect to the cluster.

**NOTE:** The OS type of the Node pool must be Linux.

**NOTE:** Make a note of the `region`. You will need this information while Installing Advanced Edition on Azure Kubernetes Services.

Run the following command to configure kubectl to connect to your Kubernetes cluster:

```
az aks get-credentials
```

The following command downloads credentials and configures the Kubernetes CLI to use them:

```
az aks get-credentials --resource-group myResourceGroup --name myAKSCluster
```

Run the following command to verify the connection to your cluster to return a list of the cluster nodes:

```
kubectl get nodes
```

The status of cluster nodes must be ready as shown in the following snippet:

```
dinesh@Azure:~$ kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-nslpool-29196012-vmss000000     Ready    agent   12d   v1.19.11
aks-nslpool-29196012-vmss000001     Ready    agent   12d   v1.19.11
```

For information about deploying AKS cluster, see Deploy an AKS cluster. For information about using AKS, see Best practices for using AKS.

3 Create a Namespace.

On Cloud Shell, run command `kubectl create namespace <name-of-the-namespace>`.

---

**NOTE:** Wherever applicable, use the same namespace.

---

4 Create an Ingress Controller as load balancer.

Ingress manages the external access to the Advanced Edition services in a Kubernetes cluster. To make the Ingress resources work, you need an Ingress controller. Advanced Edition uses the NGINX Ingress controller.

For information about creating an Ingress Controller, see Create an Ingress Controller.

---

**NOTE:** The following conditions must be true while creating the ingress controller:

- Use the namespace that you created in Step 3.
- DNS label must be in lowercase. Foe example, nsl-demo

---

Run the following command to list Ingress Controller pods:

```
kubectl get pods -n <Namespace>
```

The number of pods depends on the number of replica count configured during its deployment. For example, if the replica count is 2, you will see two pods with status Running.

```
dinesh@Azure:~$ kubectl get pods -n nsl-dinesh3
NAME                                                READY   STATUS    RESTARTS   AGE
nginx-ingress-ingress-nginx-controller-85484674f4-5q7rr   1/1     Running   0          12d
nginx-ingress-ingress-nginx-controller-85484674f4-xjpfr   1/1     Running   0          12d
```

5 Install Certificate Manager and CA Cluster Issuer. For more information, see Install a Certificate Manager.

Run this command to list Certificate Manager pods. It lists different components of Certificate Manager (cert-manager, cert-manager-cainjector, and cert-manager-webhook) with status Running.

```
kubectl get pods -n <Namespace>
```

```
dinesh@Azure:~$ kubectl get pods -n nsl-dinesh3
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-69bf4b9b75-7ph2k              1/1     Running   0          12d
cert-manager-cainjector-55db655cd8-8vhsw   1/1     Running   2          12d
cert-manager-webhook-78c5f4464c-5tx79      1/1     Running   0          12d
```

Certificate Manager requires an issuer or cluster issuer resource to issue certificates. Perform the following steps to create a CA cluster issuer:

1. Create the `cluster-issuer.yaml` file and add the following content:

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
name: letsencrypt
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: john.doe@example.com
    privateKeySecretRef:
      name: letsencrypt
    solvers:
    - http01:
        ingress:
          class: nginx
          podTemplate:
            spec:
              nodeSelector:
                "kubernetes.io/os": linux
```

**NOTE:** Ensure that the email ID is valid as the certificate expiration notifications will be sent to the same address.

2. Run the following command to create the cluster-issuer:

   `kubectl apply -f cluster-issuer.yaml -n <namespace>`

   Run the following command to list cluster-issuer. It lists the cluster-issuer with name `letsencrypt`.

   `kubectl get clusterissuer -n <Namespace>`

```
dinesh@Azure:~$ kubectl get clusterissuer -n nsl-dinesh3
NAME          READY   AGE
letsencrypt   True    12d
```

**NOTE:** To use your certificate instead of the default one, see Using Your CA Signed Certificate.

6 Create an Azure Container Registry (ACR).

ACR is a private Docker registry in Azure. You can store and manage Advanced Edition container images and related artifacts in ACR. For information about creating ACR, see Create an ACR.

## Enabling Role-based Access and Token Validation

Before you deploy Advanced Edition, you must define roles for users, such as admin and user roles. You can then assign a user to a particular job function or set of permissions to control access. The role assignment enables you to control who can access what content and functionality.

For example, an administrator can view the details of other users. However, a user cannot view the details of another user.

Perform the following steps to enable role-based access and token validation:

1 Log in to the Microsoft Azure portal.

2 Register an application with Azure AD and create a service principal.

A service principal is an identity created for use with applications, hosted services, and automated tools to access Azure resources.

Registering the application establishes a trust relationship between your app and the Microsoft identity platform.

For more information, see Register an application with Azure AD and create a service principal.

---

**NOTE:** While registering an application with Azure AD:

- Select **Account in this organizational directory only** for **Who can use the application?**.
- Ignore the **Redirect URI (optional)**.
- Create a new application secret and make a note of the Application (client ID), Client secret, and Directory (tenant) ID. You will need this information while Configuring Identity Store.

---

3 Under the client application, select **Expose an API**.

**3a** In **Application ID URI**, specify `api://{clientID}`.

**3b** Click **Add a scope** and create a scope named `All`.

- In **Who can consent**, select **Admins and Users**.
- Specify the display name as `All`.
- Set **State** to **Enabled**.

4 Select **Token configuration** and click **Add optional claim**.

**4a** Select **Access** > **aud** and **upn**.

**4b** Select **Turn on the Microsoft Graph profile permission (required for claims to appear in token)**.

**4c** Click **Add**.

5 Select **API permissions** and add the permissions as follows:

**5a** Click **Add a permission** > **Microsoft Graph** > **Delegated permissions**.

**5b** Select **User.Read.All** and **Group.Read.All** and then click **Add permissions**.

The following permissions are added by default:

- **User.Read**: It is added when you register an application.
- **Profile**: It is added when you configure an optional claim. See Step 4 on page 17.

---

**NOTE:** A global admin or an admin with rights to grant consent can approve these permissions.

---

**5c** Click **Add a permission** > **My APIs** > *your client application* > **Delegated permissions** > **All** and then click **Add permissions**.

6 Under your tenant, select **App registrations** > **Authentication**.

**6a** In **Platform configurations**, click **Add a platform** > **Mobile and desktop applications**, select all three suggested redirect URIs, and click **Configure**.

**6b** In **Supported account types**, select **Accounts in this organizational directory only**.

**6c** In **Advanced settings** > **Allow public client flows** > **Enable the following mobile and desktop flows**, select **Yes**.

**6d** Click **Save**.

**7** Select **App roles** and create admin and user roles.

To create the admin role:

1. Click **Create app role** and specify the following details:
   - **Display name**: Specify Admin.
   - **Allowed member types**: Select Both (Users/Groups + Applications).
   - **Value**: Specify admin.
   - **Description**: Specify a detailed description of the app role.
   - **Do you want to enable this app role?**: Select the option.
2. Click **Apply**.

To create the user role:

1. Click **Create app role** and specify the following details:
   - **Display name**: Specify User.
   - **Allowed member types**: Select Both (Users/Groups + Applications).
   - **Value**: Specify user.
   - **Description**: Specify a detailed description of the app role.
   - **Do you want to enable this app role?**: Select the option.
2. Click **Apply**.

**8** To set the token issuer to `https://login.microsoftonline.com/{tenantID}/v2.0`, perform the following steps:

**8a** Under your tenant, select **App registrations** > **Manifest**.

**8b** change the accessTokenAcceptedVersion param in the manifest JSON to 2:

```
"id": "b7b06474-8d47-429d-81e9-85090044c1b8",
 "acceptMappedClaims": null,
 "accessTokenAcceptedVersion": 2,
 "addIns": [],
```

**8c** Click **Save**.

**9** Under your tenant, select **Enterprise applications.**

**9a** Select your client application > **Assign users and groups**.

**9b** Assign roles (User or Admin) to the users and groups as required.

## 3.2 Deployment Scenarios

Advanced Edition installation is supported on the Microsoft Azure cloud. It supports the following deployment approaches. Use a deployment strategy that fits the needs of your company.

- Basic Deployment Flow
- Cloud-only Deployment
- Hybrid Deployment

### 3.2.1 Basic Deployment Flow

1. Log in to the Microsoft Azure portal.
2. Create an application. For example, app A.
3. Configure role-based access and token validation. See Enabling Role-based Access and Token Validation.
4. Take a note of client ID, client secret, tenant name, and tenant ID. Deploy the Advanced Edition server using these details. For more information, Installing Advanced Edition.

   **NOTE:** To get the tenant name, navigate to **Home** > **Subscription** > **Subscription name**. The value of **Directory** excluding the bracket is the tenant name. For example, `NSL Directory`.
5. Configure an identity store on the web console. For information about adding an identity store, see "Configuring Identity Store"in the *SecureLogin 9 Advanced Edition Installation and Configuration Guide*.
6. Ensure that your device is connected to Azure AD. For more information, see Microsoft documentation.
7. Install the SecureLogin client in the Advanced Edition mode. For information about the installation steps, see "Installing and Deploying in the SecureLogin Advanced Edition Environment" in the *SecureLogin 9.0 Installation Guide*.
8. Configure the passphrase for the new user. For more information, see "Setting Up a Passphrase" in the *SecureLogin 9.0 Installation Guide*.

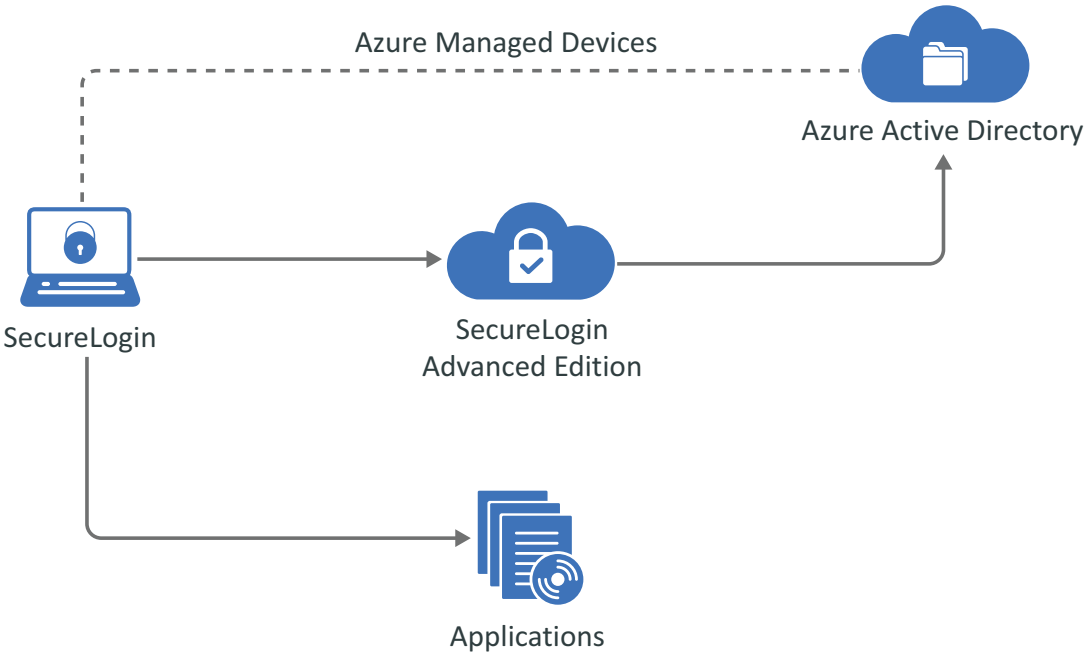
   A connection is established between the SecureLogin client and Advanced Edition.
9. To confirm that the connection is established, right-click the SecureLogin icon in the system tray, click **About**. The value for **Primary** must be **Primary-Available**.

### 3.2.2 Cloud-only Deployment

In this deployment model, Advanced Edition is deployed on the Azure cloud, user groups are managed in Azure AD, and the SecureLogin client is deployed in the Advanced Edition mode to manage the Azure AD joined devices.

This model suits best when you are deploying SecureLogin for the first time for your company and want to use Azure AD as the identity provider. Figure 3-1 illustrates this scenario.

***Figure 3-1*** *Advanced Edition Cloud-Only Deployment*




You can perform only a fresh installation in the cloud-only deployment model.

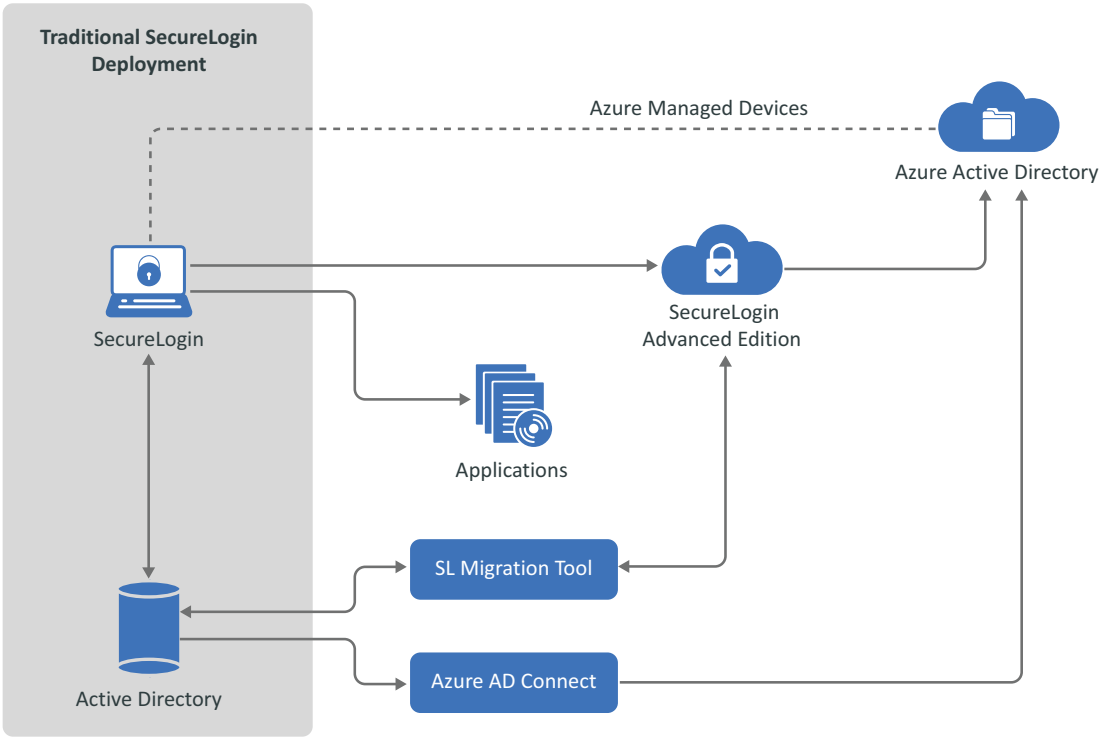
The high-level steps are as follows:

1. Install Advanced Edition on Azure Kubernetes Services. See Installing Advanced Edition.
2. Configure Advanced Edition. See Configuring Advanced Edition.
3. Install and configure the SecureLogin client in the Advanced Edition mode. See Installing and Deploying in the SecureLogin Advanced Edition Environment.

### 3.2.3 Hybrid Deployment

In the hybrid deployment model, Advanced Edition is deployed on the Azure cloud, user groups are managed in Active Directory, Azure AD or both, and the SecureLogin client is deployed in the Advanced Edition mode to manage the Azure AD joined devices. Figure 3-2 illustrates this deployment model.

***Figure 3-2*** *Advanced Edition Hybrid Deployment*




Using this model, you can deploy SecureLogin to achieve one of the following scenarios:

- Manage some user groups in Active Directory and some in Azure AD
- Manage all user groups in both Active Directory and Azure AD

## Manage some user groups in Active Directory and some in Azure AD

If you are already using SecureLogin in the Active Directory mode, you can migrate your user groups from Active Directory to Azure AD. You can choose to migrate only a certain user groups to Azure AD. The remaining user groups can be managed in Active Directory. This can be useful when you have many user groups, and the number of users in each user group is enormous. Migrating and managing all the user groups at once can be a complex task. Therefore, you can migrate the groups in phases.

## Manage all user groups in both Active Directory and Azure AD

This scenario is similar to the previous one. However, you can choose to maintain all your user groups in both Active Directory and Azure AD. This can be useful when the number of users and user groups are less.

The following are the high-level steps of the hybrid deployment model. You can perform a fresh installation or migration.

## Fresh Installation

Perform a fresh installation when you are deploying SecureLogin for the first time.

1. Install Advanced Edition on Azure Kubernetes Services. See Installing Advanced Edition.

2. Configure Advanced Edition. See Configuring Advanced Edition.
3. Sync the Active Directory users to Azure AD using the Azure AD connect tool. See Synchronizing Active Directory Users with Azure AD.
4. Install and configure the SecureLogin client in the Advanced Edition mode. See Installing and Deploying in the SecureLogin Advanced Edition Environment.

### Migration

Migration is applicable when you have an existing SecureLogin deployment.

1. Install Advanced Edition on Azure Kubernetes Services. See Installing Advanced Edition.
2. Configure Advanced Edition. See Configuring Advanced Edition.
3. Sync the Active Directory users to Azure AD using the Azure AD connect tool. See Synchronizing Active Directory Users with Azure AD.
4. Install and configure the SecureLogin client in the Active Directory mode. See Installing and Configuring in Active Directory Environment in the *SecureLogin 9.0 Installation Guide*.
5. Migrate the datastore using the slMigrationHelper tool. See Migrating the Data Through the slMigrationHelper Tool.

# 4 Installing Advanced Edition



## 4.1 Installing Advanced Edition on Azure Kubernetes Services

1 Ensure that you have completed the tasks mentioned in the Requirements for Deploying Advanced Edition section before beginning the Advanced Edition installation.

2 Download the Advanced Edition docker image and helm chart from Software Licenses and Downloads.

3 Unzip the docker image and helm chart.

4 Upload the Advanced Edition docker image to ACR:

**4a** Load the Advanced Edition docker image by running the following command:

```
docker load --input .\<name-of-the-Advanced-Edition-docker-image>
```

**4b** Tag the docker image by running the following command:

```
docker tag <source repo:tag> <acr-login-server>/<repository-name>:<tag>
```

For example, `docker tag security-securelogin-docker.btpartifactory.swinfra.net/sl_server:9.0.0.0-326 nslacr.azurecr.io/sl_server:9.0.0.0-326`

**4c** Push the docker image to the registry by running the following command:

```
docker push <acr_login-server>/< repository-name>:<tag>
```

For example, `docker push nslacr.azurecr.io/sl_server:9.0.0.0-326`

5 Create an image pull secret. For information, see Create an image pull secret.

---

**NOTE:** For higher security, use a text file containing the password as an argument to `docker-password` in the command. For example, `docker-password=$(< principal-password.txt)`.

---

6 On Cloud Shell, edit the values of the `SecureLogin-Server-x.x.x.x\values.yaml` file.

---

**IMPORTANT:** The PostgreSQL database must be installed before performing this step.

---

Specify or modify the following values:

| Section | Value |
|---|---|
| **image**<br>This section includes information about the docker image. | |
| repository | The container image repository to be used.<br>Path: <acr_login-server>/< repository-name><br>For example, `nslacr.azurecr.io/sl_server` |
| tag | The tag or version of the docker image.<br>For example, `9.0.0.0-326` |
| imagePullSecrets | Specify the image pull secret that you created in Step 5.<br>For example, `my-secret`<br>The image pull secret is used to pull images from ACR to the Kubernetes cluster. For more information, see Pull images from an Azure container registry to a Kubernetes cluster. |
| ingresshost | The host route for the ingress resource.<br><dns>.<cluster_region>.cloudapp.azure.com<br>For example, `nsl-dns.southeastasia.cloudapp.azure.com` |
| **serverAdmin**<br>This section creates the username and password of the Advanced Edition's administrator. | |
| secret | Specifies the name of the generic secret having the credentials.<br>For example, `my-k8s-secret` |
| username | Specifies the username of the administrator.<br>For example, `john_doe` |
| password | The password of the administrator. You must change the default value before deployment.<br>**NOTE:** To change the password for the first time, no need to change the value of secret. However, the next time onwards, you must change both password and secret under **serverAdmin**. |
| **DBProperties**<br>This section includes the database configuration details. | |
| secret | The secret name.<br>For example, `my-db-secret`<br>SecureLogin does not save username, password, host, port, and data base name into a text file. These are converted into a secret. The server pod uses this secret. All pods refer to this secret to use the same credentials. |

| Section | Value |
|---|---|
| username | The username of the PostgreSQL database. |
| password | The password of the PostgreSQL database. You must change the default value before deployment.<br>**NOTE:** To change this password for the first time, no need to change the value of secret. However, the next time onwards, you must change both password and secret in DBProperties. |
| host | The database's IP address or service name. |
| port | The port used by the database.<br>For example, 5432 |
| dbName | The database name.<br>For example, `securelogin_top_db` |

7 Install Advanced Edition by running the following command:

```
helm install <name-of-the-release> <name-of-the-helm-chart> -n <namespace>
```

For example, `helm install slserver001 SecureLogin-Server-x.x.x.x -n nsl-namespace`

where, `slserver001` is the release name, `SecureLogin-Server-x.x.x.x` is the name of the helm chart, and `nsl-namespace` is the name of the namespace.

8 (Optional) Replace the default certificate with a third-party certificate. For more information, see Using Your CA Signed Certificate.

**NOTE:** You can also perform this step after configuring Advanced Edition.

9 Configure Advanced Edition. Log in to the Advanced Edition web console at `https://<dns>.<cluster_geo_location>.cloudapp.azure.com`. For more information, see Configuring Advanced Edition.

You can view the Advanced Edition version on the web console by clicking <username> > About.

## 4.2 Understanding Helm Install, Upgrade, and Uninstall Actions

- Helm Install
- Helm Upgrade
- Helm Uninstall

### Helm Install

Use this action when you create SecureLogin server configuration for the first time. Run the following command to perform Helm installation:

```
helm install <name-of-the-release> <name-of-the-helm-chart> -n <namespace>
```

For example, `helm install slserver001 SecureLogin-Server-x.x.x.x -n nsl-namespace`

where, slserver001 is the release name, SecureLogin-Server-x.x.x.x is the name of the helm chart, and nsl-namespace is the name of the namespace.

---

**NOTE:** Helm uninstall and re-install is not recommended when a configuration change is needed. This process removes all the pods and service before a new instance is deployed.

---

### Helm Upgrade

HELM upgrade is recommended when you make any change in the configuration of the SecureLogin server. This process upgrades the replica of server instances one after another. This approach keeps the service available during the upgrade process.

Run the following command to perform Helm upgrade:

```
helm upgrade <release-name> <name-of-the-helm-chart> -n <name-of-the-namespace>
```

For example, `helm upgrade slserver SecureLogin-Server-x.x.x.x -n nsl-ingress`

---

**NOTE:** If you change a value, you must change the associated secret. When the old secret name exists in the current deployment, the change does not take place after the upgrade.

If you change a value and do not change the associated secret, the change will not take place after the upgrade.

For example, if you change the database credentials in the values.yaml file, you must change its secret name for the new value of credentials to take effect.

1. List all secret values for the current deployment.

   Use the following command to list all the secrets for the current deployment:

   kubectl get secret -n <ingress namespace>
2. If the DB secret name is already in use with the current deployment, modify the required DB credentials and its secret name.
3. Perform Helm upgrade.

---

### Helm Uninstall

Ensure that you have successfully uninstalled the helm release, before reinstalling. To list all the releases, use the following command:

```
helm list -n <namespace>
```

For example, `helm list -n nsl-ingress`

```
dinesh@Azure:~$ helm list -n nsl-ingress --short
cert-manager
nginx-ingress
postgres-db-chart
slaedinesh3
```

To uninstall a release, use the following command:

`helm uninstall <name-of-the-release> -n <namespace>`

## 4.3 Post-Installation Tasks

- Using Your CA Signed Certificate
- Modifying the Life Span of a JWT Token
- Changing the Administrator Username and Password

### 4.3.1 Using Your CA Signed Certificate

While installing Advanced Edition, SecureLogin uses *Let's Encrypt* to generate the certificate. You can replace the default certificate with a third-party certificate authority (CA) issued certificate, such as Verisign.

Perform the following steps:

1. Extract the private key (.key) from the certificate by using the following command:

   `openssl pkcs12 -in <name-of-the-certificate>.pem -out <name-of-the-key>.key -nodes -nocerts`

   For example, `openssl pkcs12 -in my-certificate.pem -out private.key -nodes -nocerts`

2. Extract the public certificate (.crt) from the certificate file.

   `openssl pkcs12 -in <name-of-the-certificate>.pem -out <name-of-the-public-certificate>.crt -nodes -nokeys`

   For example, `openssl pkcs12 -in my-certificate.pem -out pub-certificate.crt -nodes -nokeys`

3. Create a TLS secret using the public certificate and private key.

   `kubectl create secret tls <name-of-the-secret> --namespace <name-of-the-namespace> --key private.key --cert pub-certificate.crt`

   For example, `kubectl create secret tls my-tls-secret --namespace my-ingress --key private.key --cert pub-certificate.crt`

4. Open the `SecureLogin-Server-x.x.x.x\values.yaml` file and modify the values in the **certificate** section as follows:

| Field | Value |
|---|---|
| default | false |
| secret | The name of the secret. For example, my-tls-secret.<br>To configure the certificate for the first time, no need to change the value of secret. However, the next time onwards, you must change this value also. |

5 Perform a helm install or upgrade using the following command:

- To install:

  `helm install <name-of-the-release> <name-of-the-helm-chart> -n <name-of-the-namespace>`

  For example, `helm install slserver001 SecureLogin-Server-x.x.x.x -n nsl-namespace`

- To upgrade:

  `helm upgrade <release-name> <name-of-the-helm-chart> -n <name-of-the-namespace>`

  For example, `helm upgrade slserver SecureLogin-Server-x.x.x.x -n my-ingress`

## 4.3.2 Modifying the Life Span of a JWT Token

You can modify the life span of a JWT token using the following steps:

1 Open the `SecureLogin-Server-x.x.x.x\values.yaml` file.

2 In the **JWTToken** section, modify the following value:

expiration: Change to a preferred value. However, this value must be greater than inactivity timeout that is 15 min.

The expiration time is 60 minutes by default.

3 Save the file.

4 Perform a helm install or upgrade using the following command:

- To install:

  `helm install <name-of-the-release> <name-of-the-helm-chart> -n <name-of-the-namespace>`

  For example, `helm install slserver001 SecureLogin-Server-x.x.x.x -n nsl-namespace`

- To upgrade:

  `helm upgrade <release-name> <name-of-the-helm-chart> -n <name-of-the-namespace>`

  For example, `helm upgrade slserver SecureLogin-Server-x.x.x.x -n my-ingress`

### 4.3.3 Changing the Administrator Username and Password

1. Open the `SecureLogin-Server-x.x.x.x\values.yaml` file.
2. In the **serverAdmin** section, modify the following details:

| Section | Value |
|---|---|
| secret | To change the password for the first time, no need to change the value of secret. However, the next time onwards, you must change both **password** and **secret**. |
| username | Specify the new username. |
| password | Specify the new password. |

3. Save the file.
4. Perform a helm install or upgrade using the following command:
   - To install:

     `helm install <name-of-the-release> <name-of-the-helm-chart> -n <name-of-the-namespace>`

     For example, `helm install slserver001 SecureLogin-Server-x.x.x.x -n nsl-namespace`
   - To upgrade:

     `helm upgrade <release-name> <name-of-the-helm-chart> -n <name-of-the-namespace>`

     For example, `helm upgrade slserver SecureLogin-Server-x.x.x.x -n my-ingress`

# 5 Configuring Advanced Edition



## 5.1 Configuring Data Store

You can modify the database details in the `SecureLogin-Server-x.x.x.x\values.yaml` file. Configuring or modifying the datastore from the web console is not supported in this release.

1. Open the `SecureLogin-Server-x.x.x.x\values.yaml` file.
2. Under **DBProperties**, specify the following details:

| Field | Value |
|---|---|
| **secret** | The secret name.<br>For example, `my-db-secret`<br>SecureLogin does not save username, password, host, port, and data base name into a text file. These are converted into a secret. The server pod uses this secret. All pods refer to this secret to use the same credentials. |
| **username** | The username of the PostgreSQL database. |
| **password** | The password of the PostgreSQL database. You must change the default value after deployment.<br>**NOTE:** To change the password second time onward, you must also change the value of **secret** under **DBProperties**. |
| **host** | The database's IP address or service name. |
| **port** | The port number used by the database.<br>For example, 5432 |
| **dbName** | The database name.<br>For example, `securelogin_Top_db` |

3. Save the file.
4. Perform a helm install or upgrade using the following command:
   - To install:

     ```
     helm install <name-of-the-release> <name-of-the-helm-chart> -n
     <name-of-the-namespace>
     ```

For example, `helm install slserver001 SecureLogin-Server-x.x.x.x -n nsl-namespace`

- To upgrade:

  `helm upgrade <release-name> <name-of-the-helm-chart> -n <name-of-the-namespace>`

  For example, `helm upgrade slserver SecureLogin-Server-x.x.x.x -n my-ingress`

## 5.2 Configuring Audit Server

You can configure an external auditing server, such as a syslog server, for auditing and monitoring the Advanced Edition events.

You must configure the root certificate for the audit server in SecureLogin-Server-x.x.x.x\values.yaml before configuring the audit server on the web console. Advanced Edition supports only TCP connections using TLS1.2.

### Configuring the Root Certificate in values.yaml

1. Create a folder named `certs` inside the server folder of the helm charts.
2. Copy the audit server root certificate to the `certs` folder.
3. Open `SecureLogin-Server-x.x.x.x\values.yaml` and specify the following details in the **SSL** section:
   - **auditCert**: Specify the name of the audit server certificate that you copied to the `certs` folder.
   - **auditCertSecret:** To configure the certificate for the first time, no need to change the value of secret. However, the next time onwards, you must modify both **auditCert** and **auditCertSecret**.
4. Save the file.
5. Perform a helm install or upgrade using the following command:
   - To install:

     `helm install <name-of-the-release> <name-of-the-helm-chart> -n <name-of-the-namespace>`

     For example, `helm install slserver001 SecureLogin-Server-x.x.x.x -n nsl-namespace`
   - To upgrade:

     `helm upgrade <release-name> <name-of-the-helm-chart> -n <name-of-the-namespace>`

     For example, `helm upgrade slserver SecureLogin-Server-x.x.x.x -n my-ingress`

### Configuring the Audit Server on the Web Console

1. Log in to the Advanced Edition web console.

   URL: `https://<dns>.<cluster_geo_location>.cloudapp.azure.com`

2 Click **Audit Server** > **Configure Audit Server**.

3 Specify the following details:

| Field | Description |
|---|---|
| **Audit Server Name** | Specify a name for the audit server. |
| **IP address or Domain Name** | Specify the IP address of the audit server. |
| **Port Number** | Specify the port number of the audit server.<br>You must configure a TCP port. For example, 512 |
| **Format** | Only CEF format is supported. |

4 Click **Save**.

You can later modify these values if required.

## 5.3 Configuring Identity Store

Identity Store serves as an identity provider for the SecureLogin client and as an identity consumer for Advanced Edition.

Perform the following steps to configure Identity Store:

1 Log in to the Advanced Edition web console.

URL: `https://<dns>.<cluster_geo_location>.cloudapp.azure.com`

2 Click **Identity Store** > **Configure Identity Store**.

3 in the **Configure** tab, specify the following details:

| Field | Description |
|---|---|
| **Name** | Specify a name for Identity Store. |
| **Available Presets** | Select **Azure AD**.<br>**NOTE:** Only Azure AD preset is supported in this release. |
| **Client ID** | Specify the client ID that you created in the Azure portal. |
| **Client Secret** | Specify the client secret that you created in the Azure portal.<br>You can view the secret by clicking the eye icon. |
| **Tenant Name** | Specify the Azure AD tenant name. |
| **Tenant ID** | Specify the Azure AD tenant ID. |

4 In the **Advanced** tab, specify the following details:

| Field | Value |
|---|---|
| **Token Endpoint** | https://login.microsoftonline.com/$tenantid/oauth2/v2.0/token<br><br>SecureLogin uses the endpoint link to retrieve token and user details from Advanced Edition. |
| **Scope** | api://$clientid/All |
| **Key Field** | id |
| **Group filter starts with** | Specify the value based on which you want to filter and display the groups in SLManager. For example, if you specify `secure`, all groups starting with `secure` are displayed. |

5 Click **Save**.

You can later modify these values if required.

# 6 Installing SecureLogin Client in the Advanced Edition Environment

Install the SecureLogin client after you have installed and configured Advanced Edition. For information about installing the SecureLogin client, see "Installing and Deploying in the SecureLogin Advanced Edition Environment" in the *SecureLogin 9.0 Installation Guide*.

# 7 Migrating the SecureLogin Data from Active Directory to Azure AD

If your SecureLogin is installed in the Active Directory mode, you can migrate the datastore from Active Directory to Azure AD (Advanced Edition mode).

---

**IMPORTANT:** Before you migrate the data from Active Directory to Azure AD, you must synchronize Active Directory users with Azure Active Directory.

---

## Synchronizing Active Directory Users with Azure AD

1. On the Active Directory server, install Azure AD Connect.
2. Launch Azure AD Connect and perform the following steps to configure it:
   - **2a** Under **Tasks**, select **Configure device options**, and click **Next**.
   - **2b** Specify your Azure AD global administrator credentials.
   - **2c** Select **Configure Hybrid Azure AD join**.
   - **2d** Click **Next** > **Next** > **Next** > **Configure**.
3. Synchronize the password hash in Active Directory to Azure AD.
   - **3a** Launch Azure AD Connect.
   - **3b** In **Additional Tasks**, select **Customize synchronization options**.
   - **3c** Click **Next** > **Next** > **Next**.
   - **3d** In **Optional features**, select **Password hash synchronization**.
   - **3e** Click **Next** > **Configure** > **Exit**.

   For more information, see Microsoft Documentation.
4. Verify that the Active Directory users are displayed in the Azure portal after synchronization.
5. Join the client device to the hybrid Azure AD domain. On the hybrid Azure AD joined device, install SecureLogin in the AD mode.

   You must use the same user that was used to synchronize AD to Azure AD using Azure AD Connect.
6. Continue with Migrating the Data Through the slMigrationHelper Tool.

## Migrating the Data Through the slMigrationHelper Tool

---

**IMPORTANT:** Consider the following point when you migrate to Advanced Edition:

- You can migrate to Advanced Edition only from Active Directory or ADAM

- SLAESERVERADDRESS (server address of Azure AD) is required while migrating
- By default, the value of SLAEPORT is TLS 443

---

1. Go to the `SecureLogin\Tools\Administration\Provision Tools` folder.
2. Run the following command in the command prompt:

   ```
   slmigrationhelper.exe -m -t SLAE SLAESERVERADDRESS=10.198.1.2 /q
   ```

   The data is exported to `nslexport.xml` and saved in the `SecureLogin\Tools\Administration\Provision Tools` folder.
3. Specify the administration credentials. The system is restarted automatically.
4. Run the following command to import the data:

   ```
   slmigrationhelper.exe -i -f nslexport.xml -p
   ```
5. Restart the system.

For more information about migrating a datastore, see "Migrating the Datastore Using the slMigrationHelper Tool" in the *NetIQ SecureLogin 9.0 Administration Guide*.

# 8 Users and Groups Management in the Advanced Edition Mode

In the Advanced Edition mode, users and groups are well-organized in Azure AD to manage the SecureLogin data. Organizing users into a group makes it easier to manage settings and permissions. Settings include application definitions, preferences, credentials, and password policies. You can assign specific settings to members of a group at once instead of doing it for each member one-by-one. You can also manage the settings and permissions at the user-level.

SLManager displays the SecureLogin data in two categories as `Groups` and `Users`. You can manage various settings and permissions at the group level or user level based on your requirements.

This section includes the following topics:

- Managing Advanced Edition Users and Groups
- Understanding How Settings Inheritance Work for SecureLogin Groups and Users

## 8.1 Managing Advanced Edition Users and Groups

You can create groups and assign users to specific groups in Azure AD. SLManager lists all SecureLogin groups and users. You can apply SecureLogin settings to the groups or specific users in SLManager.

Perform the following steps in Azure AD:

1. Add groups based on your requirement.
2. Assign roles and API permissions for SecureLogin.

   The required permissions are User.Read, User.Read.All, and Group.Read.All. For more information, see Enabling Role-based Access and Token Validation.

   Roles can be user or admin. To assign a role, perform the following steps:

   **2a** Under your tenant, select **Enterprise applications.**

   **2b** Select your client application > **Assign users and groups**.

   **2c** Assign roles (User or Admin) to the users and groups as required.
3. Add SecureLogin users to the appropriate groups if you want to manage users at the group level.

   For information about how to create a group and add users to a group, see Microsoft documentation.

Perform the following steps in SLManager to manage the SecureLogin groups and users:

These steps are for managing settings at the group level. You can perform the similar steps to manage settings of an individual user.

1. Open SLManager.

2 Expand **Groups** and select the group for which you want to apply or modify settings.

3 Click **Distribution** > **Load**.

4 Select the settings you want to import and apply to the group:

| Setting | Description |
|---|---|
| **Applications** | Imports all configured application definitions. |
| **Credentials** | Imports all credentials excluding passwords. |
| **Password Policies** | Imports password policies. |
| **Preferences** | Imports manual preferences. |

5 Click **OK**.

6 Select the SecureLogin script (ESX) file containing the required settings.

7 Click **OK**.

The imported settings are applied to the group and all members. When you add a new user to this group later, the user inherits all these settings.

**NOTE:** User-level settings take precedence over the group settings.

# 8.2 Understanding How Settings Inheritance Work for SecureLogin Groups and Users

The following scenarios describe how settings are inherited when a user is a member of more than one group. Settings can include application definitions, preferences, credentials, and password policies.

- Scenario 1
- Scenario 2
- Scenario 3
- Scenario 4

### Scenario 1

When a user or admin is a member of two groups as follows:

- Group1: A security group with the user or admin role. Settings are applied to this group.
- Group2: A security group with the default user role. No specific settings are configured.

In this scenario, the user or admin inherits the settings of Group1 and the settings of Group2 are ignored.

### Scenario 2

When a user or admin is a member of two groups as follows:

- Group1: A security group with the user or admin role. Settings are applied to this group.
- Group2: A security group without SecureLogin application and roles. No settings are applied.

In this scenario, the user inherits the settings of Group1 and the settings of Group2 are ignored.

### Scenario 3

When a user or admin is a member of two groups and both groups are with the user or admin roles. Settings are applied to both groups.

In this scenario, the user or admin does not inherit any settings of Group1 and Group2. You must be careful of this scenario while adding users to multiple groups.

### Scenario 4

When a user or admin is a member of a nested group. For example, Group2 is a member of Group1. And the user is a member of Group2.

In this scenario, Group2 inherits the settings of Group1. However, the user inherits settings of only Group2.

For example, Group1 has two single sign-on applications app1 and app2. You apply setting on Group2 that adds app3 and app4. After inheriting settings from Group1, Group2 has four applications: app1, app2, app3, and app4. However, the user inherits only app3 and app4.

Azure AD does not support this configuration. When you try to add a member in a nested group, the following message is displayed:

```
When you assign a group to an application, only users in the group will
have access. The assignment does not cascade to nested groups.
```

For more information, see Manage user assignment for an app in Azure Active Directory.

# 9 Troubleshooting



## 9.1 Primary Advanced Edition Is Not Available

Perform the following actions:

1. Check if Identity Store is configured in the web console.
2. Check that the values provided during the configuration of Identity Store (Client id, Client Secret, Tenant name, Tenant id) in the web console match with your App Registration.
3. Verify that the appropriate API permissions are added and granted to your application. If you do not provide proper permissions to your App Registration, then Azure AD fails to authenticate the user.
4. Check if that specific logged-in user or a group is assigned to a specified App Role.

## 9.2 The Site Is Not Reachable After Installing Advanced Edition

Ensure that the correct DNS label is assigned to the IP address. You need to change it manually if it is referring to an incorrect value.

1. In the Azure portal, go to your nodeResourceGroup (MC_<resource-group>_<cluster>_<location>).
2. Open the nginx-ingress-ingress-nginx-controller LoadBalancer IP address > **Configuration**.

   **NOTE:** You can list the Kubernetes IP addresses by using the `kubectl get pods -n <ingress-namespace>` command.
3. Specify the **DNS name label**.
4. Click **Save**.

## 9.3 Changing the Database Configuration

You can specify the password of the database in the `values.yaml` file.

To change the password of the server, perform the following steps:

1. Edit the `SecureLogin-Server-x.x.x.x\values.yaml` file.
2. Change the database name, username, or password as needed under DBProperties. The values must match with the database configuration.
3. Perform a helm install or upgrade using the following command:
   - To install:

     `helm install <name-of-the-release> <name-of-the-helm-chart> -n <name-of-the-namespace>`

     For example, `helm install slserver001 SecureLogin-Server-x.x.x.x -n nsl-namespace`
   - To upgrade:

     `helm upgrade <release-name> <name-of-the-helm-chart> -n <name-of-the-namespace>`

     For example, `helm upgrade slserver SecureLogin-Server-x.x.x.x -n my-ingress`

## 9.4 Scaling the Node Manually

Perform the following steps to scale the nodes:

1. Open the `SecureLogin-Server-x.x.x.x\values.yaml` file and specify the number of nodes required in **replicaCount**.
2. Save the file.
3. Perform a helm install or upgrade using the following command:
   - To install:

     `helm install <name-of-the-release> <name-of-the-helm-chart> -n <name-of-the-namespace>`

     For example, `helm install slserver001 SecureLogin-Server-x.x.x.x -n nsl-namespace`
   - To upgrade:

     `helm upgrade <release-name> <name-of-the-helm-chart> -n <name-of-the-namespace>`

     For example, `helm upgrade slserver SecureLogin-Server-x.x.x.x -n my-ingress`

## 9.5 Debugging Pods

| Scenario | Use the Command |
|---|---|
| To view the status of pods | `kubectl get pods -n <ingress-namespace> -w`<br>Example: `kubectl get pods -n nsl-ingress -w` |
| To view the detailed info about the pods | `kubectl describe pod <slserver-podname> -n <ingress-namespace>`<br>Example: `kubectl describe pod slserver-59484f68d-6dkvl -n nsl-ingress` |
| To delete the pod | `kubectl delete pod <slserver-podname> -n <ingress-namespace>`<br>Example: `kubectl delete pod slserver-59484f68d-6dkvl -n nsl-ingress` |
| To list the helm charts | `helm list -n <ingress-namespace>`<br>Example: `helm list -n nsl-ingress` |
| To check the status of helm (that was used to deploy Advanced Edition) | `helm status <chart-name> -n <ingress-namespace>`<br>Example: `helm status slserver001 -n nsl-ingress` |
| To upgrade helm (to deploy updated docker images) | `helm upgrade <release-name> <name-of-the-helm-chart> -n <name-of-the-namespace>`<br>For example, `helm upgrade slserver SecureLogin-Server-x.x.x.x -n my-ingress` |
| To uninstall helm | `helm uninstall <name-of-the-release> -n <namespace>`<br>Example: `helm uninstall slserver001 -n nsl-ingress` |
| To restart the Advanced Edition server | `kubectl rollout restart deployment <deployment name> -n <namespace>`<br>Example: `kubectl rollout restart deployment slserver -n nsl-ingress` |
| To check the Advanced Edition server debug log | `kubectl logs -f pod/<slserver-podname> -n <ingress-namespace>`<br>**NOTE:** By default, the log level is set to "ERROR". Other available levels are DEBUG, INFO, ERROR, WARN, TRACE, FATAL, and OFF.<br>To change the log level:<br>1. Open `SecureLogin-Server-x.x.x.x\values.yaml`.<br>2. Specify the required level in **rootLevel.**<br>3. Perform the helm upgrade. |

| Scenario | Use the Command |
| --- | --- |
| To get all services and their IP addresses | `kubectl get services -n <ingress-namespace>`<br>Example: `kubectl get services -n nsl-ingress` |