EXTERNAL COLLABORATION WITH AZURE AD B2B

SUCCINCTLY

BY SJOUKJE ZAAL

SUCCINCTLY EBOOK SERIES

Syncfusion

External Collaboration with Azure AD B2B Succinctly

By
Sjoukje Zaal

Foreword by Daniel Jebaraj

2501 Aerial Center Parkway
Suite 200
Morrisville, NC 27560
USA

ISBN: 978-1-64200-206-5

Technical Reviewer: James McCaffrey
Copy Editor: Courtney Wright
Acquisitions Coordinator: Tres Watkins, VP of content, Syncfusion, Inc.
Proofreader: Graham High, senior content producer, Syncfusion, Inc.

Table of Contents

The Story Behind the *Succinctly* Series of Books

Daniel Jebaraj, Vice President
Syncfusion, Inc.

Staying on the cutting edge

As many of you may know, Syncfusion is a provider of software components for the Microsoft platform. This puts us in the exciting but challenging position of always being on the cutting edge.

Whenever platforms or tools are shipping out of Microsoft, which seems to be about every other week these days, we have to educate ourselves, quickly.

Information is plentiful but harder to digest

In reality, this translates into a lot of book orders, blog searches, and Twitter scans.

While more information is becoming available on the internet and more and more books are being published, even on topics that are relatively new, one aspect that continues to inhibit us is the inability to find concise technology overview books.

We are usually faced with two options: read several 500+ page books or scour the web for relevant blog posts and other articles. Just as everyone else who has a job to do and customers to serve, we find this quite frustrating.

The *Succinctly* series

This frustration translated into a deep desire to produce a series of concise technical books that would be targeted at developers working on the Microsoft platform.

We firmly believe, given the background knowledge such developers have, that most topics can be translated into books that are between 50 and 100 pages.

This is exactly what we resolved to accomplish with the *Succinctly* series. Isn't everything wonderful born out of a deep desire to change things for the better?

The best authors, the best content

Each author was carefully chosen from a pool of talented experts who shared our vision. The book you now hold in your hands, and the others available in this series, are a result of the authors' tireless work. You will find original content that is guaranteed to get you up and running in about the time it takes to drink a few cups of coffee.

Free forever

Syncfusion will be working to produce books on several topics. The books will always be free. Any updates we publish will also be free.

Free? What is the catch?

There is no catch here. Syncfusion has a vested interest in this effort.

As a component vendor, our unique claim has always been that we offer deeper and broader frameworks than anyone else on the market. Developer education greatly helps us market and sell against competing vendors who promise to “enable AJAX support with one click,” or “turn the moon to cheese!”

Let us know what you think

If you have any topics of interest, thoughts, or feedback, please feel free to send them to us at succinctly-series@syncfusion.com.

We sincerely hope you enjoy reading this book and that it helps you better understand the topic of study. Thank you for reading.

Please follow us on Twitter and “Like” us on Facebook to help us spread the word about the *Succinctly* series!

About the Author

Sjoukje Zaal is a CTO Microsoft at Capgemini Netherlands, Microsoft Regional Director, and Microsoft Azure MVP with over 20 years of experience providing architecture, development, consultancy, and design expertise. She works at Capgemini, a global leader in consulting, technology services, and digital transformation. She mainly focuses on cloud, security, productivity, and IoT.

She loves to share her knowledge and is active in the Microsoft community as a co-founder of the user groups Tech Daily Chronicle, Global XR Community, and the Mixed Reality User Group. She is also a board member of Azure Thursdays and Global Azure. Sjoukje is an international speaker involved in organizing many events. She wrote several books and writes blogs. Sjoukje is also part of the MVP Diversity and Inclusion Advisory Board.

Chapter 1 What is Azure AD B2B?

In this book, we are going to cover Azure Active Directory business-to-business (B2B) and how you can use it to collaborate with external users in Azure and Office 365.

In this first chapter, we are going to introduce Azure AD B2B and cover how it can be used to manage external accounts in Azure and Office 365. We are also going to cover how Azure AD B2B relates to Azure AD and Azure AD business-to-consumer (B2C), which are other products that can be used for managing identity and access in Azure. Lastly, we are going to cover the license structure and what features are important to consider when using Azure AD B2B in your organization.

Introducing Azure AD B2B

With Azure Active Directory business-to-business, organizations can work safely and securely with other organizations in Azure. You can securely share your company's applications and services with partners and guest users using Azure AD B2B. These users can come from different organizations, but they can also be individuals that need access to your applications and services, through the Azure AD tenant.

With Azure AD B2B, every partner uses its own identity management solution, through Azure AD. This means there is no external administrative overhead for your organization. Partner and guest users can get access using a variety of accounts, like an Azure AD or Office 365 account (work and school) with an Azure AD tenant bound to it. They can also use personal accounts, such as personal Microsoft accounts, or non-Microsoft accounts. These existing accounts can be added to Azure AD, and guest users can then use their own credentials when they log in to your applications, services, and resources.

In the next section, we will cover the different identity products in Azure and what Azure AD business-to-business has to offer.

Other identity management products in Azure

Azure offers different products that are integrated into Azure Active Directory. There is the "normal" Azure AD, Azure AD business-to-business (B2B), and Azure AD business-to-consumer (B2C). These different services can sometimes lead to confusion. In this section we are going to cover the differences and the key values of each service.

Azure Active Directory

Azure Active Directory is the core of all identity management services and features in Azure. It offers traditional username and password identity management. Users can be created in Azure AD manually using the Azure portal, but this can also be done programmatically and automatically using the Azure SDK, PowerShell, and the Graph API. It also offers roles and permission management, which is called role-based access control (RBAC). This is used to give access and permissions to various resources in Azure, such as management and resource groups, databases, applications, and so on. On top of that, Azure AD offers more enterprise-grade solutions, such as multifactor authentication (MFA) and application monitoring, solution monitoring, and alerting. Azure AD can easily be integrated with your on-premises Active Directory to create a hybrid infrastructure.

Tip: For more information about Azure AD and all the features it has to offer, you can refer to the Microsoft documentation.

Azure Active Directory Business-to-Consumer (B2C)

Azure AD B2C is a cloud identity-management solution for custom applications, such as mobile and web applications. Azure AD B2C is a global service that scales to all available Azure regions, automatically handles threats, and offers monitoring capabilities.

It offers out-of-the-box authentication providers that can be leveraged from within your apps and custom APIs. These identity providers are configured in your Azure AD B2C tenant in Azure, and can then easily be used in your custom applications. This enables developers to easily implement scalable identity management in a secure way for their applications.

Azure AD B2C offers the following authentication providers:

- **Social accounts**: Facebook, Google, LinkedIn, and more.
- **Enterprise accounts**: Accounts that use open standards protocols, such as OpenID Connect or SAML.
- **Local accounts**: Accounts that use email address/username and password.

Your application needs to be registered inside the Azure B2C tenant. After registration, built-in policies can be configured for the app where you can enable different authentication methods, set claims, enable MFA, or create a password reset policy that the app can use. You then add the required configuration settings for the application that is registered in the Azure B2C tenant to your code, and all the previously mentioned settings can be used without any further configuration.

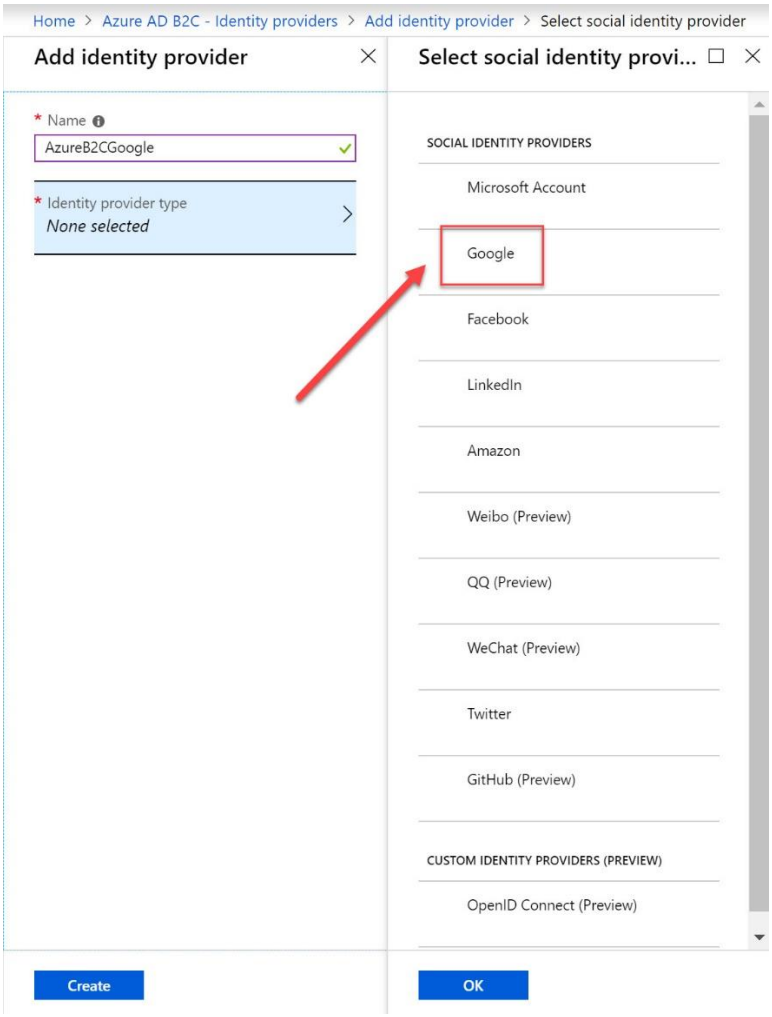

Figure 1: Overview of identity providers in Azure AD B2C

Tip: You can refer to this site for more information about Azure AD B2C.

In the next section, we are going to introduce the different features that Azure AD B2B has to offer for your organization.

User account management

Guest users can use their own identity management solutions to log in and authenticate to your Azure and Office 365 tenant. This means that an Azure AD tenant for the partner organization or guest users is not required. Users can also use their personal accounts to log in to your Azure AD tenant. This makes it easier for administrators because they don't have to manage these accounts or passwords, accounts don't need to be synced, and account lifecycles don't need to be managed.

Azure AD B2B has support for the following types of accounts:

- Work or school accounts (such as Azure Active Directory and Office 365).
- Personal Microsoft accounts (such as Outlook and Hotmail).
- Other types of accounts, called unmanaged accounts (such as Gmail and iCloud).

Guest users with Microsoft accounts, like work, school, or personal accounts, can log in to the Azure AD tenant using an email address and password. These credentials are already known by Microsoft.

Guest users who are using unmanaged accounts have to undergo a slightly different process to log in to the Azure AD tenant. After the user is added to the Azure AD tenant, the invite is sent to the guest user. The first time the user logs in to the Azure AD tenant, they need to specify a password, which is stored in Azure AD. So, in the case of unmanaged accounts, the user can only log in using their own email address, but they still need to specify a password.

> ***Note: Starting March 31, 2021, Microsoft will no longer support the creation of unmanaged Azure AD accounts and tenants for B2B collaboration scenarios. This means that external users who are using unmanaged accounts, such as non-Microsoft accounts, need to use email one-time passcode authentication (now in preview). It is highly encouraged to opt into email one-time passcode authentication. This will be covered in more depth in Chapter 2, User Account Management.***

Once the guest users are added to the Azure AD tenant, they can be added to security groups and applications. This will also be covered in the next chapter.

In the following image, you see an overview of internal and external accounts in Azure AD. You can see the different types of accounts under the **User type** column.

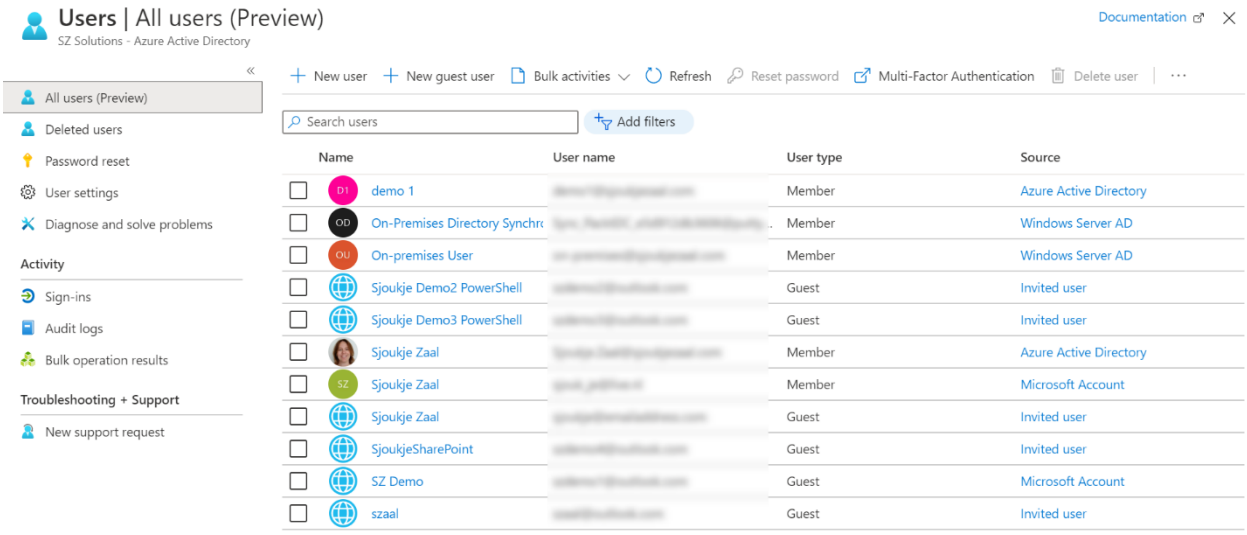

Figure 2: Overview of internal and external users in Azure AD

In the next section, we are going to introduce the onboarding experience.

Onboarding experience for B2B guest users

The onboarding experience for B2B guest users starts with adding the guest account to Azure AD. The invitation is sent to the inbox of the email address that is used to register the account. The guest user needs to accept the invitation to get access to your organization's Azure AD tenant.

In the following image, you see an example of an invitation that is sent to the inbox of the guest user account.

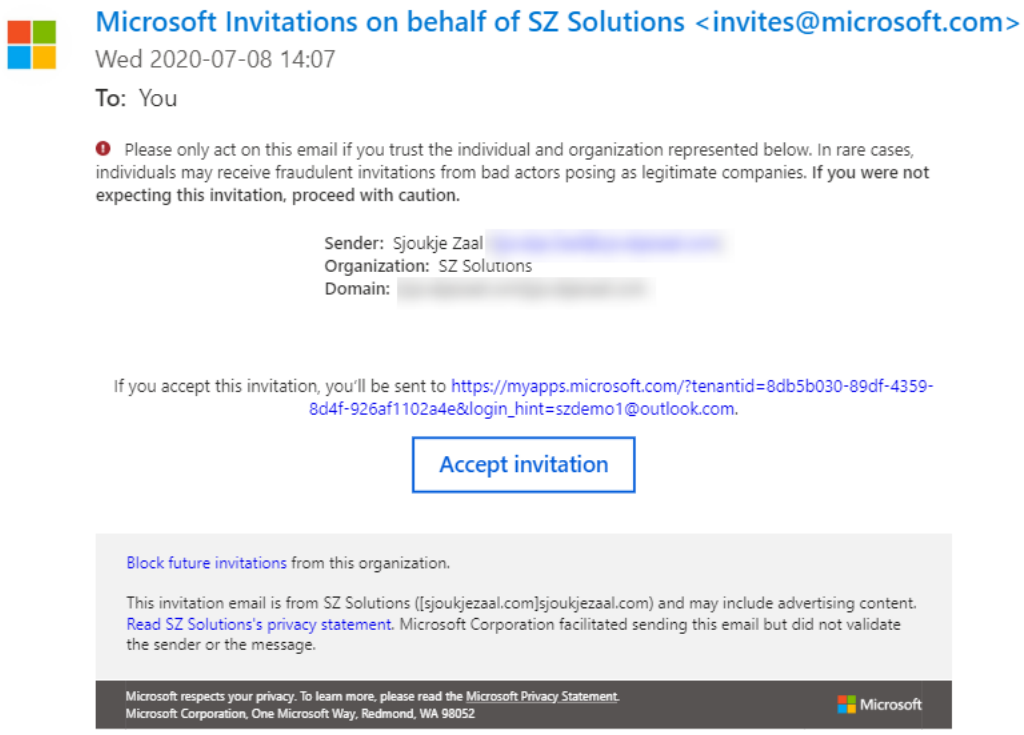

Figure 3: Example of an invitation for a guest user

It's possible to add the guest user to the Azure AD tenant without sending an invitation. This can be done by sending a direct link to an app, or by sending an invitation to the user's own access panel. We are going to cover this in more detail in Chapter 3.

Conditional access policies for secure sharing

Depending on the Azure AD licenses that you already have for your internal users, it is possible to enable conditional access policies for your external users as well. Conditional access policies offer organizations the ability to protect corporate assets in different circumstances. You can apply the right access controls only when needed, and do not have to bother users with unneeded security measures. Examples of conditional access policies are multifactor authentication, blocking sign-ins from users from specific regions or devices, and using legacy authentication protocols.

Azure AD B2B offers exactly the same conditional access policies for external users as for internal users. This means that if your organization has Azure AD premium licenses that include conditional access for your internal users, the same policies can be implemented for external users as well.

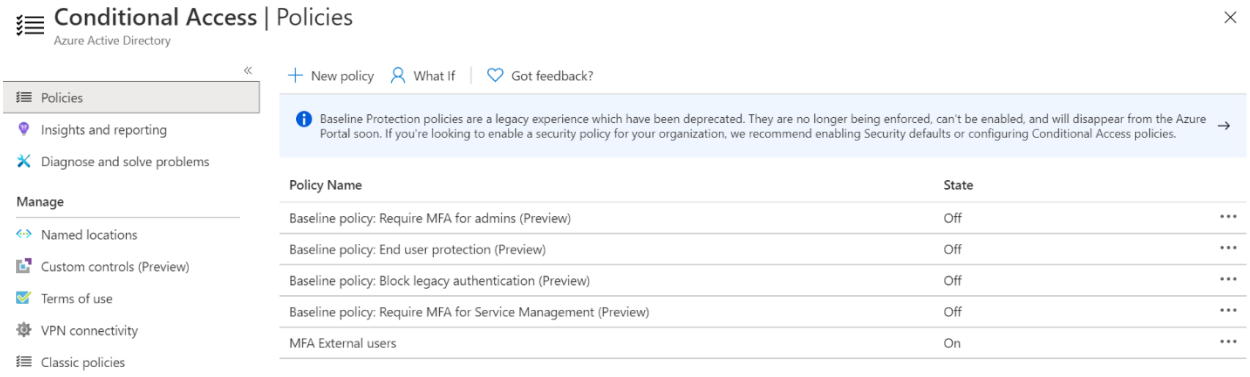

Figure 4: Overview of conditional access policies in Azure AD

Tip: Conditional access policies are not further covered in this book, but for more information you can refer to this website.

In the next section, we are going to look at the integration with identity providers.

Integration with identity providers

You can set up federation with external identity providers in Azure AD. Using these providers, guest users can sign into your organization's applications and services using existing social accounts or enterprise accounts. There is support for Google and Facebook, and you can also set up direct federation with any organization whose identity provider supports the SAML 2.0 or WS-Fed protocols. This makes it possible to collaborate easily with partners or connected organizations. By adding a federation to that company's identity provider, partners can log in using their own corporate identity.

In the next image, you see an overview of the identity providers that can be added from the Azure portal.

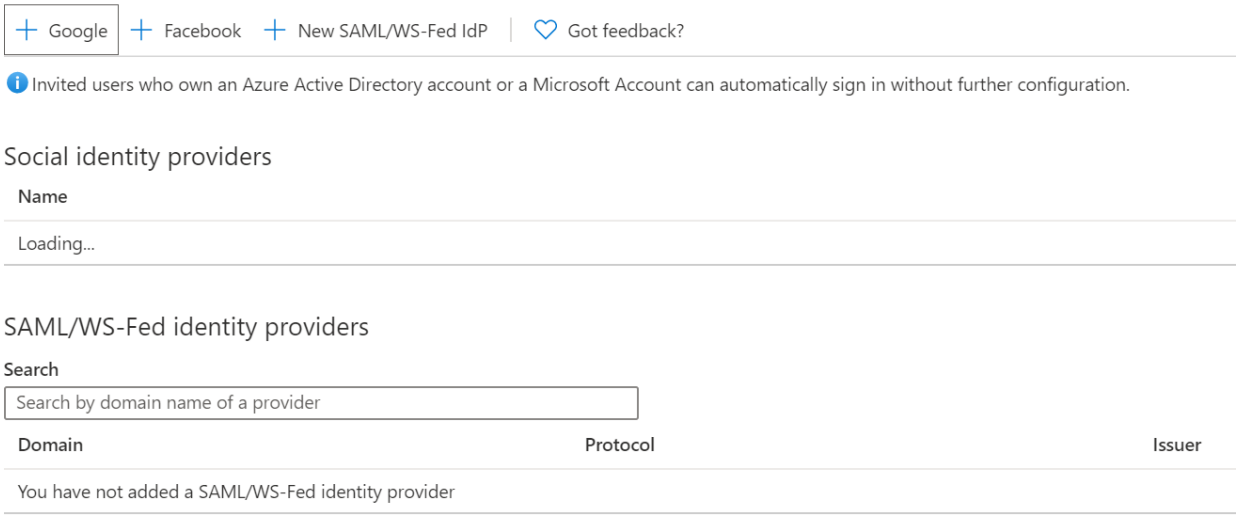

Figure 5: Adding identity providers in the Azure portal

Now that we have a high-level understanding of the different features that Azure AD B2B has to offer for your guest users, we can now look into the relationship of Azure AD B2B to other Azure AD products.

Different Azure AD licensing offerings

Azure AD B2B guest users use the same Azure AD features that internal Azure AD users do. Azure AD offers four different pricing plans; each pricing plans adds extra features on top of the previous plan:

- **Free**: This offers the most basic features, such as support for up to 500,000 objects, single sign-on (SSO), Azure B2B for external users, support for Azure AD Connect synchronization, self-service password change, groups, and standard security reports.
- **Office 365 Apps**: This offers unlimited objects, a service-level agreement (SLA) of 99.9%, self-service password reset, company branding features, and support for the application proxy.
- **Premium P1**: This offers advanced reporting, MFA, conditional access, mobile device management (MDM) auto-enrollment, cloud app discovery, and Azure AD Connect Health.
- **Premium P2**: This offers identity protection and privileged identity management (PIM).

Tip: For a detailed overview of the different pricing plans and all the features that are offered for each, you can refer to this pricing page.

Guest users can use exactly the same features as internal users. This means that if you have Premium P2 licenses for your users, you can also use the privileged identity management (PIM) features and identity protection for your external users. However, there is one caveat: there is a licensing ratio for Azure AD B2B users. For each internal license that you have, you can add up to five guest users. So, for example, if you have 10 internal Azure AD licenses, you can add a maximum of 50 guest users to your tenant. If you want to add more guest users to your tenant, you need to buy additional Azure AD licenses. Of course, this is not the case if you are using the Free Azure AD plan.

This concludes the first chapter. We have introduced Azure AD B2B and covered the different tools and features that come with Azure AD B2B. We have also covered the different licensing offerings of Azure AD and how they relate to Azure AD B2B. In the next chapter, we are going to add guest users to our Azure AD tenant.

Chapter 2 User Account Management

In the previous chapter, we covered Azure Active Directory business-to-business from a high level. In this and the upcoming chapters, we are going to cover some features in more depth. In this chapter we will fully focus on user account management. We will create a guest user in our Azure AD tenant, create a group, and assign a guest user to groups and applications.

How to create user accounts

As covered in the previous chapter, guest users can be added to Azure AD using their own credentials. In this demonstration, we are going to add a guest user by using a personal Microsoft account. The following steps explain the process:

1. Navigate to the Azure portal by opening https://portal.azure.com/.
2. In the top menu, type **Azure Active Directory** in the search box. Then select **Azure Active Directory** as shown:

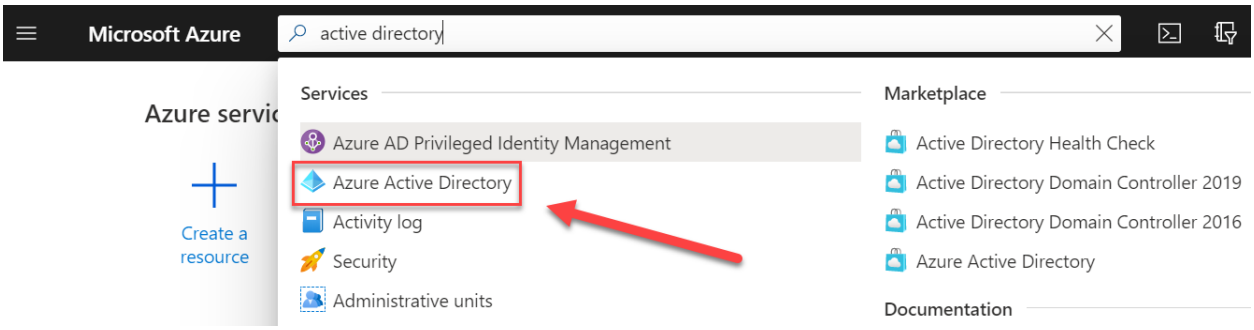

Figure 6: Open Azure AD in the Azure portal

3. In the **Overview** blade of Azure AD, in the left menu, select **Users** > **All users**.
4. Select **+ New guest user** from the top menu as follows:

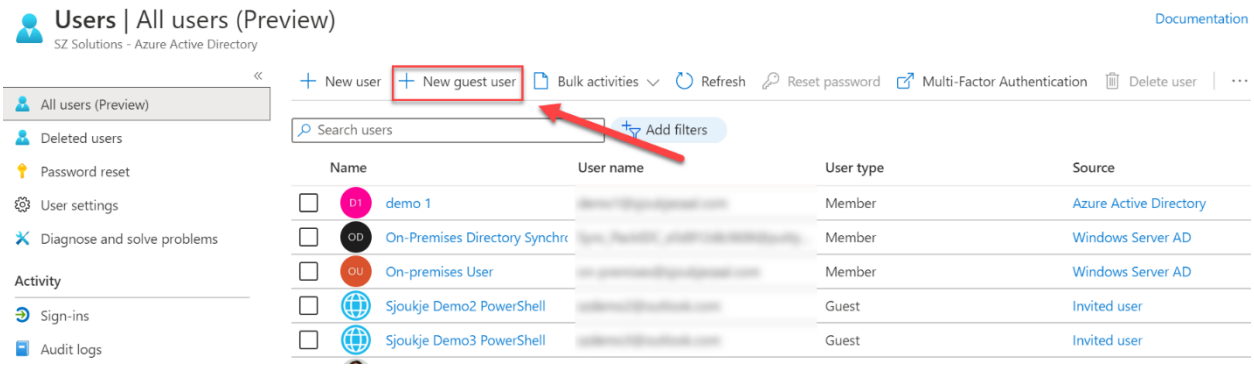

Figure 7: Add guest user

5. Provide an email address and a personal message, which is sent to the user's inbox. This personal message includes a link to log in to your Azure AD tenant. You can also add the guest user to a group and a role-based access control (RBAC) role directly from here, although we are not going to do this in this step.

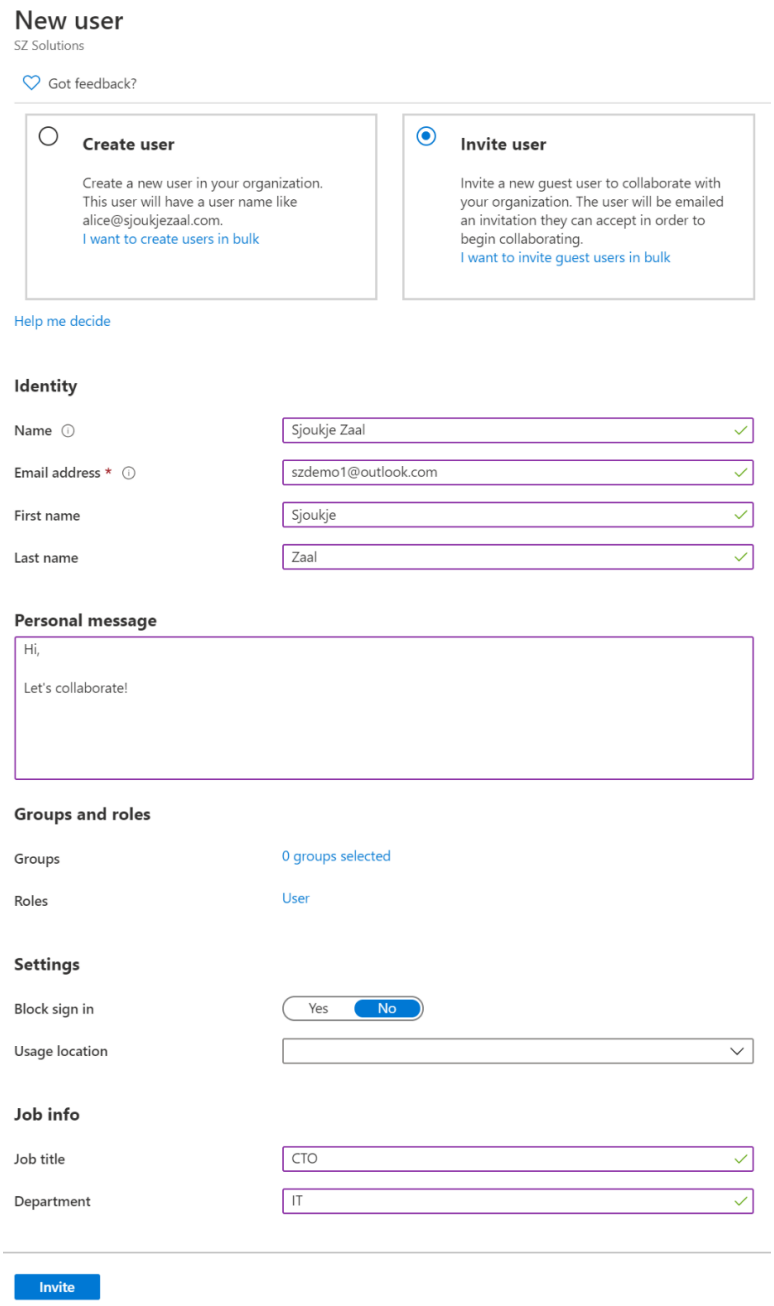

Figure 8: Add guest information

6. Click **Invite**. This will create the user in the Azure AD tenant, and the invitation email will be sent to the email address you provided.

Now that we have created the user, we can add this user to a security group in Azure AD.

Groups and member types in Azure AD

There are several group types and member types in Azure AD. In the following section, we are going to describe the different types and why they are used. This can help you determine to which group your guest user can be added.

Group types

Azure AD offers the following two group types:

- **Security:** Security groups are used to manage user, device, and application access to shared resources in Azure for a group of users. You can create a user group for a specific user policy, for instance, or create a security group for specific types of applications. Using a security group, you can set permissions to a group of members at once, instead of adding these permissions to separate users. This makes management much easier and clearer. A security group can contain users, devices, groups, and service principals.
- **Office 365:** Office 365 groups can give access to a shared mailbox, SharePoint sites, Teams channels, calendars, files, and more. An Office 365 group can only have users; devices and groups are not supported. Both users and service principals can be owners of the group.

Group members

Azure AD has the following member types:

- **Assigned:** Specific users can be added as members of this group and they have unique permissions.
- **Dynamic user:** Dynamic users can have dynamic membership rules to automatically add and remove users. When an attribute of the member changes, the dynamic rules will be evaluated to see if the member meets the rule requirements. The user will then be added to the group. If the user doesn't meet the requirement and is already part of the group, they will be removed.
- **Dynamic device:** This member type lets you define rules for devices. Based on the attributes of the device, it is added to the group or removed from the group.

In the next demonstration, we are going to create a security group and add the user that we created in the previous step to it.

How to add the guest user to a security group

The next step is to add the user to a security group in Azure AD. First, we are going to create a security group, and then add the user to it.

1. Navigate to Azure Active Directory again in the Azure portal.
2. In the left menu, select **Groups**.
3. Click **+ New group** in the top menu:

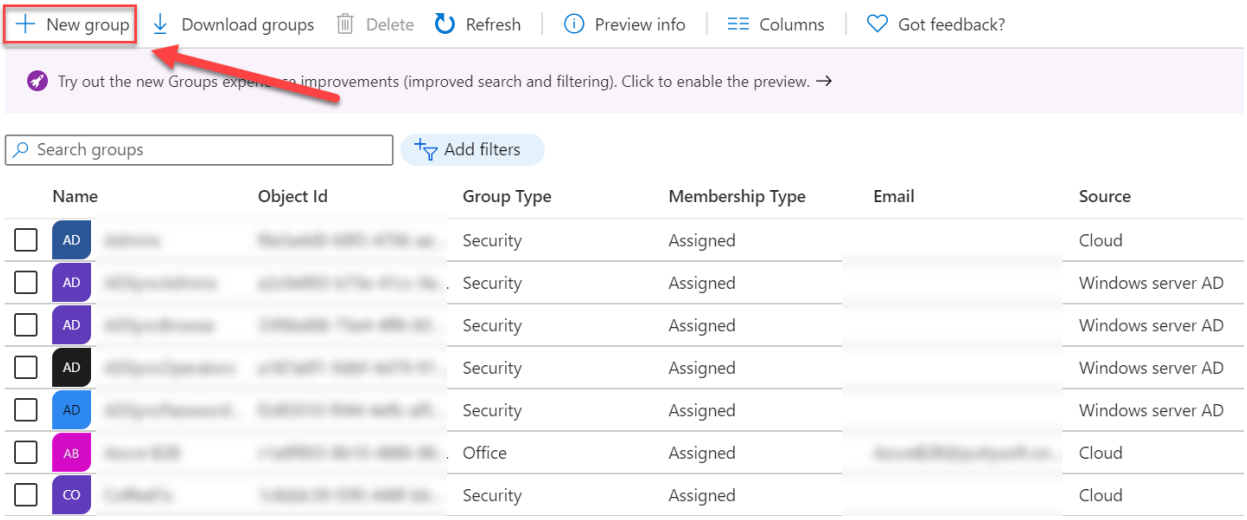

Figure 9: Add new group

4. Make sure that the group type is set to **Security**. We are not creating an Office 365 group for this demo. Give the group a name and a description:

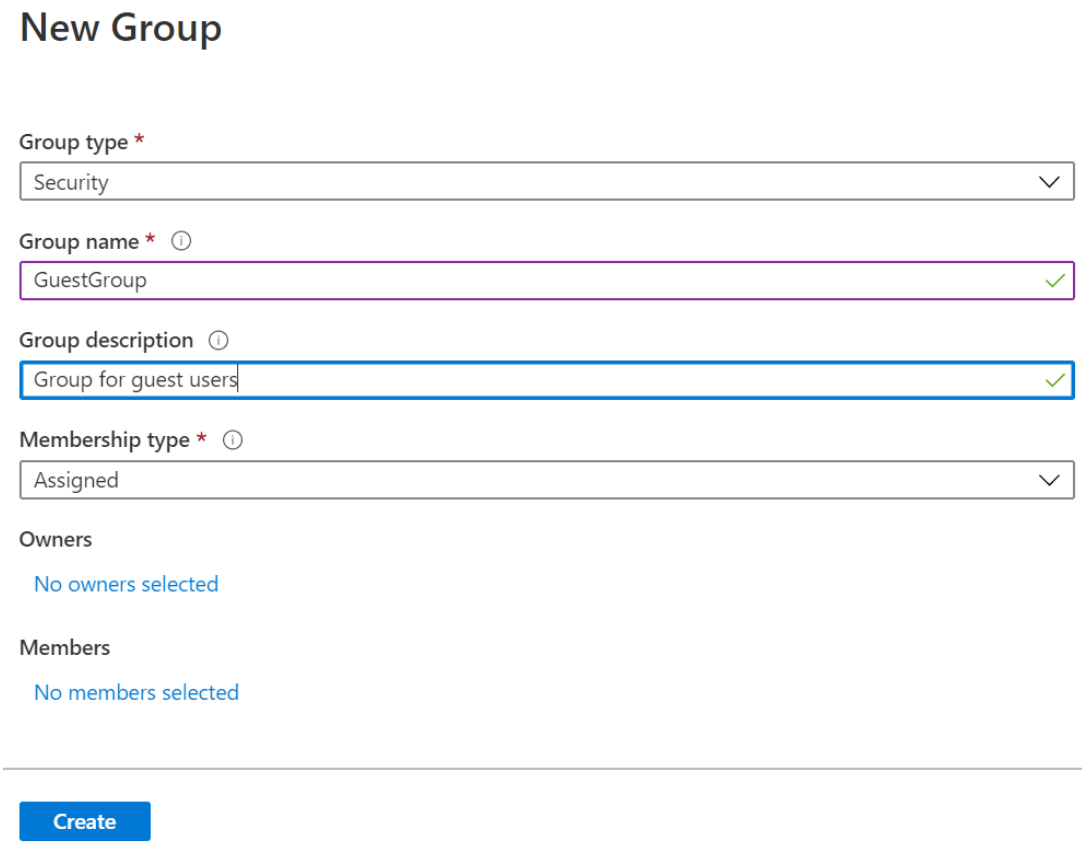

Figure 10: Specify values for the group

5. Click **Create**.

6. After creating the group, we can add the user to it. Click the created group in the list of groups to go to its settings:

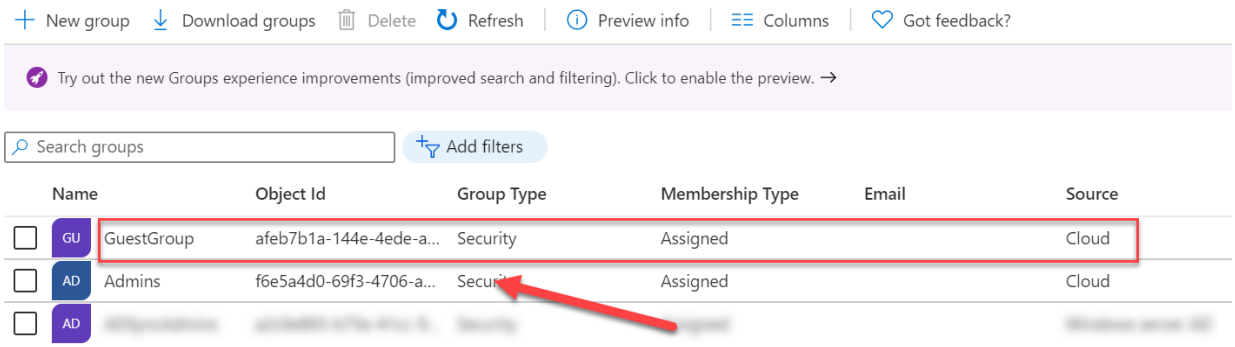

Figure 11: Select the group

7. Select **Members** in the left menu, and **Add members** in the top menu:

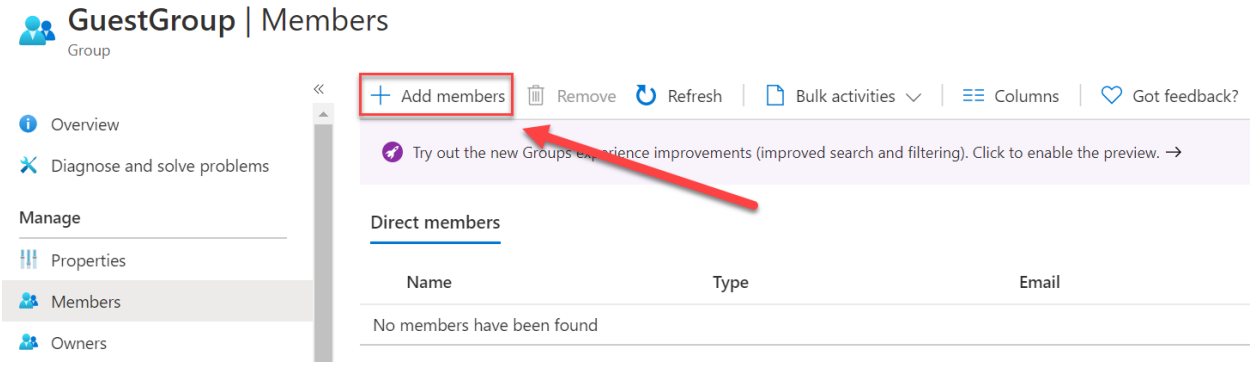

Figure 12: Add members to the group

8. In the search field, type the user account name that we created in the previous demo, and select the account to add it:

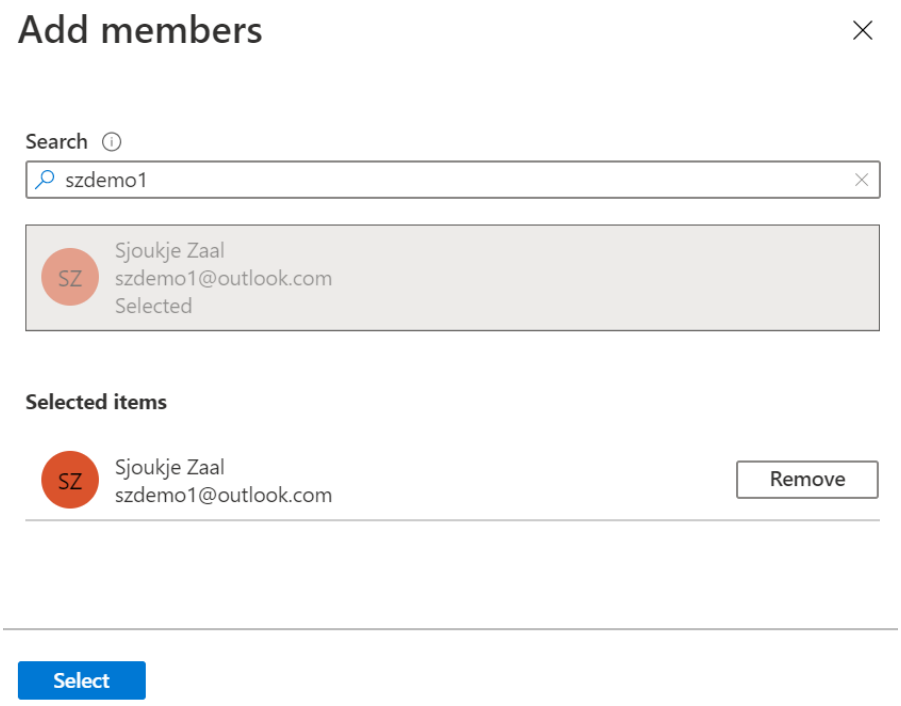

Figure 13: Select the group

9. Click **Select**.

The guest user is now added to the security group. In the next step, we are going to add the group to an application. This way, the guest user will have permissions to access the application when logging in to your Azure AD tenant.

How to assign accounts and groups to applications

The last step in this chapter is to add the security group that we created in the previous demo to an application. When the user accepts the invitation, they will automatically be redirected to the overview page of the applications they have access to.

Note: In the demonstration, an application is used that is already registered in Azure AD. If you want to add groups and members to applications, you first need to register them in Azure AD. For more information about this, you can refer to this site.

1. Navigate to Azure Active Directory again in the Azure portal.
2. In the left menu, select **Enterprise applications**.
3. Select an application from the list:

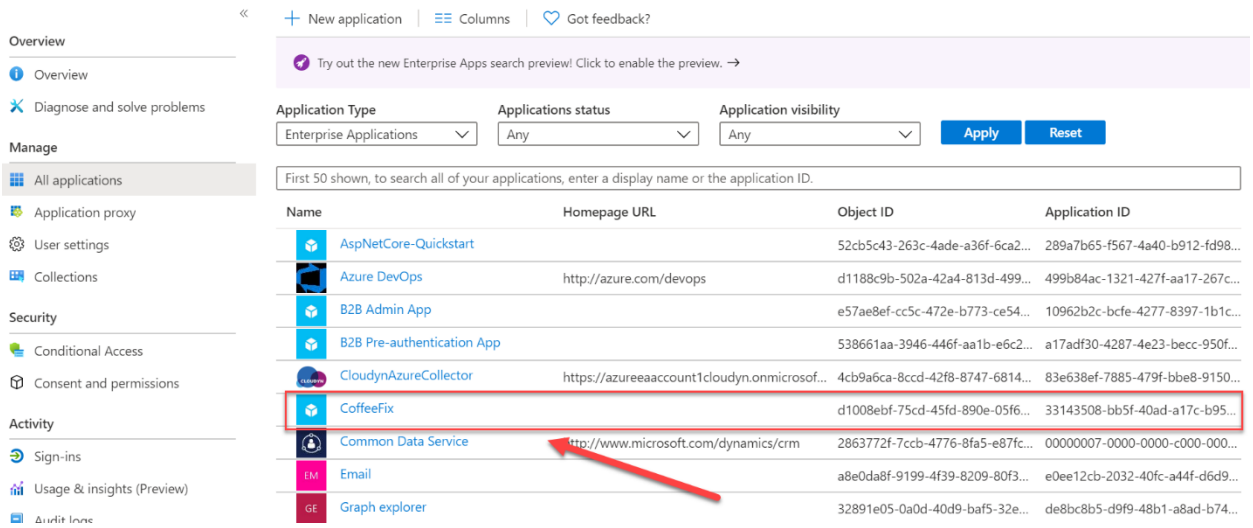

Figure 14: Select an application

4. In the left menu of the application overview page, select **Users and groups**. Then in the top menu, select **+ Add user** (you also add groups here):

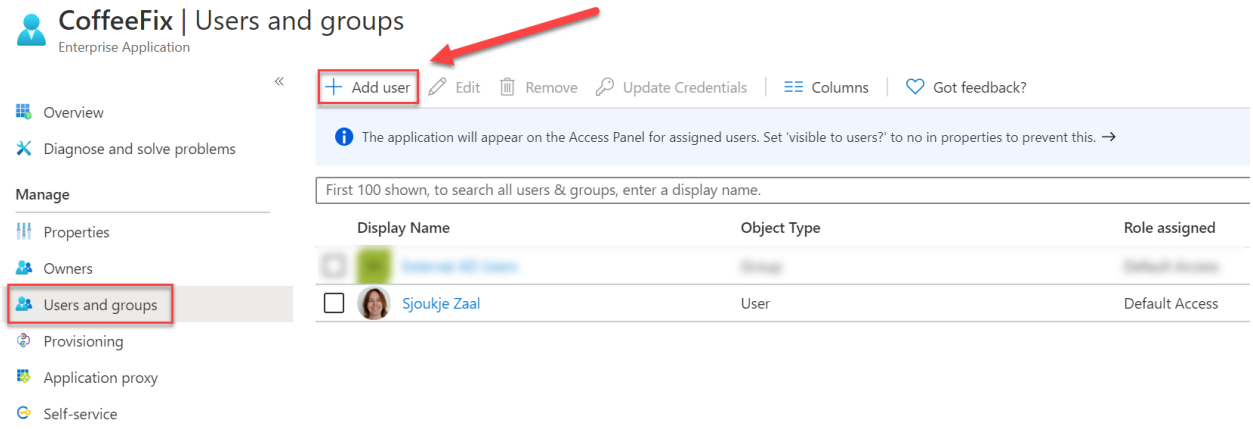

Figure 15: Add a group

5. In the **Add Assignment** blade that opens, click **User and groups**. In the search box, type the name of the group that we created, which is **GuestGroup**. Select the group and click **Select**.

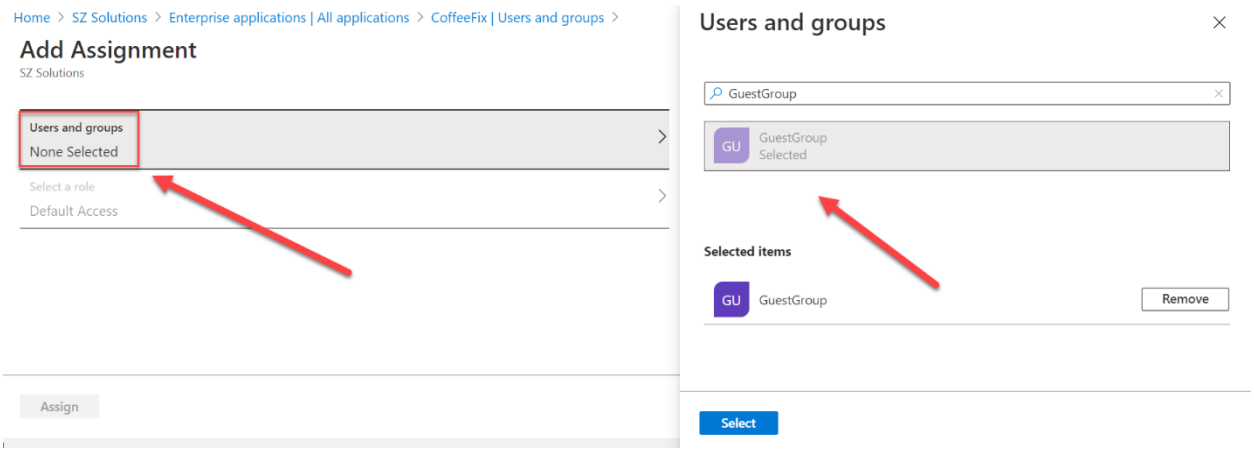

Figure 16: Add the group

6. After the group is selected, click **Assign**.

The group and the users that are added to the group now have access to this application.

In this chapter, we have covered user account management in Azure AD and Azure AD B2B. We have added a guest user, created a security group, and added the group to an application. In the next chapter, we are going to cover the invitation redemption process for the guest user.

Chapter 3 Invitation Redemption

In the previous chapter, we covered how users can be added to Azure Active Directory B2B, and how security groups and applications work in Azure AD. In this chapter, we are going to cover the invitation redemption process from the moment the guest user receives the email and logs in to the Azure AD tenant for the first time.

Flow of the guest user

Adding the guest user from the Azure portal to Azure AD B2B will kick off the following flow:

1. The administrator adds a guest user to Azure AD from the Azure portal by filling in the first name, last name, email address of the user, and a personal welcome message.
2. The guest user receives an invitation in their inbox.
3. The guest user will then click the link in the invitation email.
4. The browser is opened, and the guest user is redirected to the login page of Microsoft Azure. The user logs in using their own credentials.
5. The guest user needs to accept the privacy statement.
6. The guest user is redirected to the application landing page.

In the next sections of this chapter, we are going to cover these steps in more detail. We start with the invitation that is received by the guest user.

Receiving the invitation

Immediately after the guest user is added to the Azure AD tenant, the invitation email is sent to the email address that is specified for the guest account. If the email is not in the user's inbox, they should check the spam folder.

This invitation consists of the following parts:

- The company branding information: this includes the name of the organization.
- The name of the administrator who invited the user.
- The personal message that was provided when the user was added to the tenant.
- The Get Started button, which is the invitation redemption URL.

The email will look like the following image.

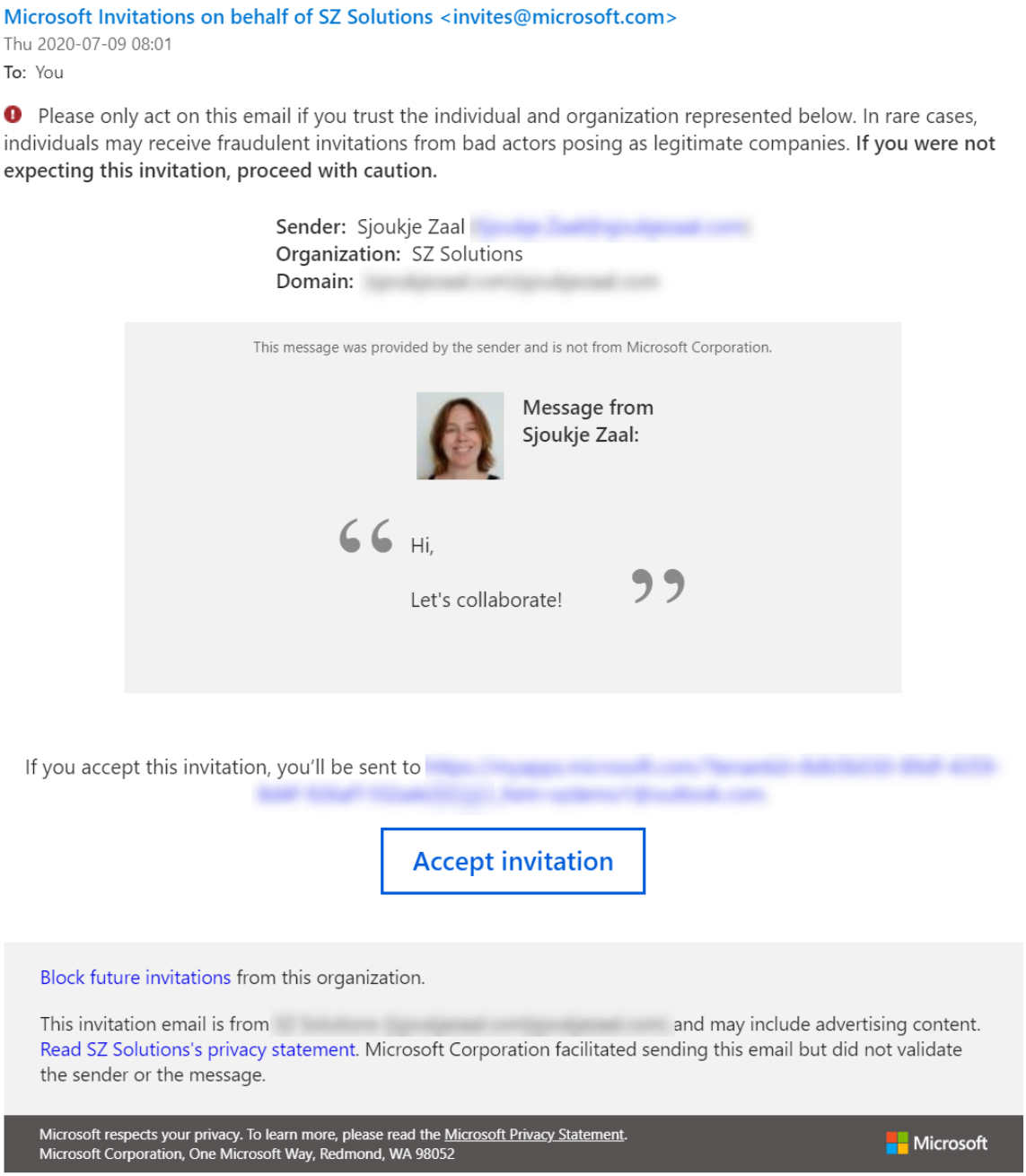

Figure 17: Invitation email

Resend the invitations to guest users

In cases where the guest user has not received the invitation email or has not redeemed the invitation, administrators can resend the email from the Azure portal with the following steps:

1. Navigate to the Azure portal by opening https://portal.azure.com/.

2. In the left menu, select **Azure Active Directory**:

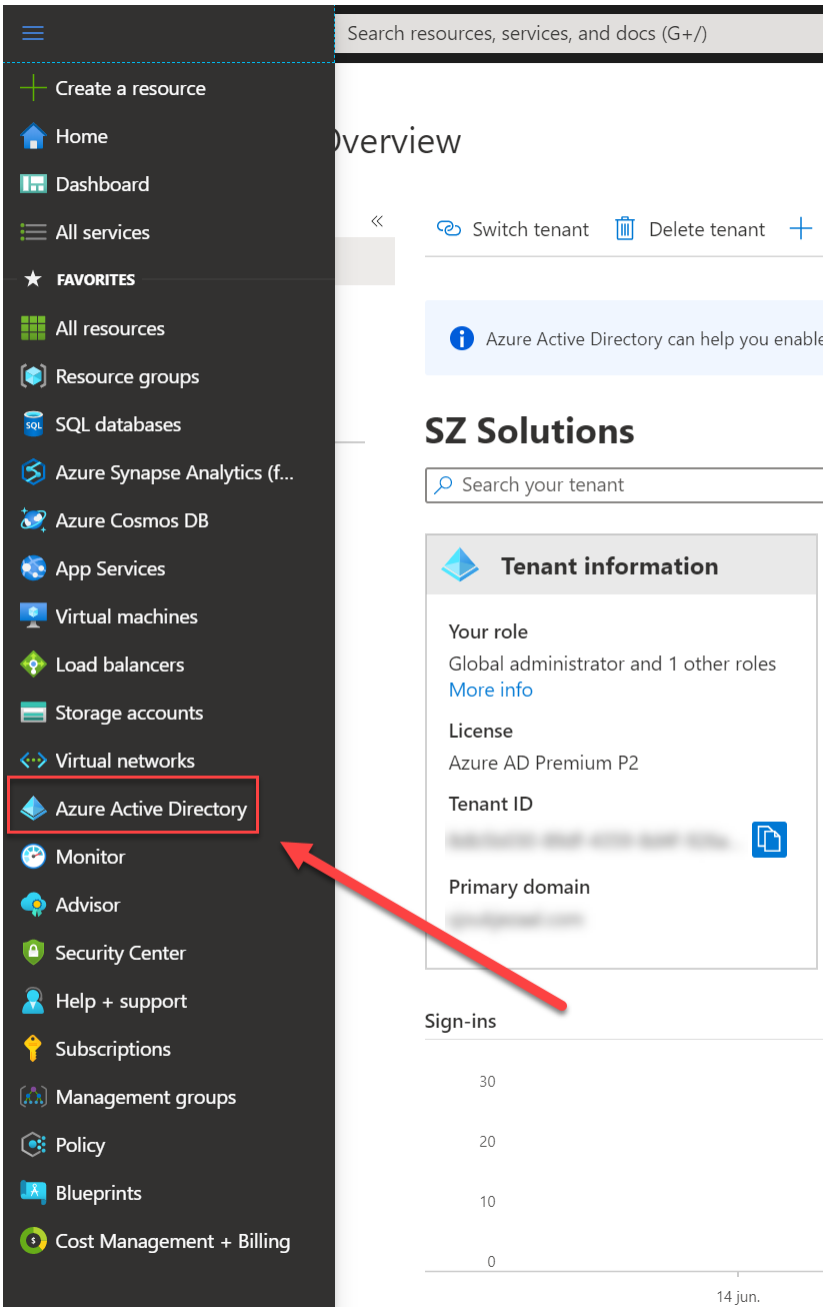

Figure 18: Left menu in Azure portal

3. In the Azure AD overview blade, in the left menu, select **Users**.
4. Select the user account that we created in Chapter 2 from the list:

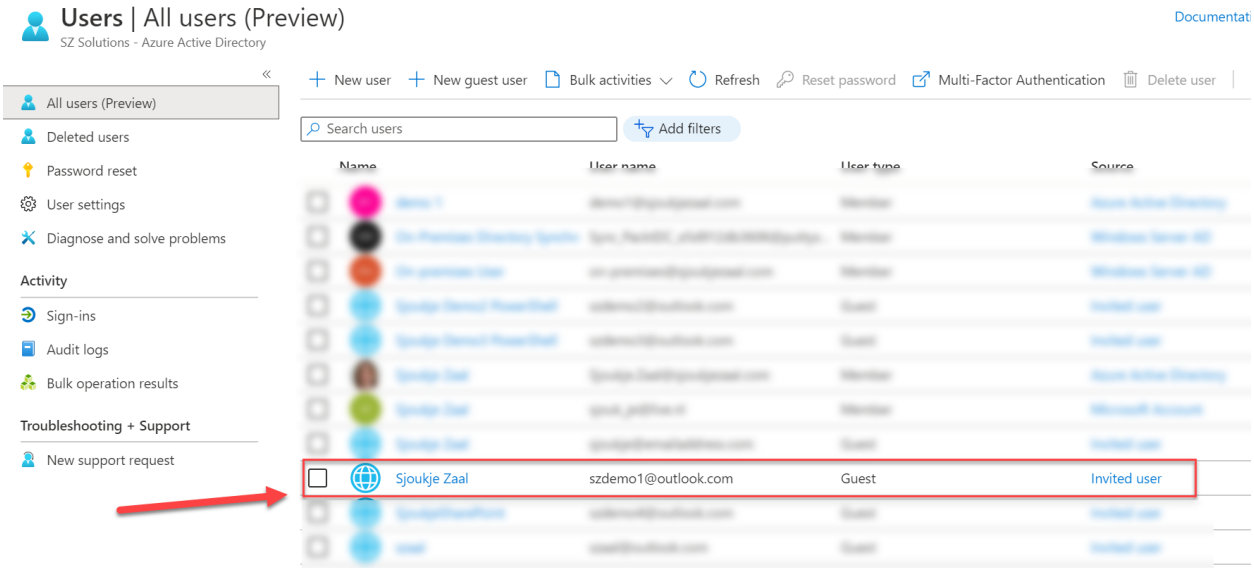

Figure 19: Select the guest user account

5. This will take you to the profile page of the user. From there you can make changes to the user attributes. You can also reset the password for the guest user, or resend the invitation by clicking **Resend invitation**:

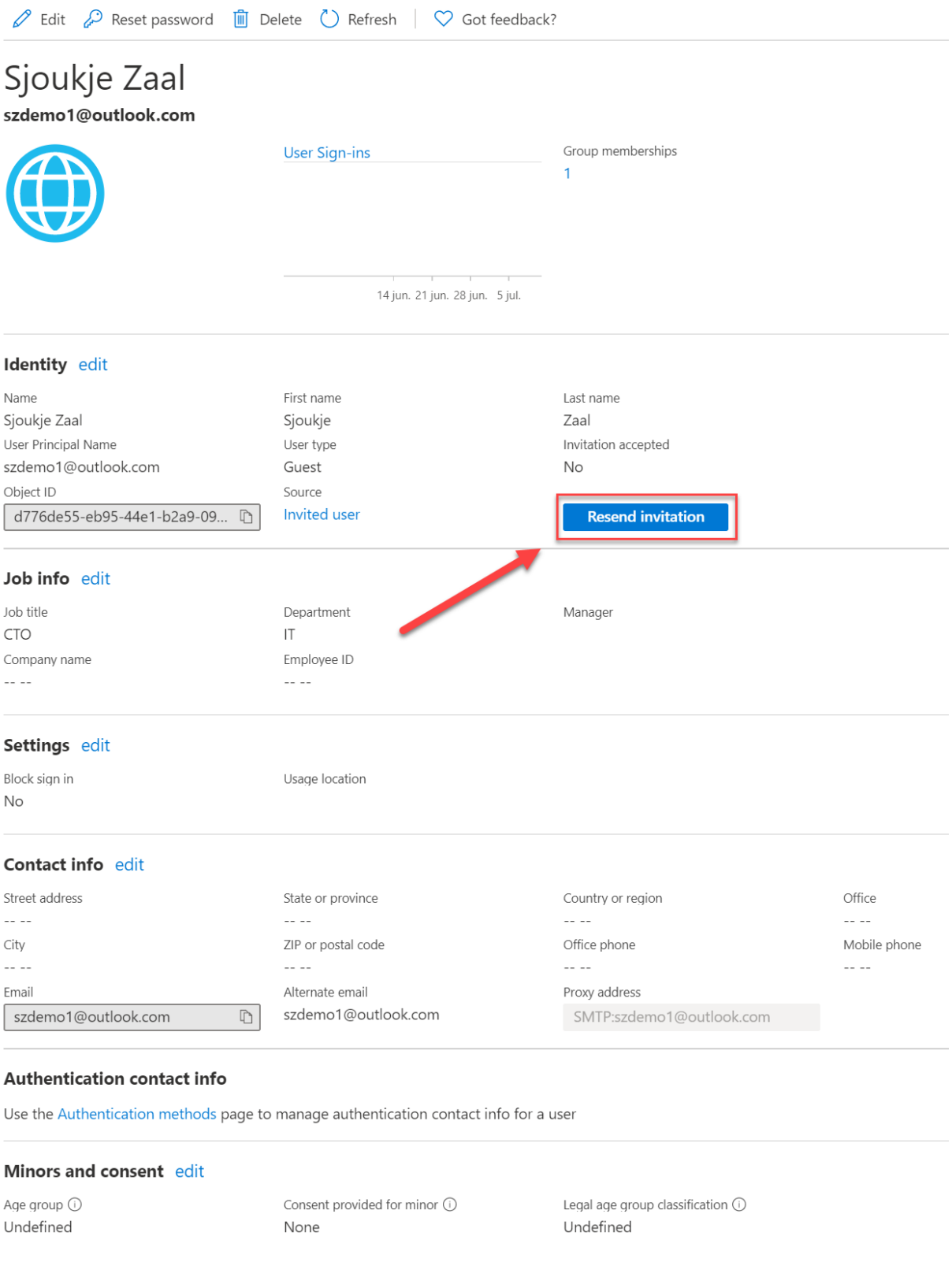

Figure 20: Profile settings for the guest user

In the next section, we are going to cover how the guest user can redeem the invitation.

Accepting the invitation

After receiving the invite, the guest user needs to accept the invitation. This is a mandatory process that needs to be followed before the user gets access to the Azure AD tenant. Therefore, the user needs to take the following steps:

1. Open or log in to the mailbox of the guest user's account. I've used an Outlook account for the demos in this book, so I opened the mailbox in the browser. There, search for the email called **Microsoft Invitations on behalf of…**.
2. Open the email and click **Accept invitation**. This will open a new browser window. You will be redirected to the login page of Azure, and the email address will automatically be filled in for you:

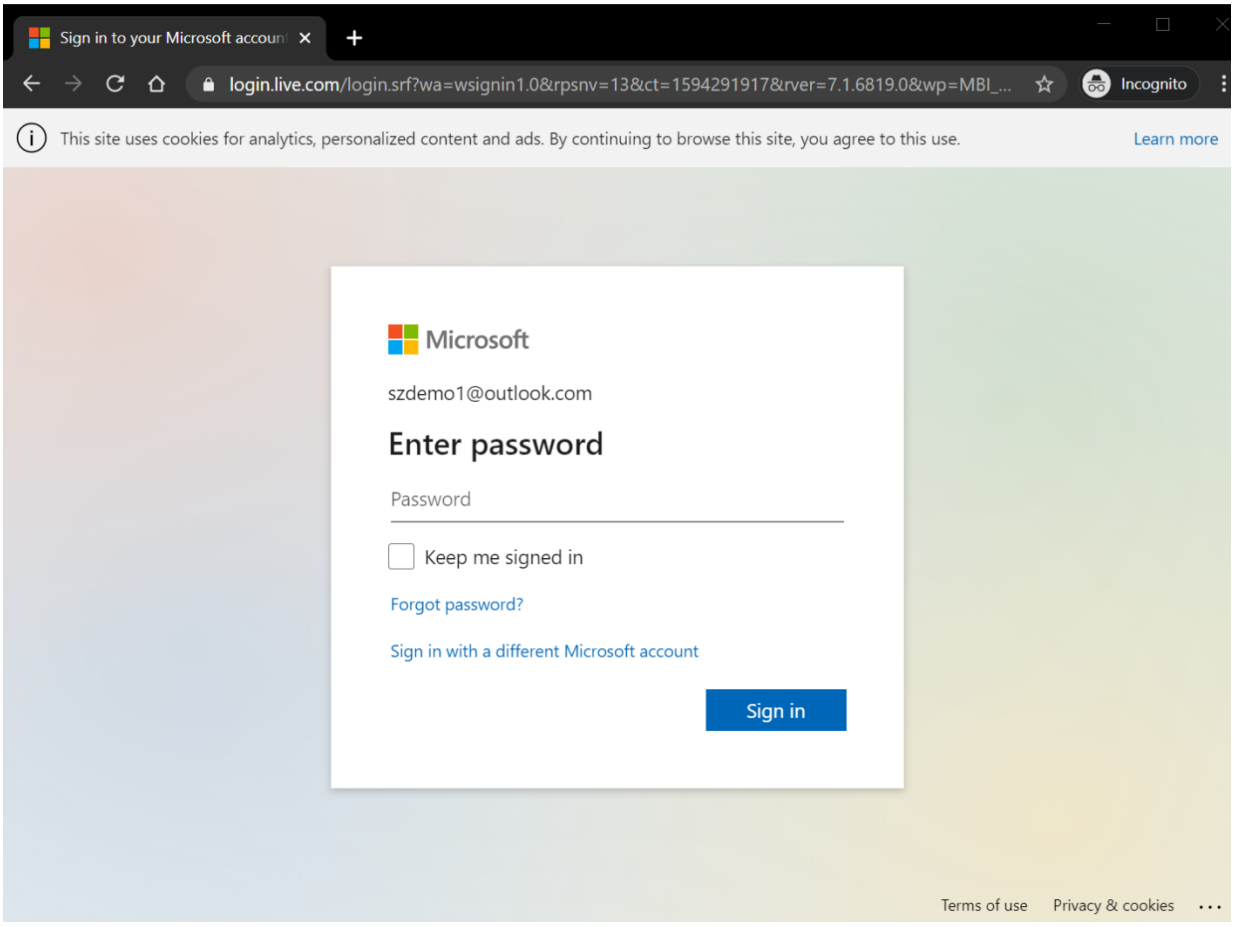

Figure 21: Microsoft login page

3. Enter your credentials in the **Password** field.
4. You will now be prompted with a privacy statement. It is mandatory to accept the privacy statement to get access to the Azure AD tenant. Click **Accept**.

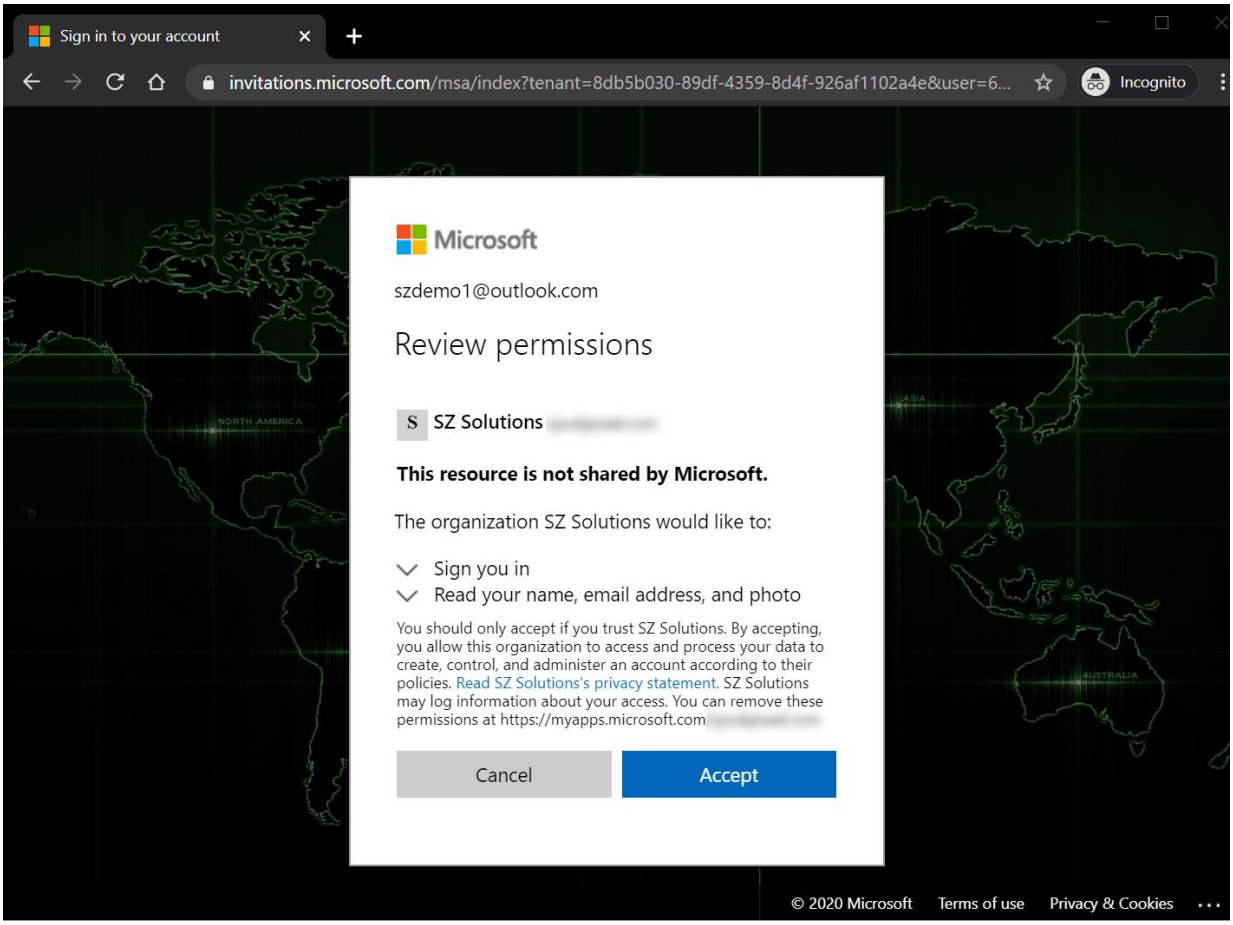

Figure 22: Privacy statement

5. You will now be redirected to the application landing page in Azure. There you will see an overview of the applications that you have access to, which in this case is the application we added the security group to in the previous chapter:

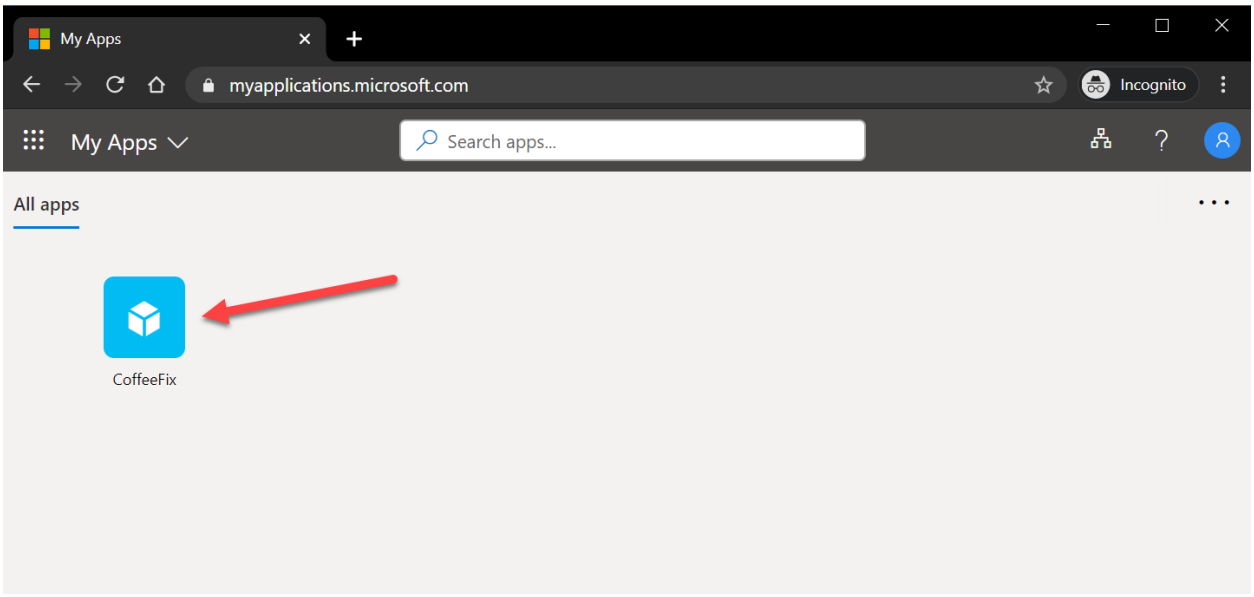

Figure 23: Application landing page

This concludes this demo and this chapter. We have accepted the invitation that was sent to the guest user, and the user now has access to the Azure AD tenant and the application.

In this chapter, we covered the complete process of adding a user to the Azure AD tenant. We accepted the invitation in the guest user's inbox and logged in to the My Applications page in Azure. In the next chapter, we are going to cover how you can use PowerShell and the Graph API to add users to Azure AD B2B.

Chapter 4 The Graph API and PowerShell

In the previous chapter, we finished the entire redemption process for the guest user. The user now has access to the Azure AD tenant and the application.

This guest user was added to the Azure AD tenant manually using the Azure portal. However, it is also possible to use PowerShell and Microsoft Graph to bulk-add a collection of users to the tenant. In addition to that, PowerShell and Graph both offer more configuration possibilities for adding guest users to Azure AD B2B that are not available from the Azure portal. In this chapter, we will cover what the Graph API and PowerShell have to offer and will add some guest users to Azure AD using both.

Introducing the Graph API

Microsoft Graph is a set of APIs that connects multiple Azure services and provides a single endpoint for developers to use in custom applications. Microsoft Graph is made up of relationships between various Azure services. By calling the endpoint on behalf of a user (delegated permissions) who is added to Azure AD, you can retrieve the documents that the user is working on, find their manager, retrieve the user's meetings, get a list of devices, and much more. It is also possible to call the Graph API based on application permissions, where the application acts as its own entity, instead of on behalf of the user. This means you can retrieve data from the Graph API for multiple accounts.

Azure AD is integrated in Microsoft Graph as well, but it can be leveraged for more than Azure AD features only. In fact, nearly all SaaS products of Azure use Azure AD, such as Office 365, Intune, Dynamics 365, and Azure SQL. All of those Azure services are integrated in Microsoft Graph and can be leveraged inside your apps and APIs.

Microsoft Graph offers two different endpoints: the v1.0 endpoint, which consists of all the APIs that are generally available, and the beta endpoint, which provides APIs that can still change over time.

Tip: To get started with the Graph API using the Graph Explorer, and to register your app in Azure AD, you can refer to this site. This is a great starting point where you can also download secure sample applications for multiple programming languages. You can also get an overview of all the different services and products that are integrated in the Graph API.

Graph API for Azure AD B2B

The Graph API for Azure AD B2B offers some additional values that cannot be set when you create a guest user from the Azure portal. The Azure portal only adds support for the display name, first and last name, email address, personal message, and job information.

Using the Graph API, you can set additional values for each user that you want to invite to your Azure AD tenant. You can set the following values:

- **Invitation Redirect URL**: You can redirect users to different locations, such as a custom web app or a SharePoint site.
- **CC Recipients**: You can add others to the CC field of the email, such as an administrator account.
- **Suppress invitation email**: You can suppress the invitation on a user level. In the portal, you can turn off sending the invitation email, but this setting will result in suppressing all invitations for each user you add to the tenant.

One of the APIs that the Graph API offers is an invitation API that can be used to add guest users to your Azure AD tenant. In the following demonstration, we are going to use the Graph Explorer. This lets us easily connect to our tenant by logging in with an administrator account.

1. Open the browser and navigate to https://developer.microsoft.com/en-us/graph/graph-explorer.
2. In the left menu, under **Authentication**, click **Sign in to Graph Explorer**:

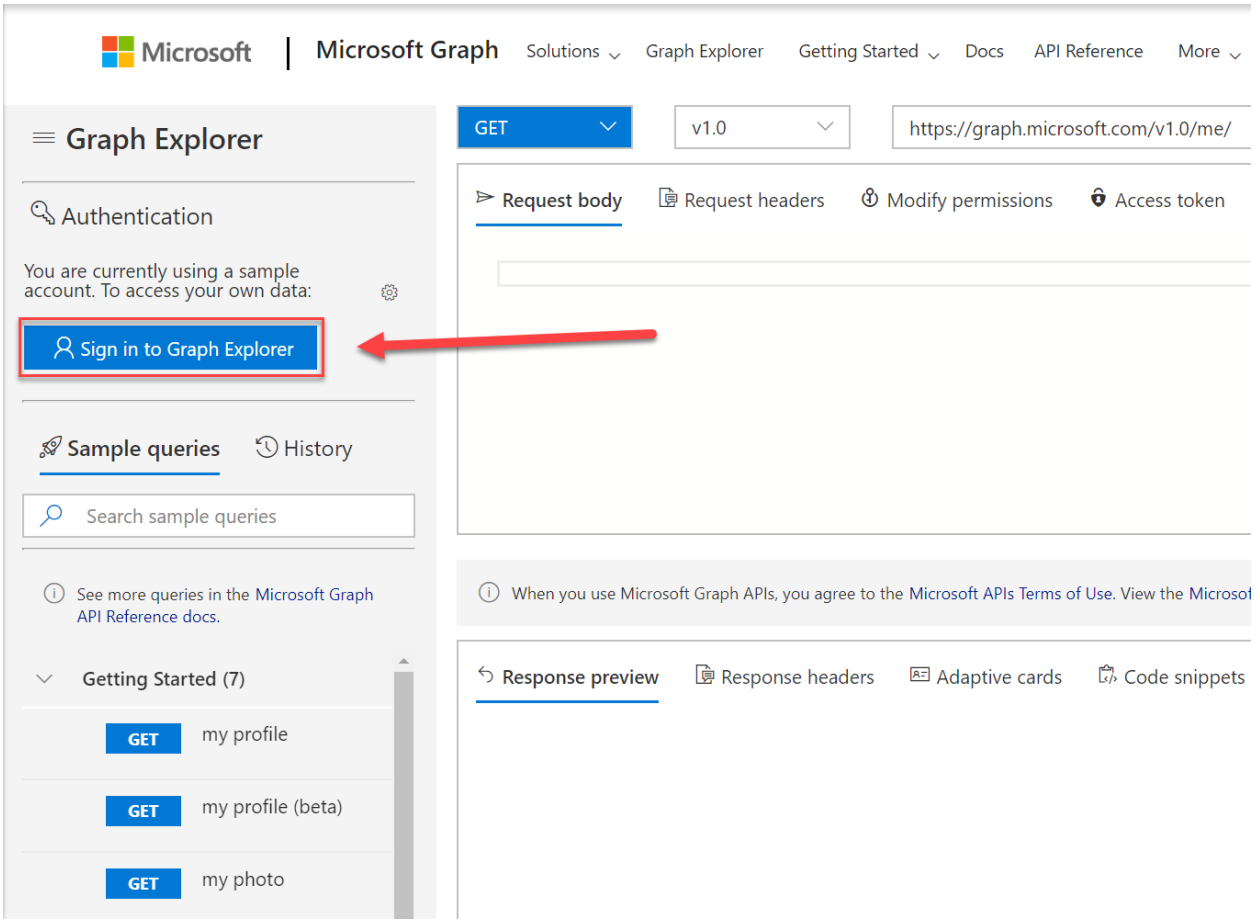

Figure 24: Graph Explorer login

3. Log in with your Azure administrator credentials.

4. Once logged in, we first need to set the permissions for the request. For the Invitation endpoint, the Azure AD tenant admin must explicitly grant consent for the requested permissions to the Graph Explorer application. We are already logged in with an administrator account, so we can set the permissions now.
5. Click the settings button in the authentication section and click **Select permissions**:

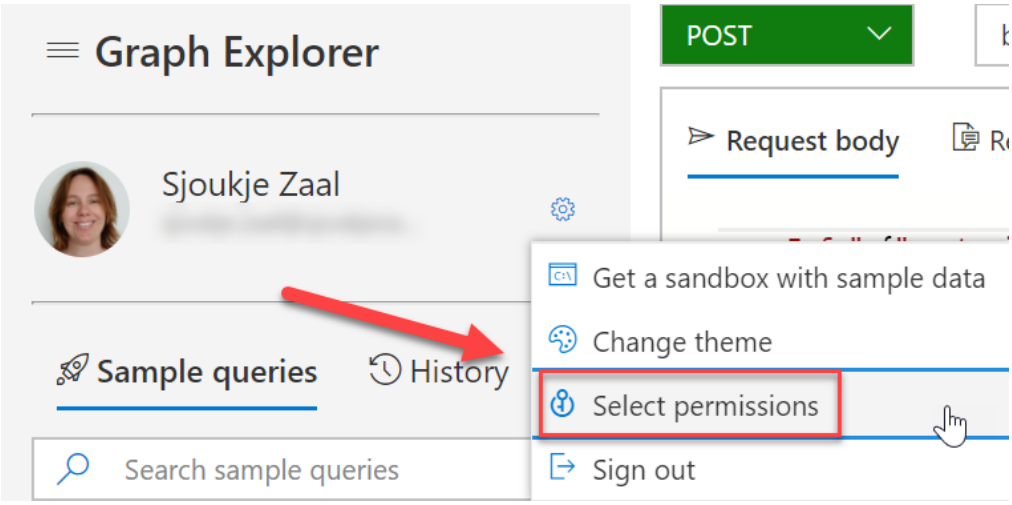

Figure 25: Graph permissions

6. In the Permissions wizard, first, scroll down to **Directory** and select **Directory.ReadWrite.All** permissions:

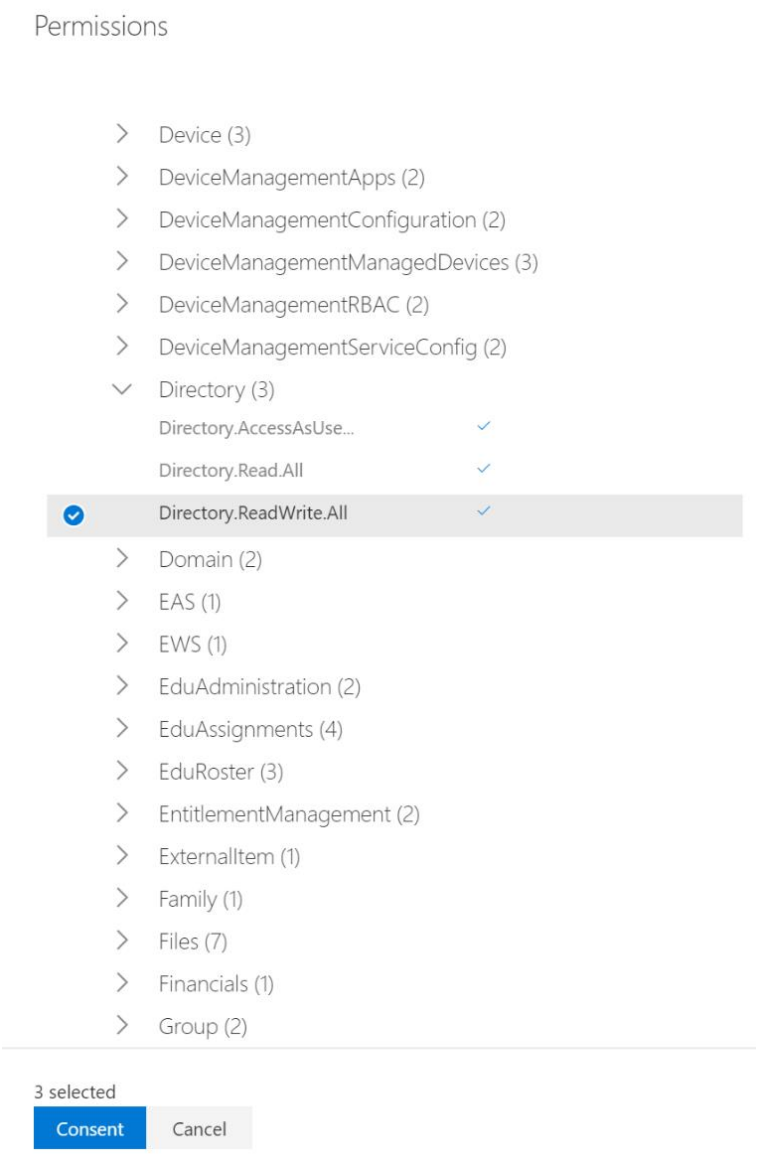

Figure 26: Directory permissions

7. Select **User.Read** and **User.ReadWrite.All**:

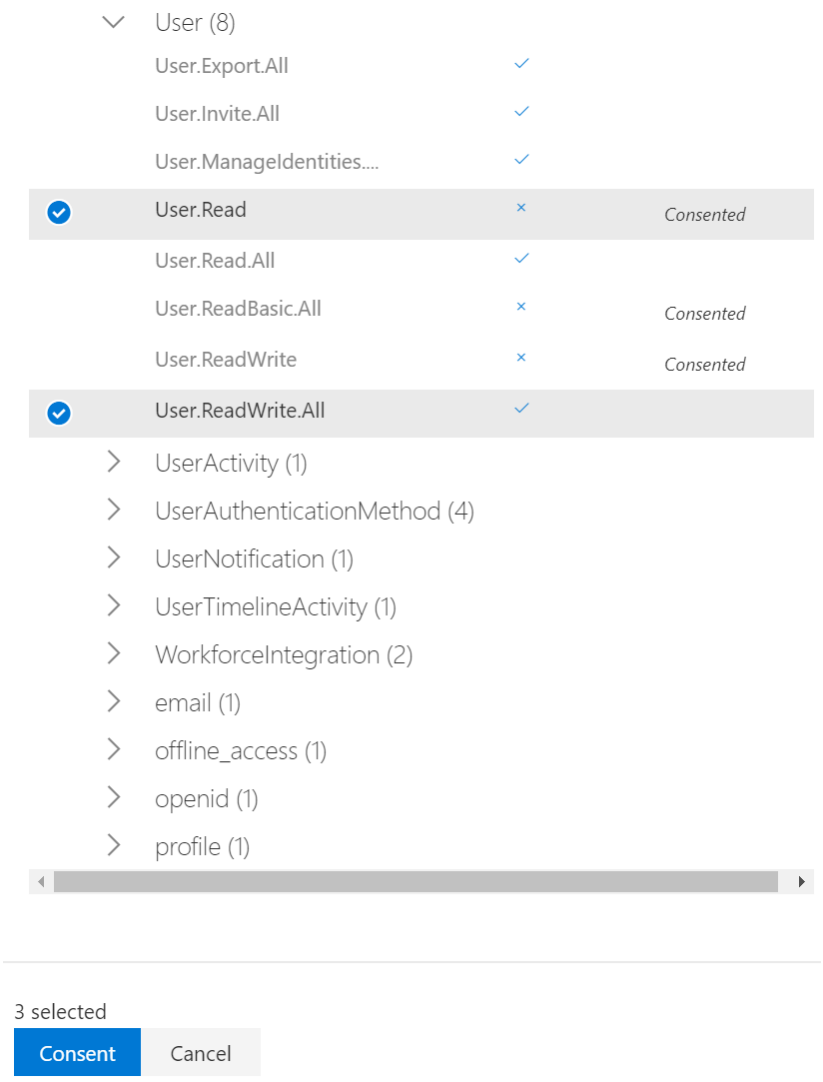

Figure 27: User permissions

8. Click **Consent**.
9. There will be a pop-up with a privacy statement that needs to be accepted. Select the **Consent on behalf of your organization** box:

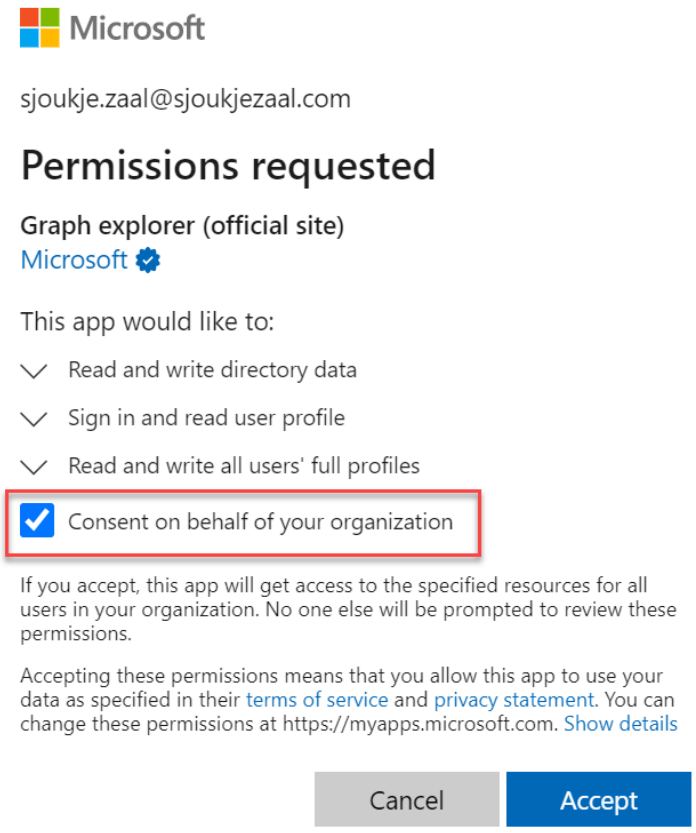

Figure 28: Privacy statement

10. Click **Accept**.
11. Change the request method to **POST** and add the following URL to the invitation API in the URL field at the top right of the screen:

    `https://graph.microsoft.com/v1.0/invitations`

    The page will look like the following image:

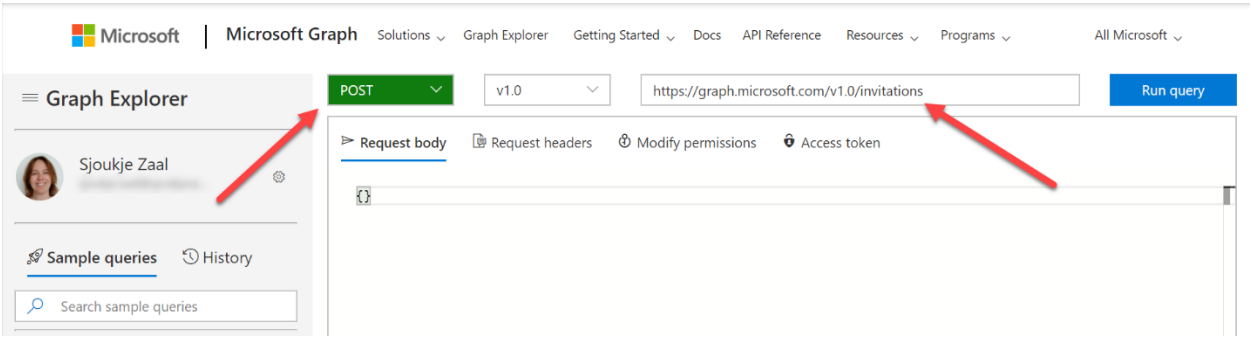

Figure 29: Creating a new Graph request

12. Add the following code, changing the values where needed:

Code Listing 1

```
{
    "invitedUserEmailAddress":"<emailaddress>",
    "inviteRedirectUrl":"https://sjoukjezaal.com",
    "invitedUserDisplayName":"Sjoukje Zaal",
    "sendInvitationMessage":true,
    "invitedUserMessageInfo":
        {
            "customizedMessageBody":"Hey there! Check this out. I created
an invitation through the Graph API"
        }
}
```

13. Click **Run query**, and the user will be created in the Azure AD tenant. This will result in the following output:

Code Listing 2

```
{
    "@odata.context":
"https://graph.microsoft.com/v1.0/$metadata#invitations/$entity",
    "id": "2305df68-fd6c-4dbc-add8-e1abb9afcec7",
    "inviteRedeemUrl":
"https://login.microsoftonline.com/redeem?rd=https%3a%2f%2finvitations.microso
ft.com%2fredeem%2f%3ftenant%3d8db5b030-89df-4359-8d4f-
926af1102a4e%26user%3d2305df68-fd6c-4dbc-add8-
e1abb9afcec7%26ticket%3d858YgoOG%252fpImLDGQenwGplt6K8ww%252fcicsUVuJWuH0Ms%25
3d%26ver%3d2.0",
    "invitedUserDisplayName": "Sjoukje Zaal",
    "invitedUserType": "Guest",
    "invitedUserEmailAddress": "szdemo1@outlook.com",
    "sendInvitationMessage": true,
    "inviteRedirectUrl": "https://sjoukjezaal.com/",
    "status": "PendingAcceptance",
    "invitedUserMessageInfo": {
        "messageLanguage": null,
        "customizedMessageBody": "Hey there! Check this out. I created an
invitation through the Graph API",
        "ccRecipients": [
            {
                "emailAddress": {
                    "name": null,
                    "address": null
                }
            }
```

```
        ]
    },
    "invitedUser": {
        "id": "d776de55-eb95-44e1-b2a9-095cf410e579"
    }
}
```

We have now created a guest user in the Azure AD tenant using the Graph API. We also set some additional values that are not available in the Azure portal, such as the display name and an alternative landing page.

In the next section, we are going to cover how you can add guest users using PowerShell.

PowerShell for Azure AD B2B

Another way to add guest users is by using PowerShell. PowerShell uses the Microsoft Graph for creating the guest user as well. Therefore, you need to download Azure Active Directory PowerShell for Graph from the PowerShell gallery. You can use an Excel file as the input file and add all the guest users to the Excel file. You can also store your guest users in a database or Azure Storage account and connect to it from your PowerShell script.

Because PowerShell is using Graph to create the guest user, PowerShell possesses the same functionality as Graph itself. In the next demonstration, we are going to add two guest users from an Excel file to Azure AD using a PowerShell script and PowerShell cmdlets.

Note: You can refer to this website to install Azure Active Directory PowerShell for Graph.

1. Install the Azure Active Directory PowerShell for Graph cmdlets on your local machine.
2. Open Excel, and add a table with two columns and the following two headers to the sheet:
    a. Name
    b. InvitedUserEmailAddress
3. Add the email addresses and display names of the guest users that you want to add to Azure AD. Your Excel file will look like the following image:

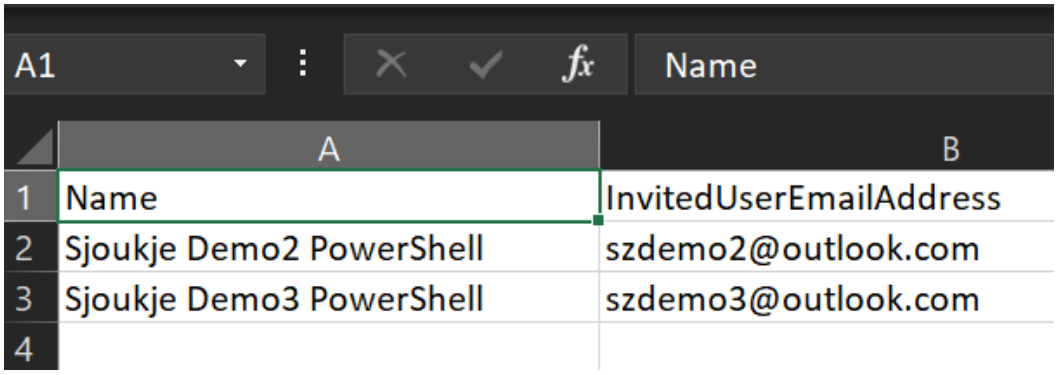

Figure 30: Creating a new Graph request

4. Save the Excel sheet as **invitations.csv**.
5. Open Visual Studio Code or PowerShell ISE and add the following lines of code to it:

Code Listing 3

```
$tenantId = "<replace-with-AzureAD-tenant-id>"

$cred = Get-Credential
Connect-AzureAD -Credential $cred -TenantId $tenantId

$invitations = import-csv "C:\....\invitations.csv"
$messageInfo = New-Object
Microsoft.Open.MSGraph.Model.InvitedUserMessageInfo
$messageInfo.customizedMessageBody = "Hey there! Check this out. I created
an invitation through PowerShell"

foreach ($email in $invitations) {
    New-AzureADMSInvitation -InvitedUserEmailAddress
$email.InvitedUserEmailAddress `
                            -InvitedUserDisplayName $email.Name `
                            -InviteRedirectUrl https://myapps.microsoft.com
`
                            -InvitedUserMessageInfo $messageInfo `
                            -SendInvitationMessage $true
                           }
```

6. Replace the tenant ID with your Azure AD tenant ID. You can retrieve this value from the Azure portal. Make sure to set the right location to where the Excel file is saved.
7. Execute the script. The script will prompt you to log in to Azure. Provide your Azure administrator credentials. The script will now create the user accounts from the Excel sheet in Azure AD.
8. Log in to the Azure portal, navigate to Azure AD, and select **Users** from the left menu. There you will see both the accounts that are created:

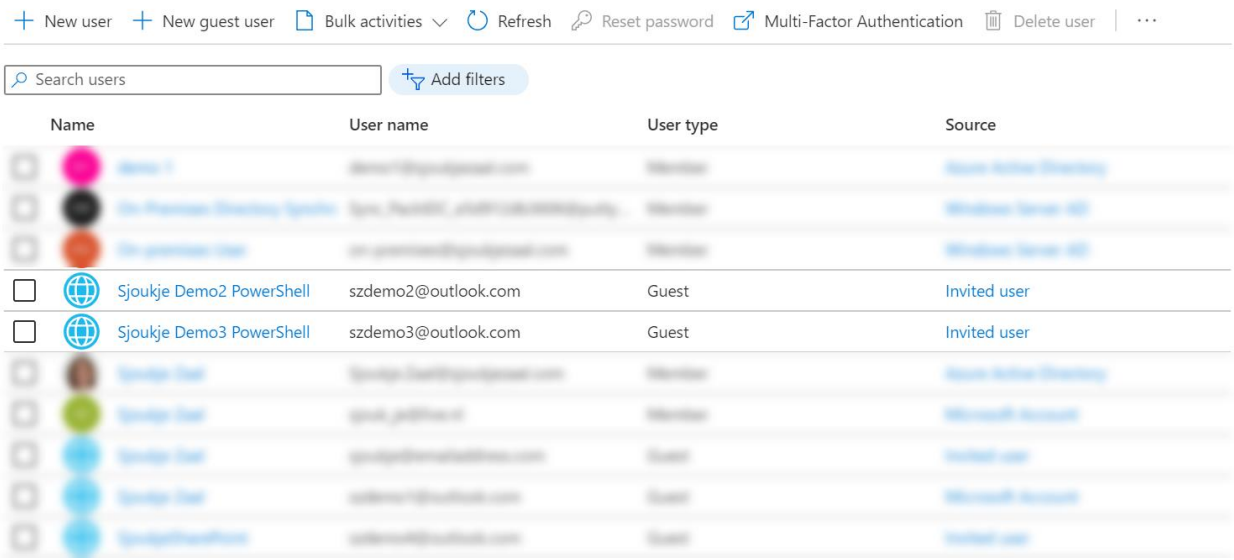

Figure 31: Accounts in Azure AD created with PowerShell

In this chapter, we have covered how you can use PowerShell and the Graph API to create guest users in Azure AD B2B. We have learned that PowerShell uses the Graph API as well to create the guest users. We have also learned that we set more properties for the guest user if we create them programmatically, and then add them manually to Azure AD using the Azure portal.

In the next chapter, we are going to focus on how we can create guest users from an Azure function.

Chapter 5 Automating Adding Guest Users with Azure Functions

In the previous chapter, we covered how you can add guest users to Azure AD using the Graph API and PowerShell. In this chapter, we are going to add external users using Azure Functions. For this, we are going to call the Graph API from an Azure function that is written in C#. Adding users via Azure Functions is slightly more complicated than adding users via PowerShell, but using Azure Functions gives you complete, fine-grained, programmatic control over the process.

Before we can call the Graph API, we need to register an application in Azure AD that has the right permissions to access the Graph API. We already did this in the previous chapter. We then set these permissions directly in the Graph Explorer.

In the next section, we are going to register the app in Azure AD.

Create an Azure AD app registration

1. Navigate to the Azure portal by opening https://portal.azure.com/.
2. In the left menu, select **Azure Active Directory**.
3. The Azure AD overview blade is opened. In the left menu, select **App registrations**:

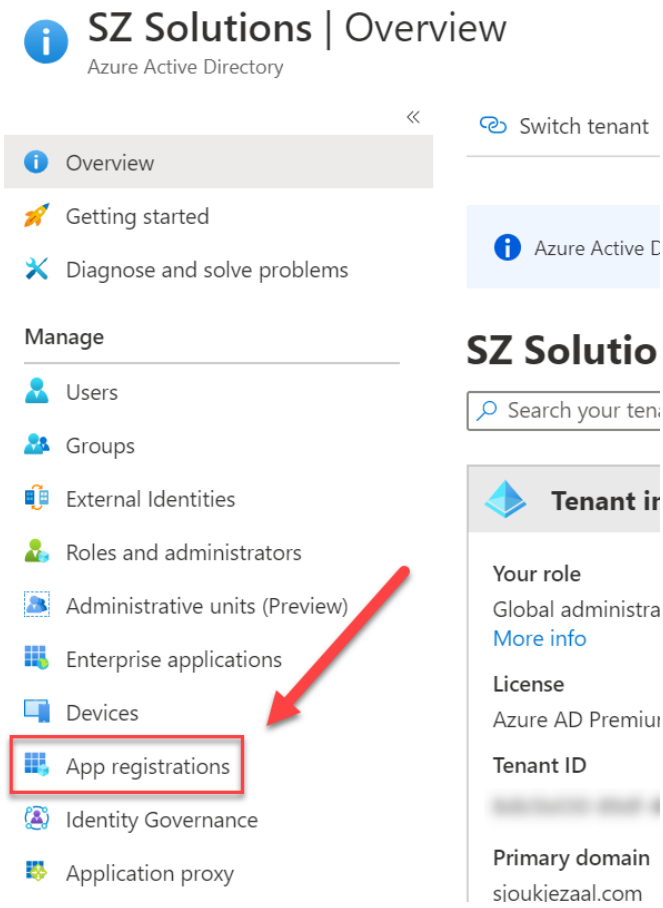

Figure 32: App registrations in Azure AD

4. Select **+ New registration** in the top menu:

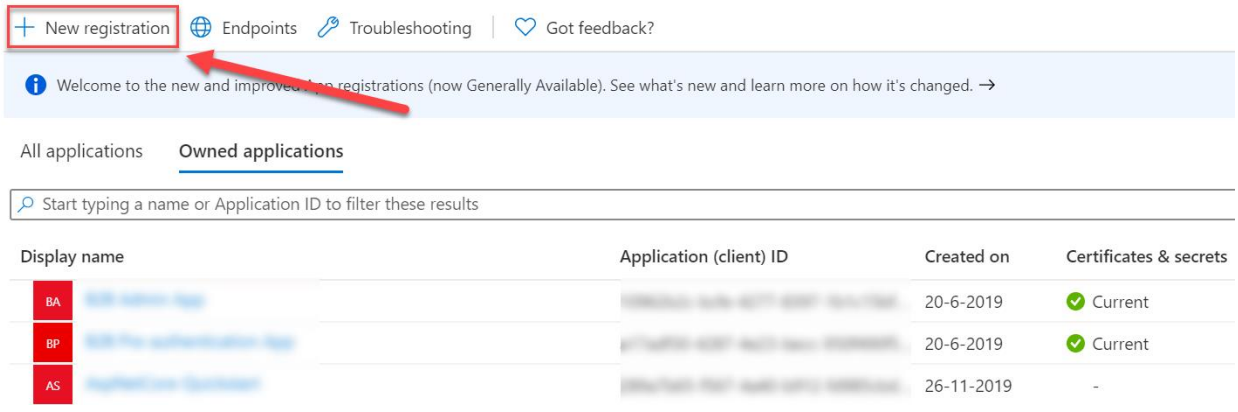

Figure 33: Create a new registration

5. Add the following values:
   a. Name: **FunctionB2B**
   b. Supported account types: **Account in this organizational directory only (Single tenant)**:

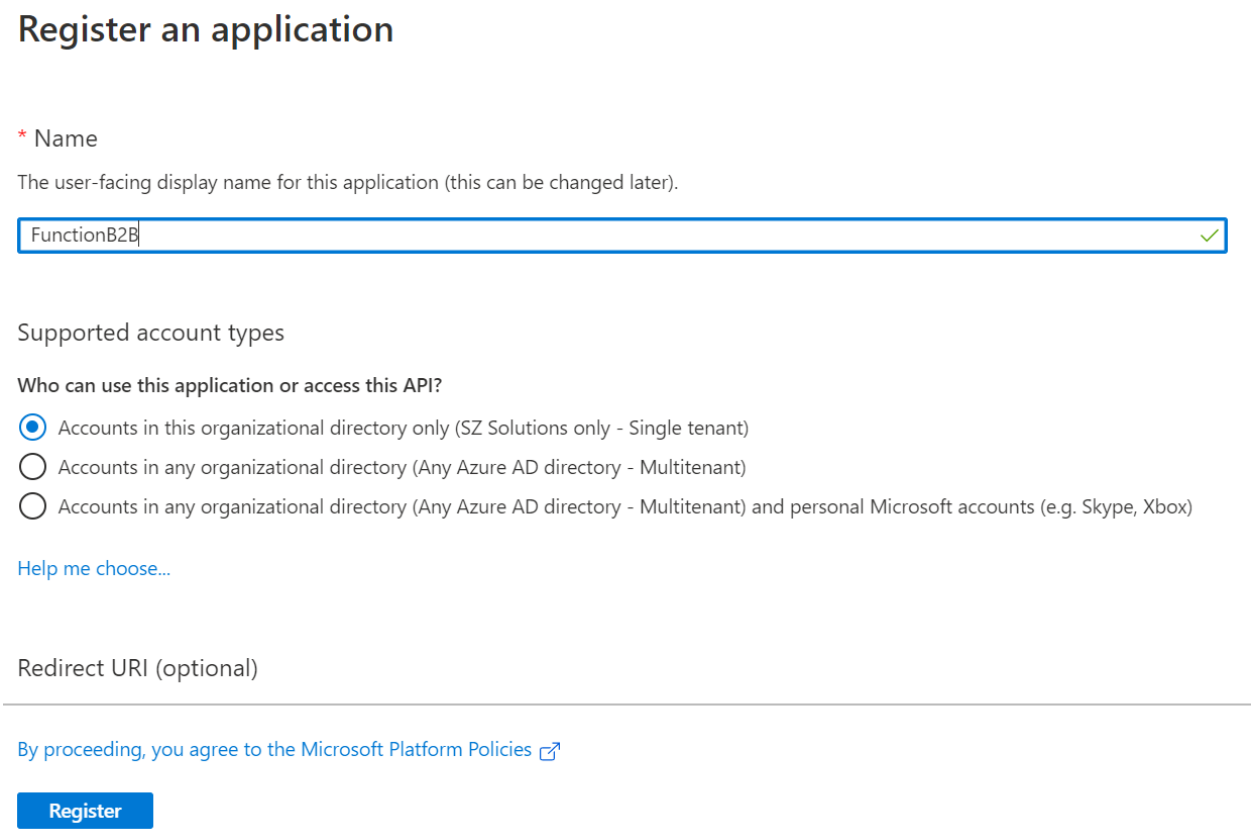

Figure 34: App settings

6. Click **Register**.
7. Once the app is created, you will be redirected to the overview page of the app. We now need to set the required permissions. In the left menu, select **API permissions**.
8. For this app, we need to set the same permissions as in the previous chapter when we added the required permissions to the Graph Explorer. In the top menu, select **+ Add a permission**:

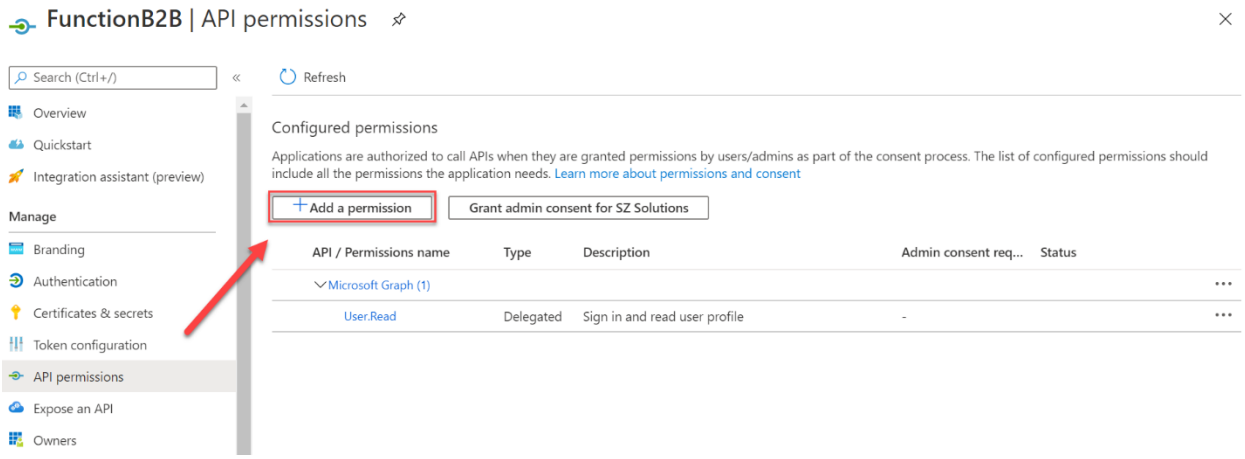

Figure 35: API permissions

9. In the next blade, select **Microsoft Graph**. We are again using Microsoft Graph for creating the guest user.
10. Click **Application permissions**:

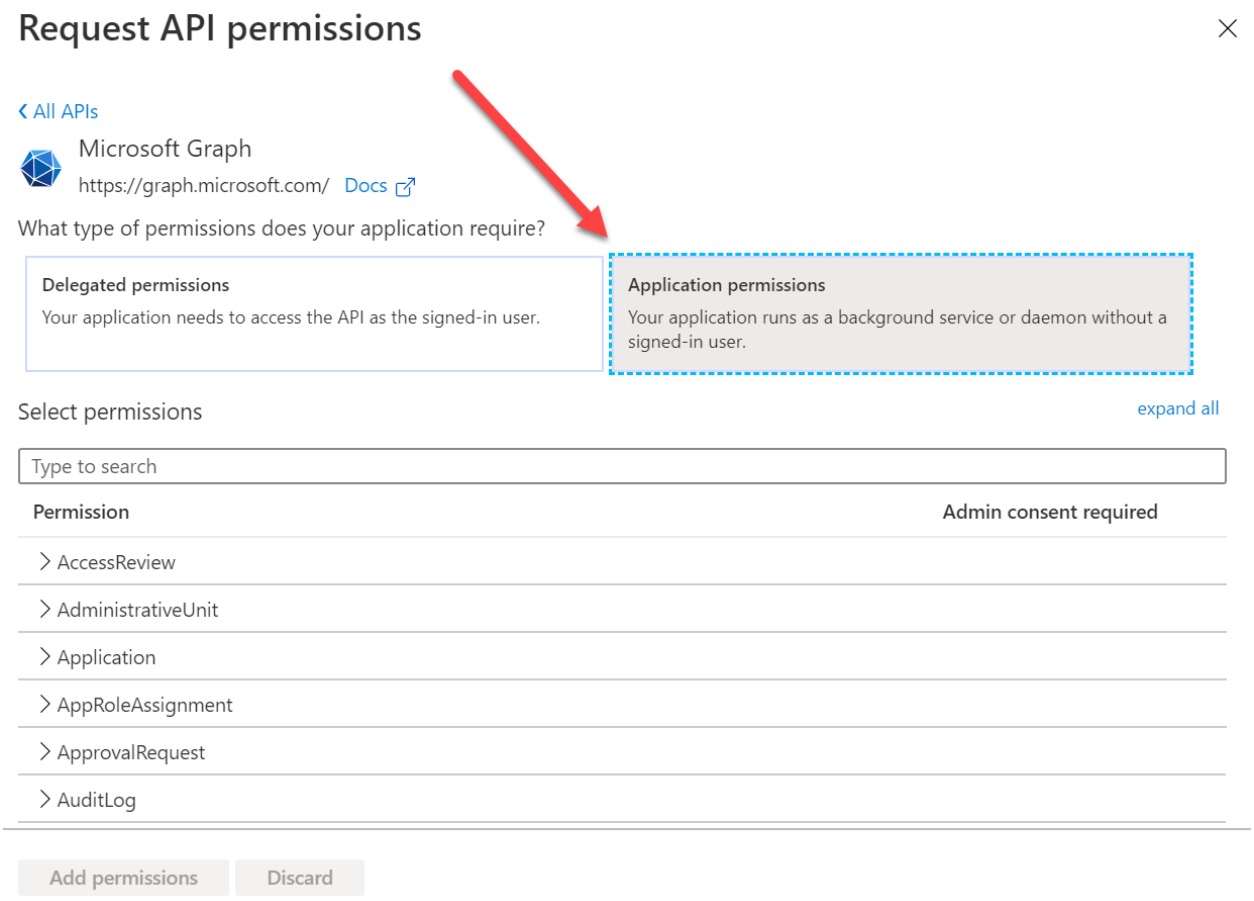

Figure 36: Application permissions

11. Add the following permissions:
    a. User.ReadWrite.All
    b. Directory.ReadWrite.All
12. Click **Add permissions**.
13. Click **Grant admin consent** for the permissions:

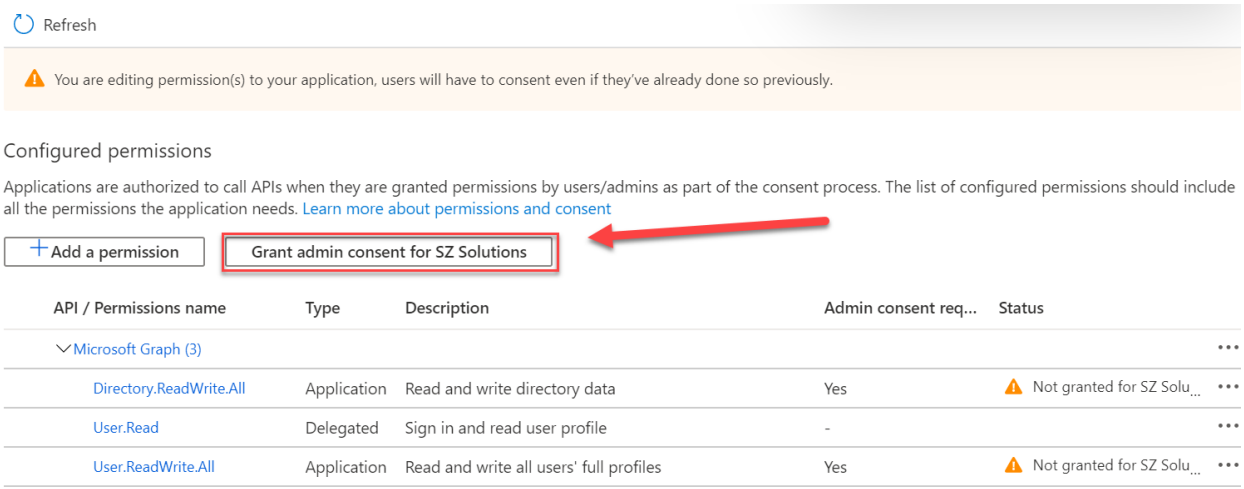

Figure 37: Grant admin consent

14. The last step is to create an app secret. We need this, together with the Azure AD tenant ID and the application ID, to make a request from our Azure Function that we are going to create in the next step. In the left menu, click **Certificates & secrets**.
15. Under **Client secrets**, click **+ New client secret**:

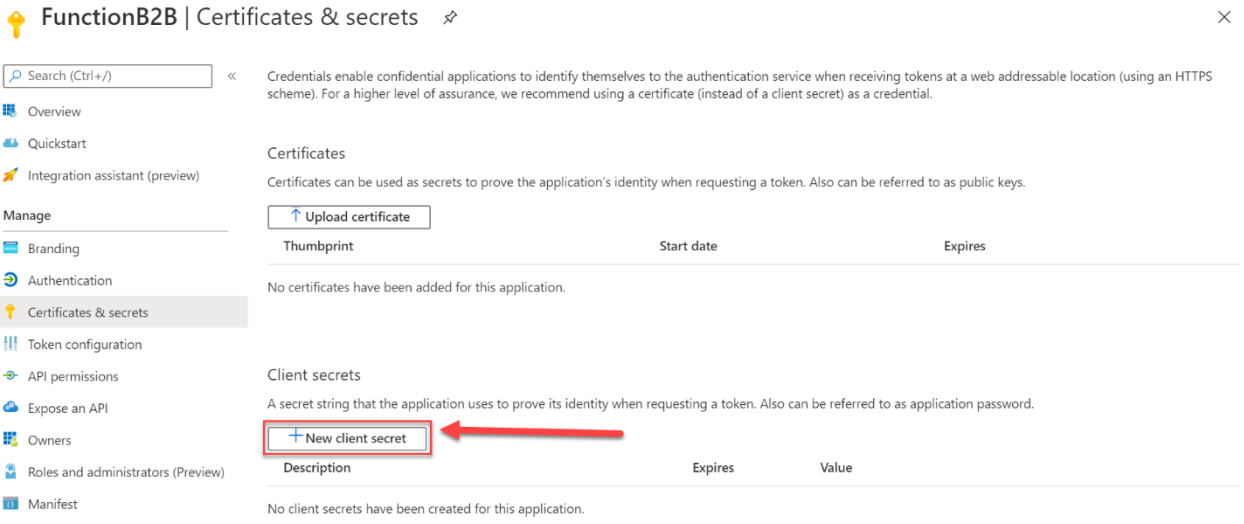

Figure 38: Add new client secret

16. Specify a name for the secret and set when you want it to expire:

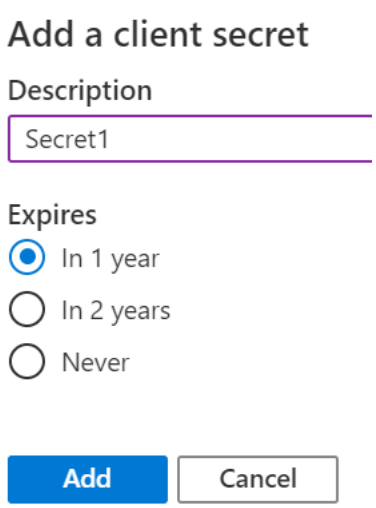

Figure 39: Set client secret values

17. Click **Add** to create the secret.
18. Copy the newly created secret's Value field to Notepad after creation, because it is only going to be displayed to you once:

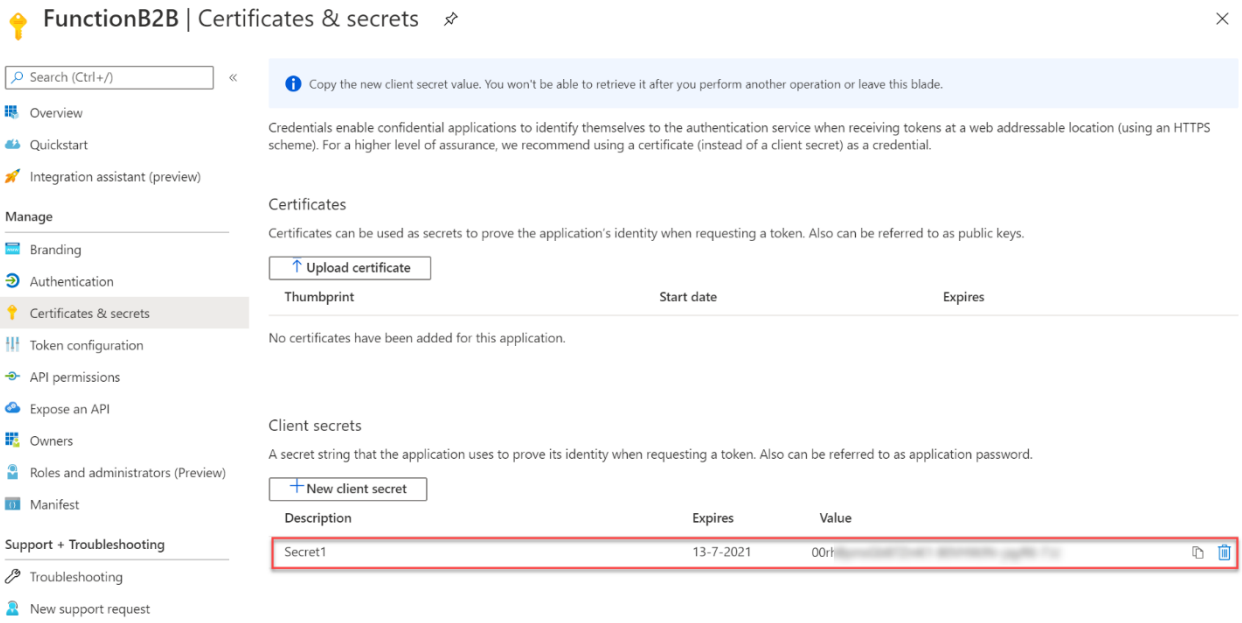

Figure 40: App secret value

19. Navigate to the overview page of the app registration and copy the app ID and the tenant ID to Notepad as well:

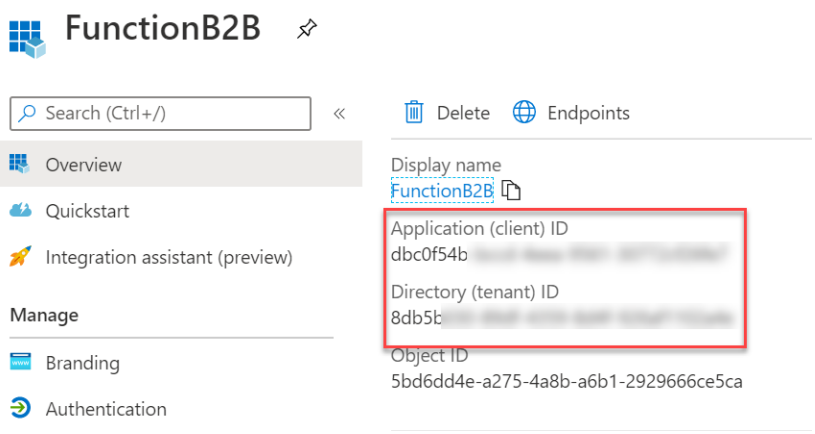

Figure 41: App ID and Azure AD tenant ID

We now have registered an app in Azure AD that has permissions to call Microsoft Graph. This means that we can create the guest user from our Azure function through this registration.

In the next part, we are going to create the function.

Adding guest users using an Azure function

In this demonstration, we are going to create an Azure function to create the guest user in Azure AD. We are using Visual Studio Code in this example to create the function.

1. Open Visual Studio Code. First, we need to install the Azure Functions extension. If you already have it installed, you can skip this step. Open the Extensions screen by clicking it in the left menu or by pressing **Ctrl**+**Shift**+**X**.
2. Search for **Azure Functions** in the search box, select it, and then click **Install**:

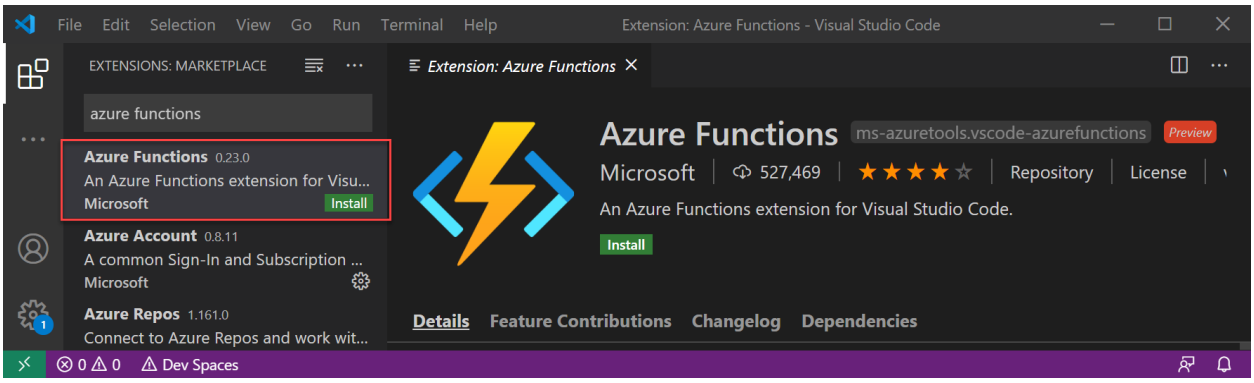

Figure 42: Install the Azure Functions extensions

3. After installing the Azure Functions extension, open the Command Palette by pressing **Ctrl**+**Shift**+**P**.
4. Search for **Azure Functions Create** and select **Create Function** from the list:

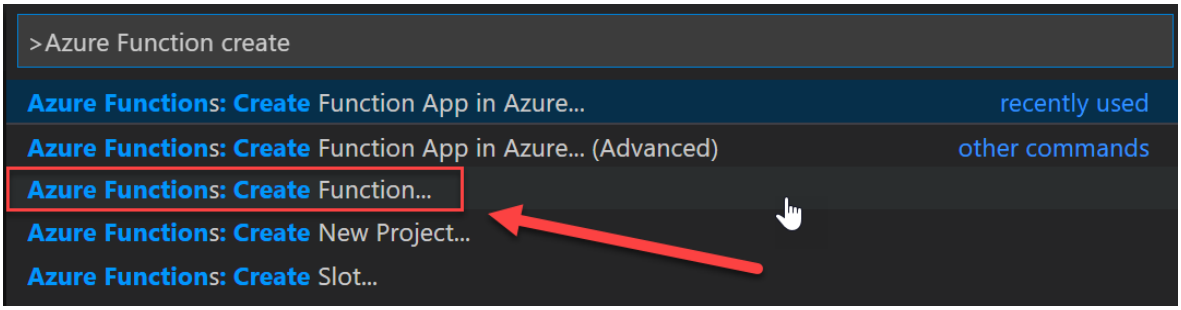

Figure 43: Create a new function

5. You will be prompted to create a new project. Select **Create new project**. Choose a folder to create the project in.
6. Select the language of the project:

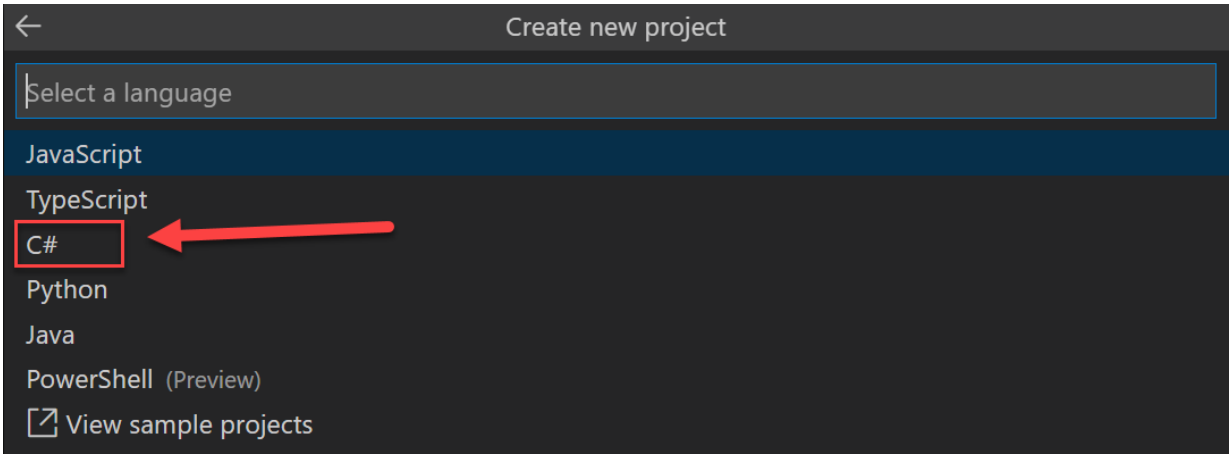

Figure 44: Select the language

7. Select a trigger for the function. We are going to use the **HTTPTrigger** for this demo:

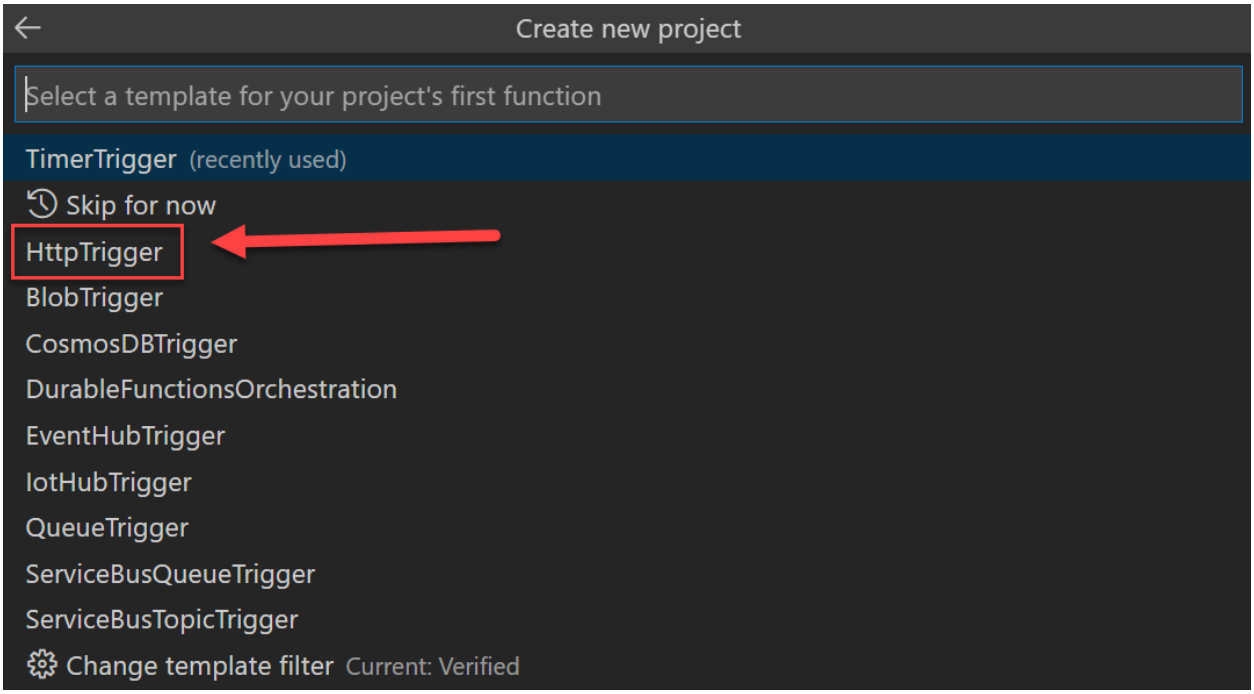

Figure 45: Select the trigger

8. Next, you need to specify a name for the trigger. You can also keep the default name and press **Enter**:

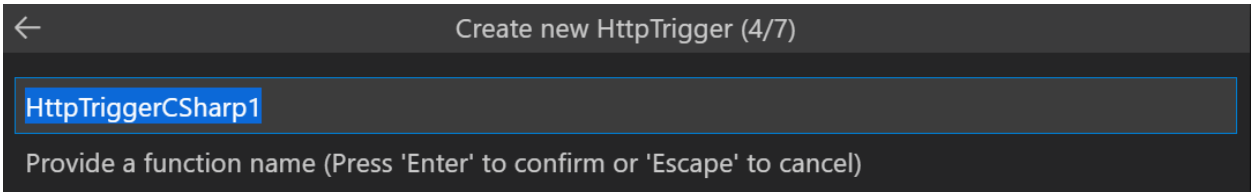

Figure 46: Specify a name for the trigger

9. Specify a name for the function, such as **CreateGuestUserGraph**, and press **Enter**:

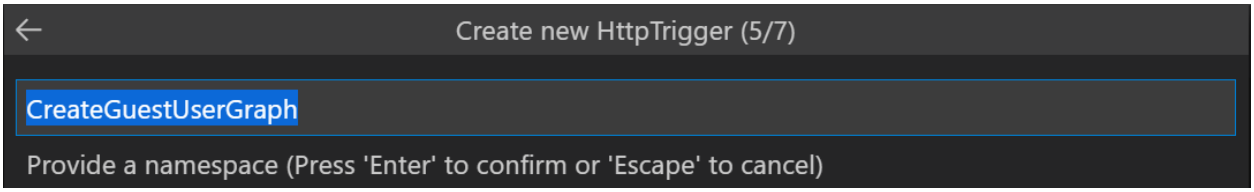

Figure 47: Specify a name for the function

10. For the **AccessRights**, select **Anonymous** and press **Enter**:

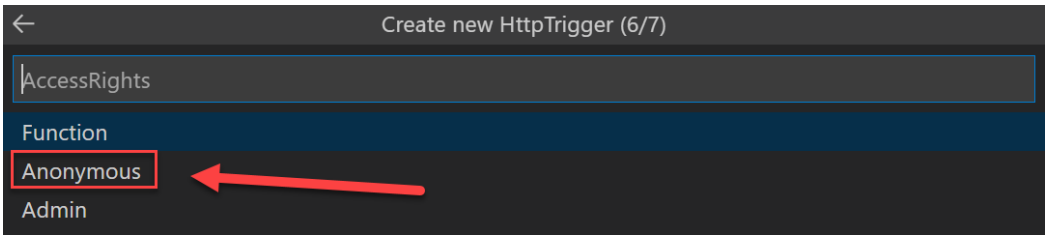

Figure 48: Select the Access Rights

11. To choose how you would like to open the newly created project, select **Add to workspace**:

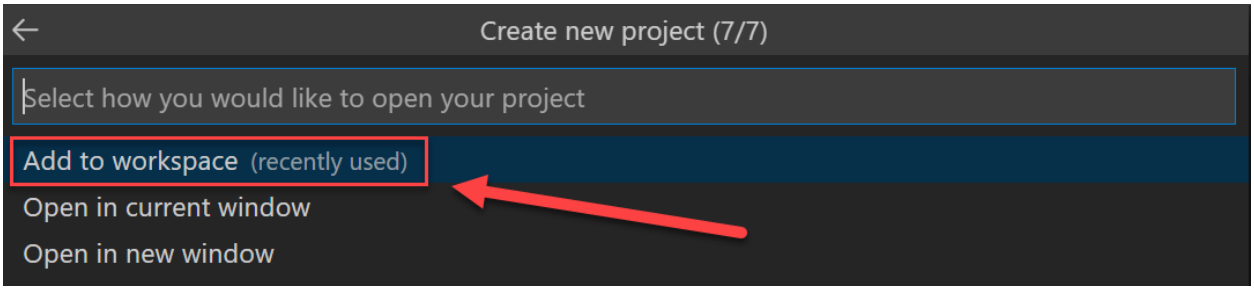

Figure 49: Open the project

12. The project is generated and added to the workspace in VS Code.
13. We need to add two NuGet packages to the project to be able to call Microsoft Graph and use the Microsoft Authentication Library (MSAL) .NET SDK.

Tip: MSAL enables developers to acquire tokens from the Microsoft identity platform endpoint to access secured web APIs. For more information, you can refer to this website.

14. Open a new terminal by pressing **Ctrl**+**Shift**+` and enter the following lines to add the required packages:

```
dotnet add package Microsoft.Graph
dotnet add package Microsoft.Identity.Client
```

15. Now that the libraries are added to the project, we can add the Azure AD app registration configuration values to the **local.settings.json** file. Open the file from the project and add the following code below the Functions Worker Runtime line:

Code Listing 4

```
"AzureADAppId": "YOUR_CLIENT_ID",
"AzureADTenantId": "YOUR_TENANT_ID",
"AzureADAppSecret": "YOUR_CLIENT_SECRET"
```

16. Add the values that we have copied to Notepad here.

17. Your file should now look like the following image:

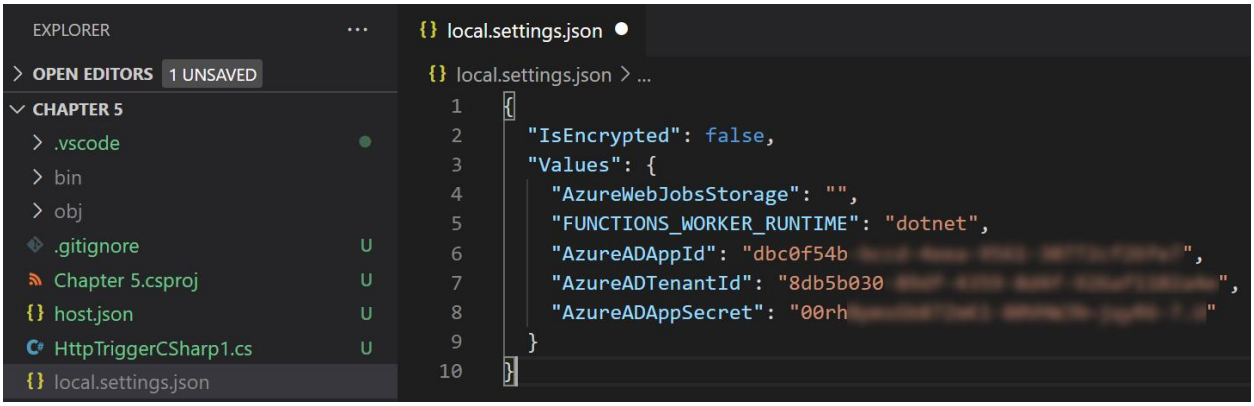

Figure 50: Azure AD registration values

18. Add a new file to the project and name it **AuthHandler.cs**. Add the following code to it:

Code Listing 5

```
using System.Net.Http;
using System.Threading.Tasks;
using Microsoft.Graph;
using System.Threading;

namespace CreateGuestUserGraph
{
     public class AuthHandler : DelegatingHandler {
        private IAuthenticationProvider _authenticationProvider;

        public AuthHandler(IAuthenticationProvider authenticationProvider,
HttpMessageHandler innerHandler)
        {
            InnerHandler = innerHandler;
            _authenticationProvider = authenticationProvider;
        }

        protected override async Task<HttpResponseMessage> SendAsync(HttpRe
questMessage request, CancellationToken cancellationToken)
        {
            await _authenticationProvider.AuthenticateRequestAsync(request)
;
            return await base.SendAsync(request,cancellationToken);
        }
}
}
```

19. Add another new file to the project and name it **MSALAuthenticationProvider.cs**. Add the following code to it:

Code Listing 6

```
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using Microsoft.Identity.Client;
using Microsoft.Graph;

namespace CreateGuestUserGraph
{
     // This class encapsulates the details of getting a token from MSAL an
d exposes it via the
    // IAuthenticationProvider interface so that GraphServiceClient or Auth
Handler can use it.
    // A significantly enhanced version of this class will be available fro
m the GraphSDK team
    // in the future. It will support all the types of
    //Client Application as defined by MSAL.
    public class MsalAuthenticationProvider : IAuthenticationProvider
    {
        private IConfidentialClientApplication _clientApplication;
        private string[] _scopes;

        public MsalAuthenticationProvider(IConfidentialClientApplication cl
ientApplication, string[] scopes) {
            _clientApplication = clientApplication;
            _scopes = scopes;
        }

        /// <summary>
        /// Update HttpRequestMessage with credentials
        /// </summary>
        public async Task AuthenticateRequestAsync(HttpRequestMessage reque
st)
        {
            var token = await GetTokenAsync();
            request.Headers.Authorization = new AuthenticationHeaderValue("
bearer", token);
        }

        /// <summary>
        /// Acquire Token
        /// </summary>
        public async Task<string> GetTokenAsync()
        {
            AuthenticationResult authResult = null;
            authResult = await _clientApplication.AcquireTokenForClient(_scopes)
```

```
                                .ExecuteAsync();
            return authResult.AccessToken;
        }
    }
}
```

20. Open the **HTTPTriggerCSharp1.cs** file and add the following code to it:

Code Listing 7

```
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;
using Microsoft.Graph;
using System.Collections.Generic;
using Microsoft.Identity.Client;

namespace CreateGuestUserGraph
{
    public static class HttpTriggerCSharp1
    {
        private static GraphServiceClient _graphServiceClient;

        [FunctionName("HttpTriggerCSharp1")]
        public static async Task<IActionResult> Run(
            [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post", Route
 = null)] HttpRequest req,
            ILogger log)
        {
            log.LogInformation("C# HTTP trigger function processed a reques
t.");

            string firstname = req.Query["firstname"];
            string lastname = req.Query["lastname"];
            string emailaddress = req.Query["emailaddress"];

            string requestBody = await new StreamReader(req.Body).ReadToEnd
Async();
            dynamic data = JsonConvert.DeserializeObject(requestBody);

            firstname = firstname ?? data?.firstname;
            lastname = lastname ?? data?.lastname;
```

```
            emailaddress = emailaddress ?? data?.emailaddress;

            var invitation = new Invitation
            {
                InvitedUserDisplayName = $"{firstname} {lastname}",
                InvitedUserEmailAddress = emailaddress,
                InviteRedirectUrl= "https://myapplications.microsoft.com/",
                InvitedUserMessageInfo = new InvitedUserMessageInfo {
                    CustomizedMessageBody = "Hey there! Check this out. I c
reated an invitation through an Azure Function!"
                },
                SendInvitationMessage = true
            };

            GraphServiceClient graphClient = GetAuthenticatedGraphClient();
            var graphResult = await graphClient.Invitations
                    .Request()
                    .AddAsync(invitation);

        return graphResult != null
        ? (ActionResult)new OkObjectResult($"Following user created: {first
name}, {lastname}, {emailaddress}")
        : new BadRequestObjectResult("Please provide guest user information
 details on the query string or in the request body.");
        }
        private static GraphServiceClient GetAuthenticatedGraphClient()
        {
            var authenticationProvider = CreateAuthorizationProvider();
            _graphServiceClient = new GraphServiceClient(authenticationProv
ider);
            return _graphServiceClient;
        }
        private static IAuthenticationProvider CreateAuthorizationProvider(
)
        {
            var clientId = System.Environment.GetEnvironmentVariable("Azure
ADAppId", EnvironmentVariableTarget.Process);
            var clientSecret = System.Environment.GetEnvironmentVariable("A
zureADAppSecret", EnvironmentVariableTarget.Process);
           // var redirectUri = System.Environment.GetEnvironmentVariable("
AzureADAppRedirectUri", EnvironmentVariableTarget.Process);
            var tenantId = System.Environment.GetEnvironmentVariable("Azure
ADTenantId", EnvironmentVariableTarget.Process);
            var authority = $"https://login.microsoftonline.com/{tenantId}/
v2.0";

            //this specific scope means that application will default to wh
at is defined in the application registration rather than using dynamic sco
pes
```

```
            List<string> scopes = new List<string>();
            scopes.Add("https://graph.microsoft.com/.default");

            var cca = ConfidentialClientApplicationBuilder.Create(clientId)
                                                .WithAuthority(authority)
                                               // .WithRedirectUri(redirectUr
i)
                                                .WithClientSecret(clientSecre
t)
                                                .Build();

            return new MsalAuthenticationProvider(cca, scopes.ToArray());
        }
    }
}
```

21. To test the Azure function, open a new terminal and run **`dotnet build`**.
22. When the function is built successfully, you can press **F5** to execute the function locally. This will result in the following output:

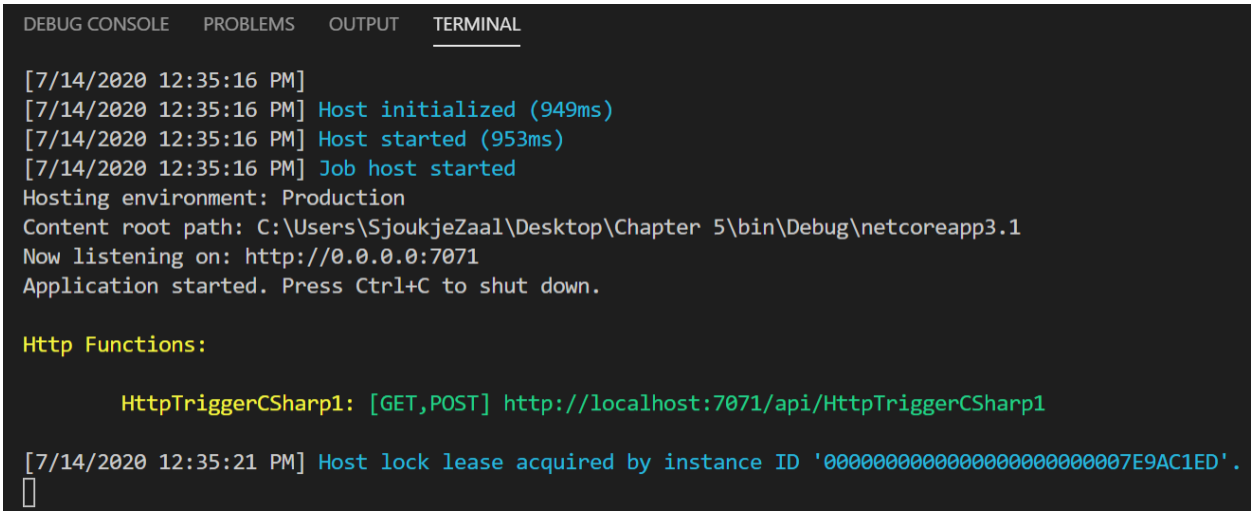

Figure 51: Debugging the function locally

23. The URL for the function is also displayed here. Copy the URL and add the query string to it to make a **POST** request to the Azure function. The URL will look like the following example:

    http://localhost:7071/api/HttpTriggerCSharp1?firstname=Sjoukje&lastname=Zaal&emailaddress=<your-email-address>

24. Open a browser window and paste in the URL. This will result in the following outcome:

Figure 52: Output after creating the guest user

25. Now that the function has executed without any errors, we can deploy it to Azure. We can do this directly from VS Code as well. Press **Ctrl**+**Shift**+**P** again and search for **Azure Functions: Deploy**, and select the following item:

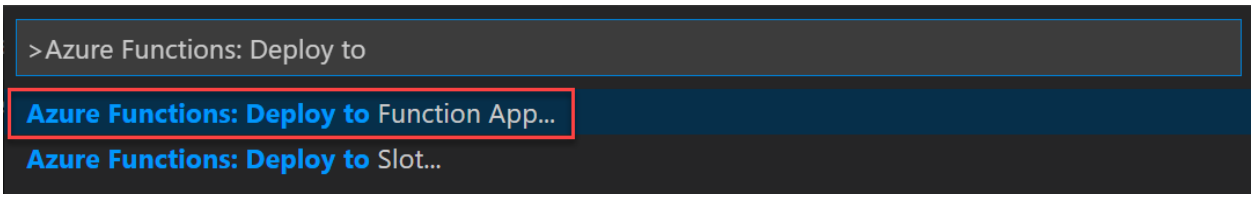

Figure 53: Deploying the function from VS Code

26. Next, you need to sign in to Azure with an administrator account:

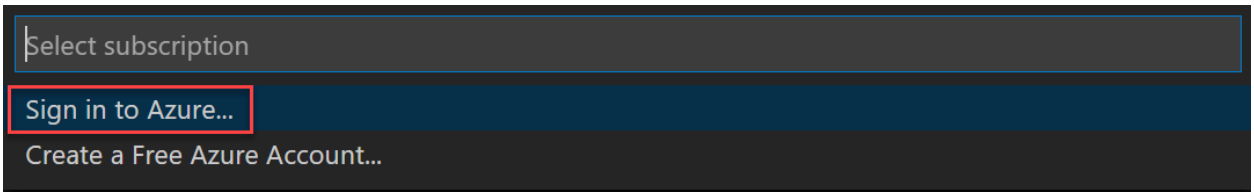

Figure 54: Sign in to Azure

27. Select the subscription you want to deploy the function to.
28. Create a new function app:

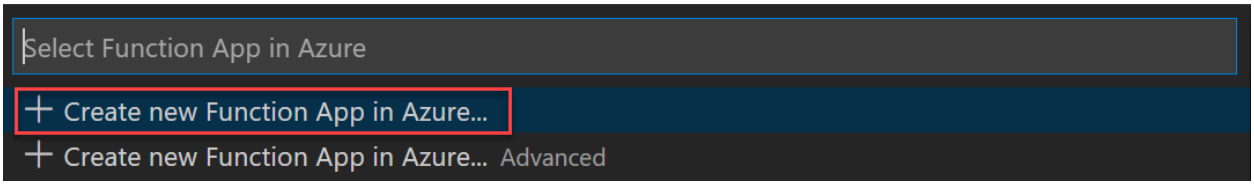

Figure 55: Create a new Function App

29. Specify a unique name for the function and press **Enter**:

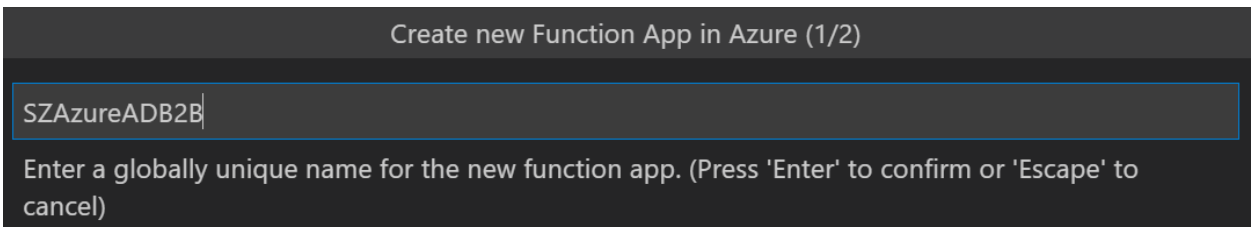

Figure 56: Specify a unique name

30. Pick the region you want to deploy your app to:

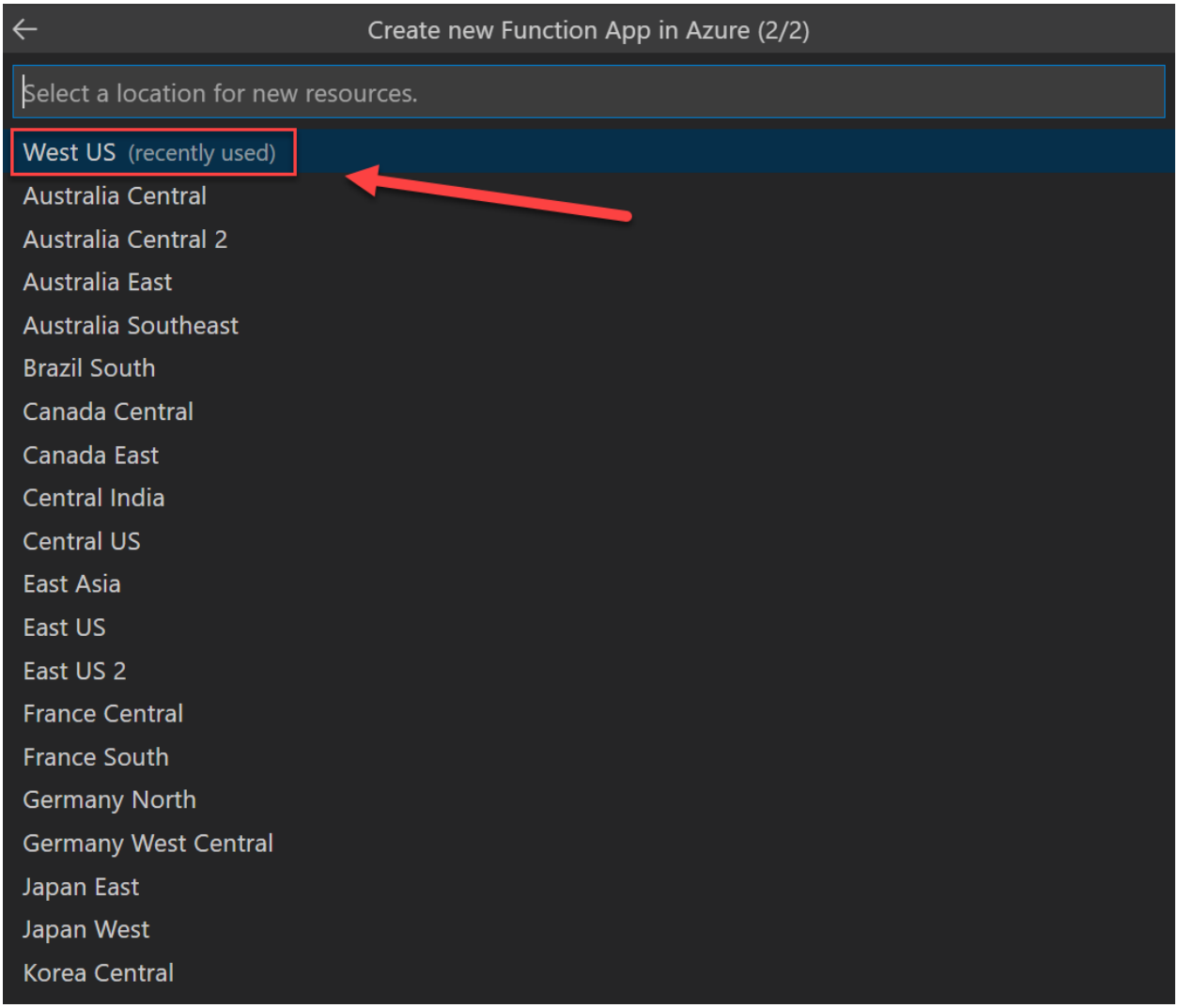

Figure 57: Pick the region in Azure

The Function App will now be deployed to Azure. If you navigate to the Azure portal after deployment, you will see that the function is deployed:

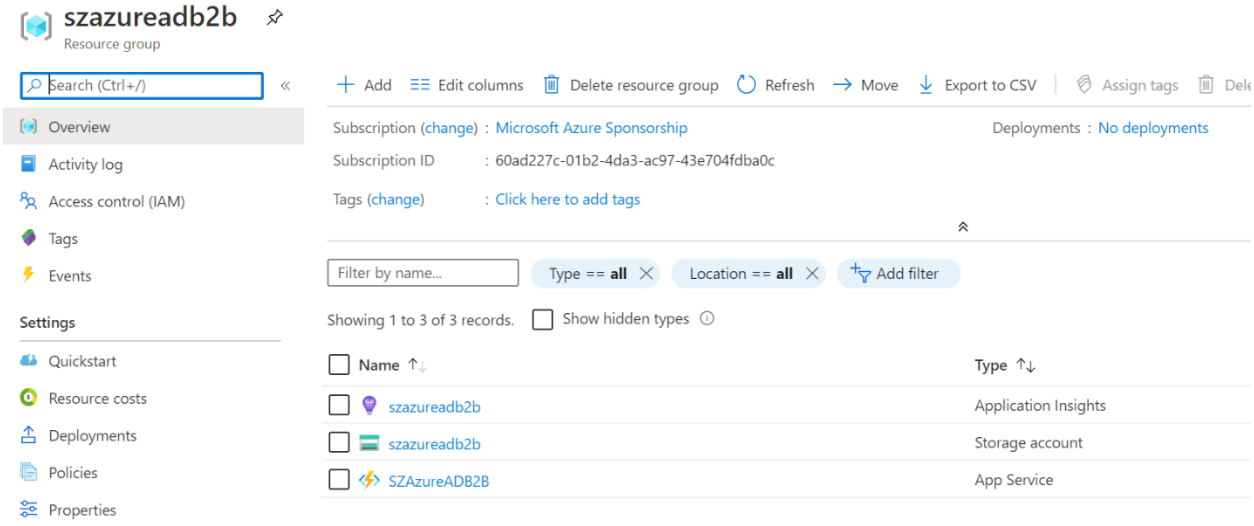

Figure 58: Deployed Azure function

We have now created a guest user in Azure AD using an Azure function written in C#. This function uses the Microsoft.Graph and the Microsoft.Identity.Client NuGet packages, and an Azure AD app registration to get access to the Azure AD tenant.

In this chapter, we have covered how you can add guest users to Azure AD B2B using an Azure AD app registration, an Azure function, and the Graph API. This Azure function was built in VS Code. In the next chapter, we are going to cover how you can manage access using Azure AD B2B entitlement management.

Chapter 6 Entitlement Management

In the previous chapter, we covered how you can use Azure Functions to create guest users in Azure AD. In this chapter, we are going to cover entitlement management, which is part of Azure AD Identity Governance. With entitlement management, organizations can manage identity and access lifecycle at scale.

Before we dive into entitlement management and creating access packages for your guest users, we will first look into Identity Governance in more detail.

Identity Governance

Identity Governance is one of the newest features in Azure AD. It offers tools and support for balancing organizational needs for security and productivity with the right processes and visibility. It provides the ability govern the complete identity and access lifecycle across employees, business partners, vendors, applications, and services, both in the cloud and on-premises. On top of that, it offers secured privileged access for administration.

Identity Governance is mainly intended to help organizations address the following key questions:

- Which users should have access to which resources in Azure and Microsoft 365?
- What are those users doing with all the different services and applications they have access to?
- Are there effective controls for managing access across all the different applications and services?
- Is it possible for auditors to verify that the controls are effectively working?

One of the key challenges of organizations is to maintain balance between productivity and security. You want your services and applications to be accessed quickly and easily when needed, but there is also a need for keeping all those assets secure.

The foundation of Identity Governance is identity lifecycle management. Azure AD Premium has comprehensive integration with the Workday HR system. It also includes Microsoft Identity Manager, which can import records from human capital management (HCM) systems such as Oracle E-Business, Oracle PeopleSoft, and SAP.

Note: For more information about Microsoft Identity Manager, you can refer to this website.

For scenarios that require collaboration with people outside the organization, Azure AD B2B entitlement management offers the tools to manage the complete lifecycle of guest users.

In the next section, we are going to cover entitlement management in more detail.

Entitlement management

Azure Active Directory entitlement management is an identity governance feature that helps organizations manage identity and access lifecycle at scale. Therefore, it offers features for automating access assignments, request workflows, reviews, and access expirations.

Entitlement management can be used for internal and guest users. This can be a cumbersome process because requirements change over time. This is already a difficult process for internal users, but it becomes even more complicated when you are planning to collaborate with external users as well.

Most enterprises face challenges when managing internal access to resources, such as:

- Users don't know exactly what access they have across the different services and applications, and have trouble locating the different applications and services.
- Users have difficulties locating the right individuals to approve their access when needed.
- Once users have the appropriate access and permissions, they hold on to access longer than required.

In scenarios where collaboration with guest users is required, access becomes even more difficult to manage. Additional challenges arise:

- No one person may know all the specific individuals in the external organization's directories. This makes it difficult to invite them.
- When guest users are invited, no one in the organization is able to remember to manage all of these users' access consistently.

Azure AD entitlement management helps address these challenges. It offers the following capabilities:

- Delegate the creation of access packages to non-administrators. Access packages contain all the resources that an internal or guest user can request access to, such as groups, applications, SaaS applications, custom integrated applications, and SharePoint Online sites. The delegated access managers can define rules and policies for which users can request access, who approves their access, and when their access expires.
- In entitlement management, you can select connected organizations whose users can request access. When the request is approved, the user becomes a B2B user and is automatically added to the Azure AD tenant. When the access expires, the user is removed automatically from the tenant as well.

Access packages are not suited for all scenarios. They also don't replace the other mechanisms for access assignment. Access packages are most suited in cases when employees and guest users need limited-time access to a particular task, application, or other resource in Azure or SharePoint Online. They are also suitable when access to certain resources needs to be approved by a manager, for instance. In cases where certain departments or connected organizations are responsible for managing access, access packages are also the best solution.

In the next section, we are going to create an access package in Azure AD entitlement management.

Creating an access package for guest users

In this demonstration, we are going to create an access package for guest users. We are going to add the required permissions to this package, add a security group to it (one where the guest user is automatically added), and add permissions to an application.

Therefore, we are going to take the following steps:

1. Navigate to the Azure portal by opening https://portal.azure.com/.
2. In the left menu, select **Azure Active Directory**.
3. The Azure AD overview blade is opened. In the left menu, select **Identity Governance**:

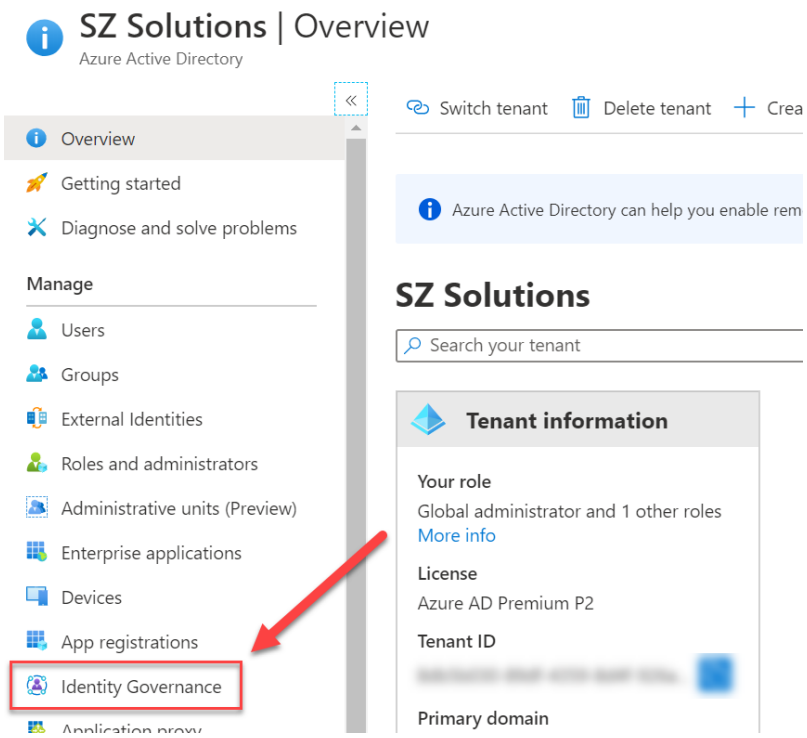

Figure 59: Identity Governance in the Azure portal

4. In the Identity Governance overview blade, click **Create an access package** on the **Getting started** page:

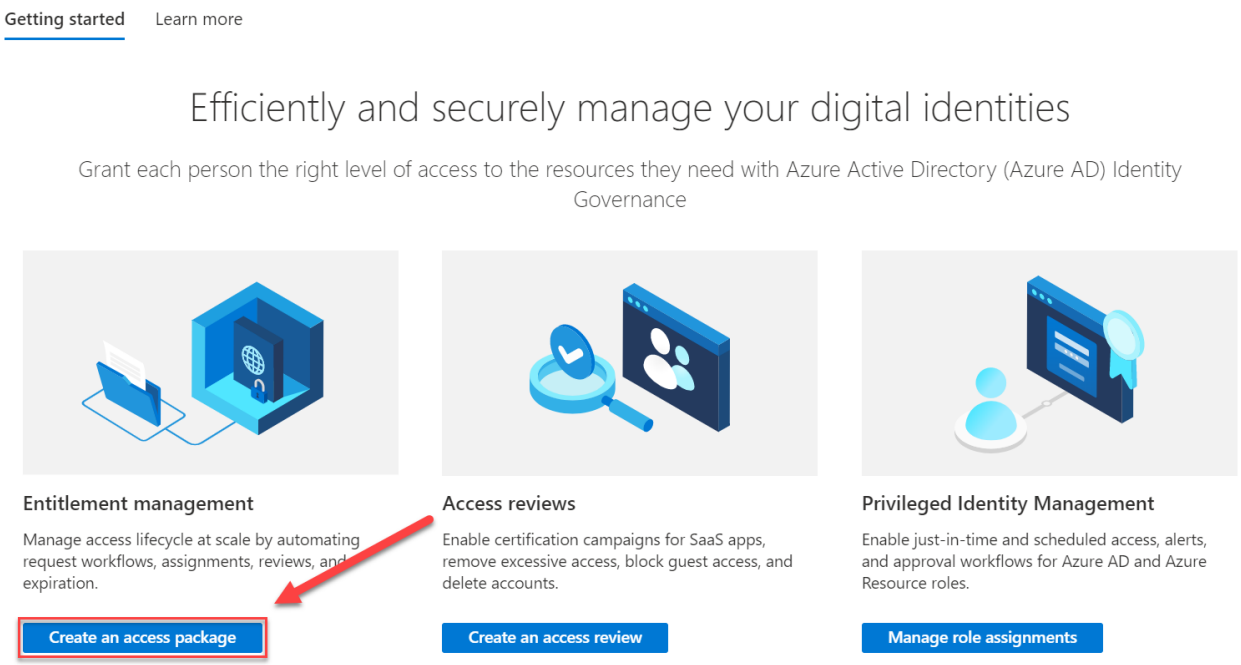

Figure 60: Identity Governance overview blade

5. The wizard is started for creating a new access package. This starts with the following basic settings:
    a. **Name:** Marketing collaboration package.
    b. **Description:** This package gives access to the different marketing collaboration tools and features.
    c. **Catalog:** General. You can create multiple catalogs to store access packages. A catalog is basically a container for the packages, and the containers can be assigned to different catalog owners. For now, keep **General** selected.

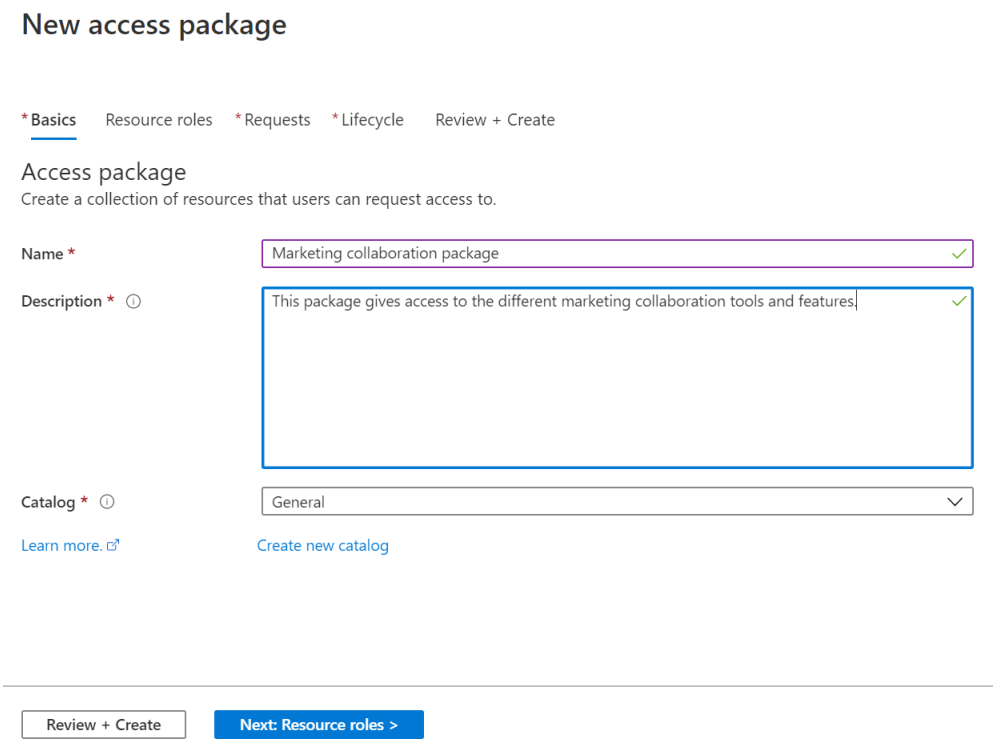

Figure 61: Basic access package settings

6. Click **Next: Resource roles**.
7. In the **Resource roles** blade, you can add the different resources you want to give the user access to. First, click **+ Groups and Teams**. Select the **GuestGroup** that we created in Chapter 2:

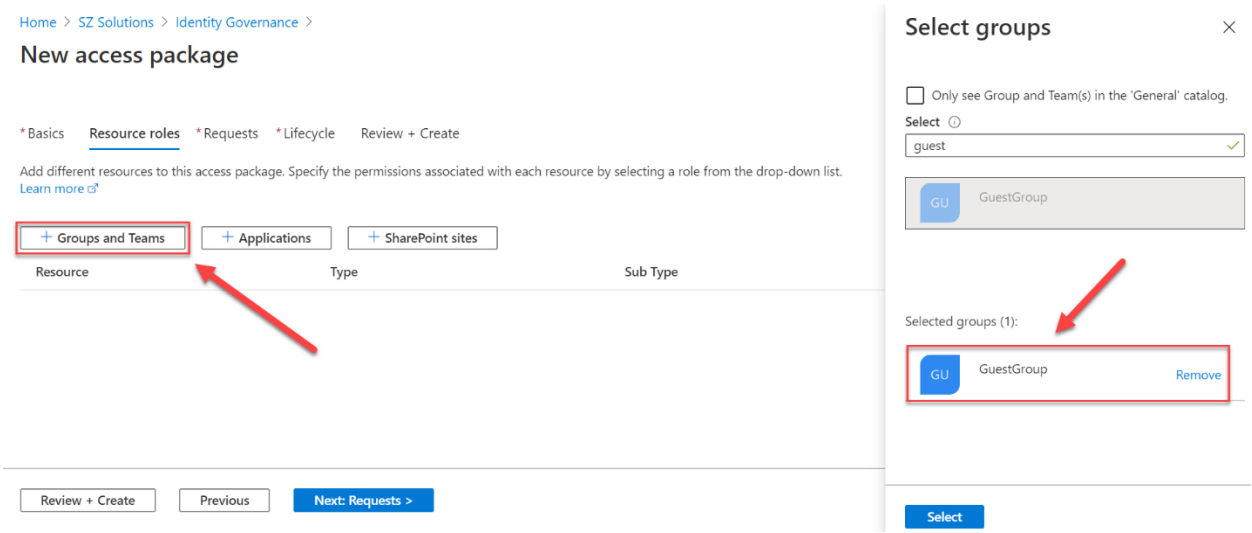

Figure 62: Add a security group

8. Click **Select**.

9. Click **+ Applications** in the top menu. There, select the same application that you used in Chapter 2:

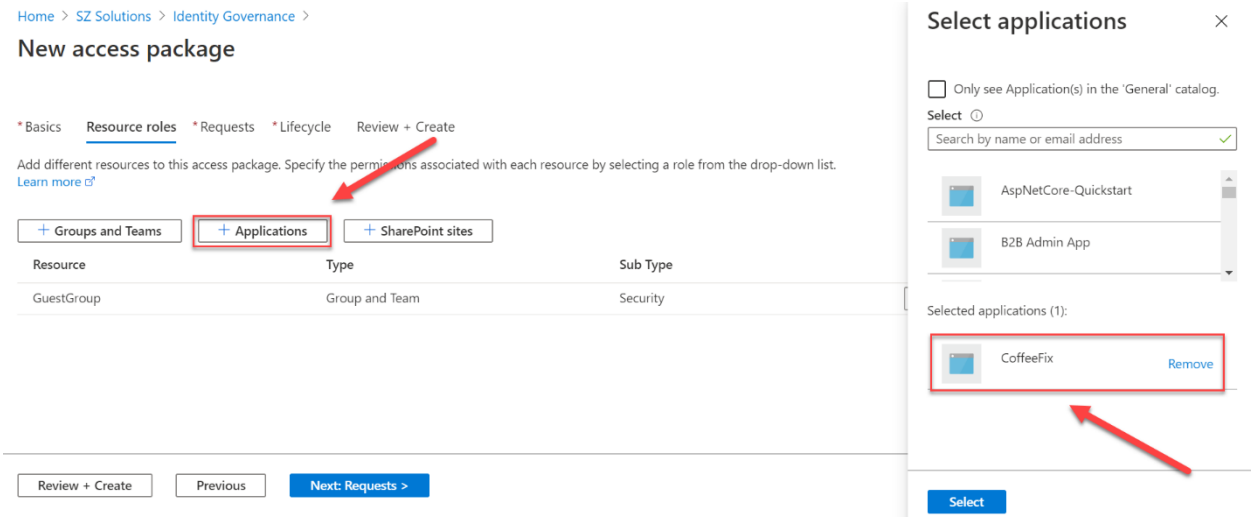

Figure 63: Add applications

10. You need to select a role for the security group and the application. For the group, select **Member**, and for the role, select **Default Access**:

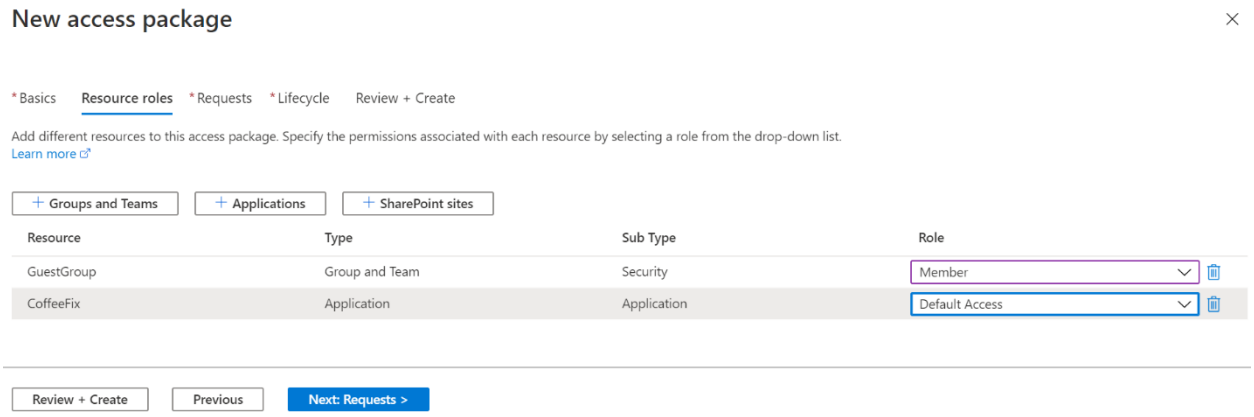

Figure 64: Set resource roles

11. Click **Next: Requests**.
12. On the Requests page of the wizard, you can create a policy to establish who can request an access package:
    a. We are going to create an access package for guest users, so we choose **For users in your directory** (this also includes guest users).
    b. Next, you can choose which type of users can access this package: specific users and groups, all members (excluding guest users), or all users. Select **All users (including guests)**.
    c. Set the **Require approval** option to **No**.
    d. Set the **Enable new requests and assignments** option to **Yes**. This is where you can enable or disable this policy for new requests.

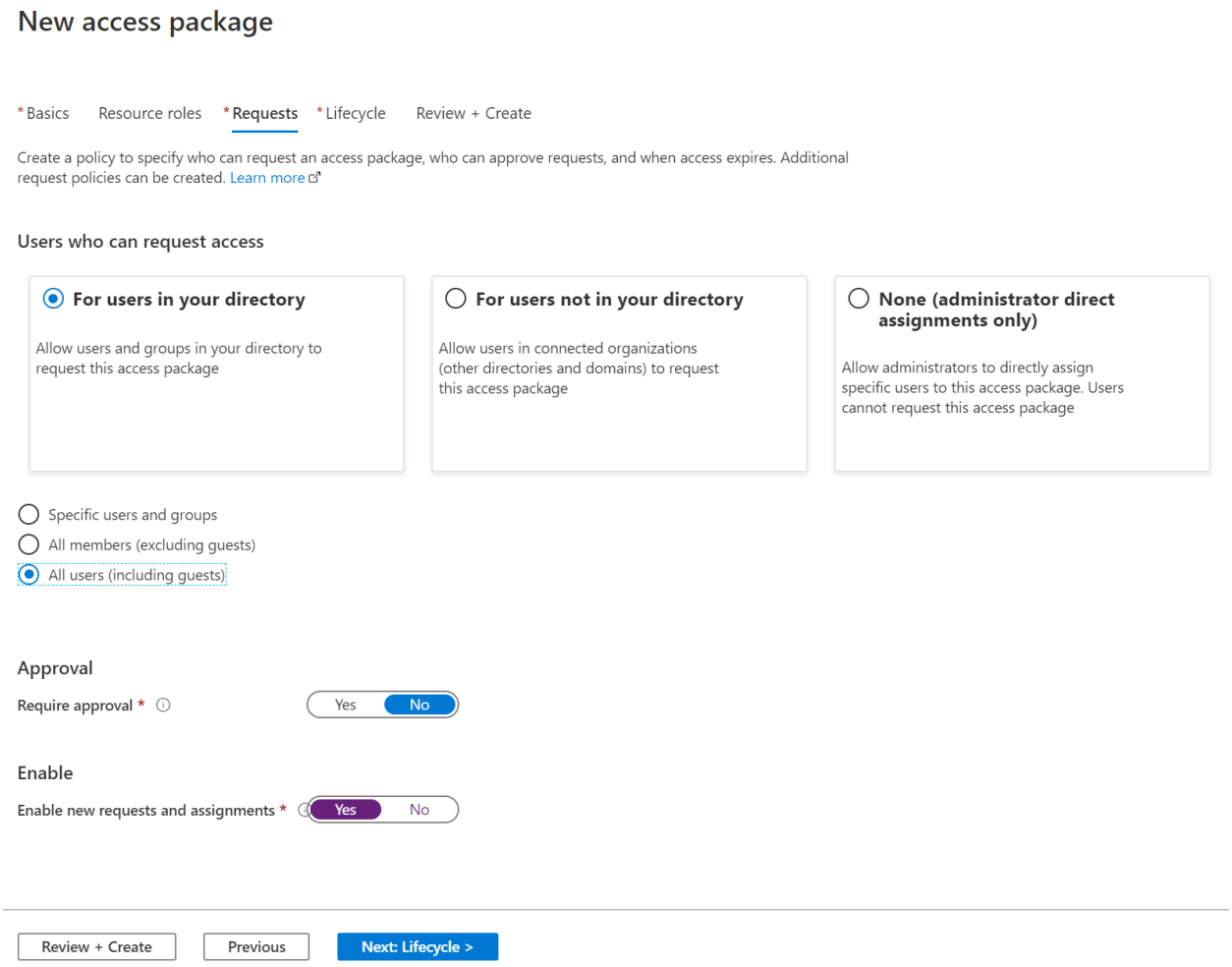

Figure 65: Create access policy

13. Click **Next: Lifecycle**. Here, you can set the expiration date for the package.
    a. You can choose a specific date, a number of days, or never. Select **Number of days**.
    b. Set the **Assignments expire after** field to **14**. This will let the access expire after two weeks.
    c. Set **Require access reviews** to **No**.

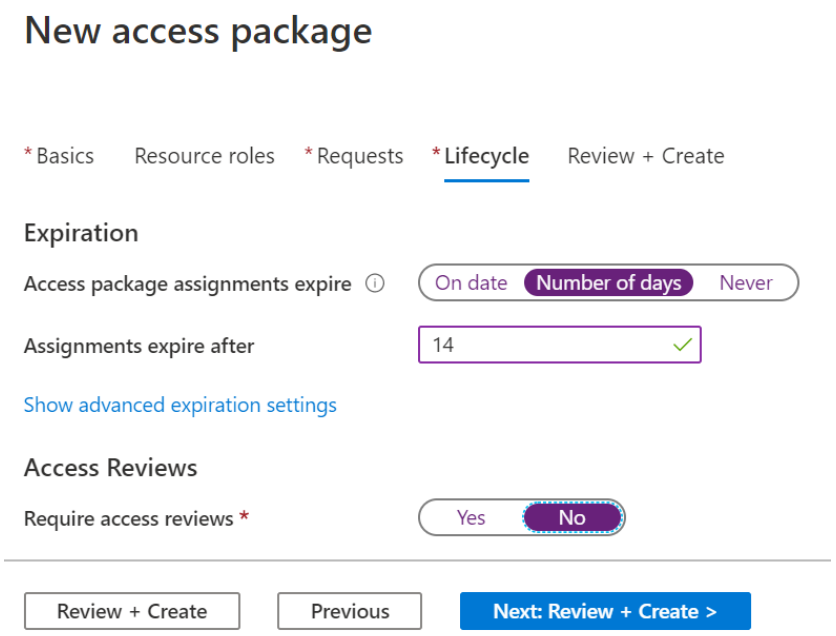

Figure 66: Set expiration

14. Click **Next: Review + Create**. You will get an overview of all the configuration values we specified:

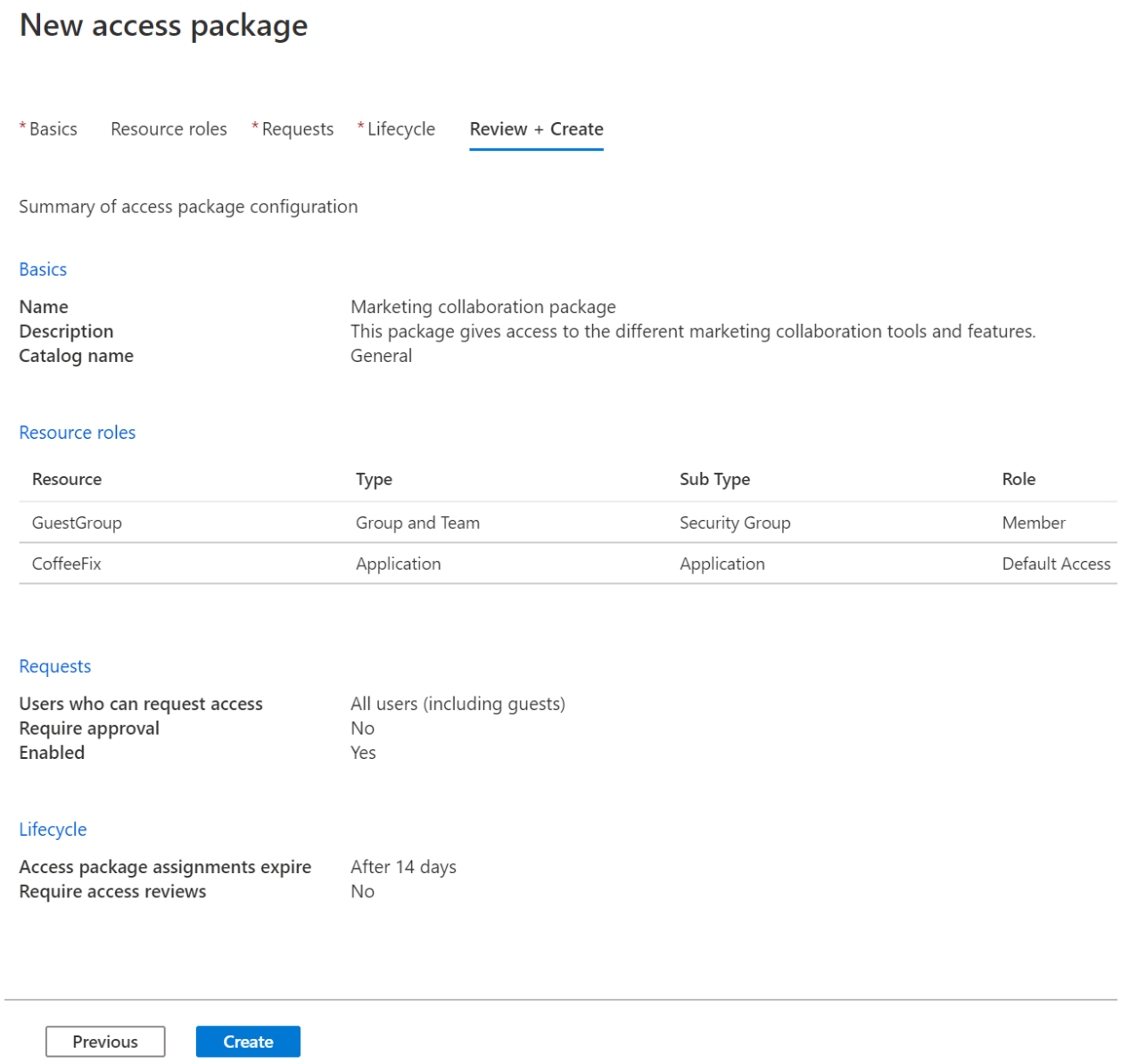

Figure 67: Review all the settings

15. Click **Create** to create the access package.

The package is now created. In the next section, we are going to cover how this access package can be redeemed by the guest user.

Redeeming the access package

To get an overview of the access packages that the guest user has access to, the user can log in to the Access overview page. It is also where the user can request access to other resources, renew access to resources, and so on.

In the next section, we are going to obtain the MyAccess link from the Azure portal. However, there are other ways to get access to this package. Users that already have access to other packages will be familiar with the https://myaccess.microsoft.com/ website. Since this package is scoped to all guest users, they can request access from there as well. In the next demo, we are going to obtain the direct link and use that link to access the package.

1. Once the package is created, it is added to the list of available access packages. You can navigate to it by selecting **Access packages** from the left menu.
2. From there, the direct URL can be shared with the user. Click the three dots on the right side of the access package that you have created, and then select **Copy MyAccess link**:

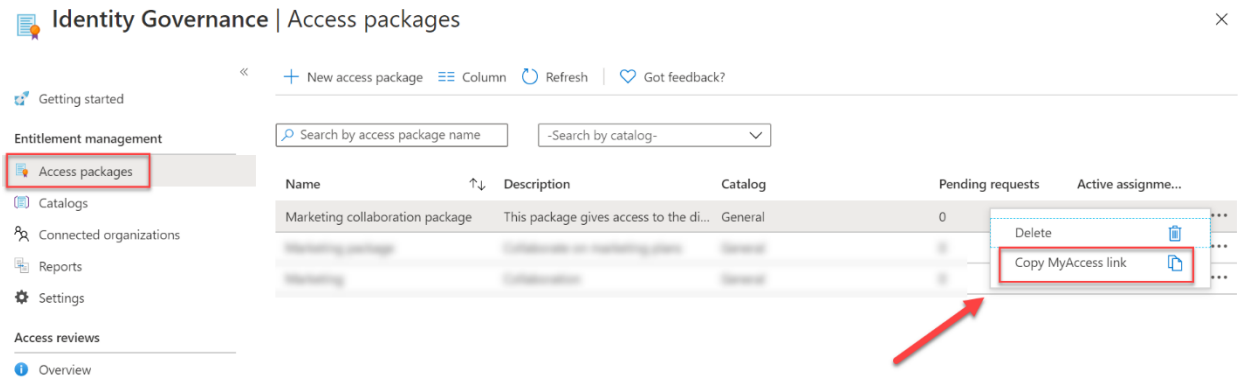

Figure 68: Retrieve package link

The direct link will be displayed:

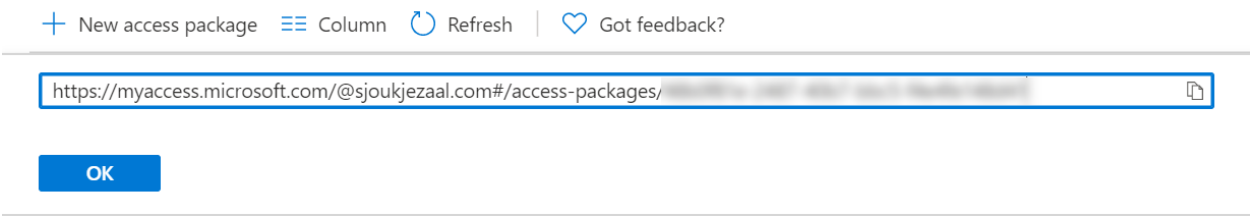

Figure 69: Copy package link

3. Copy this link and paste it in a new browser window. You will now navigate to the My Access page.
4. Log in using your guest user account credentials. The Access packages overview page will be displayed, and you will see that the package that we just created is there.

In this demo, we have retrieved the access package. This added the guest user to the required security group and gave the user access to an application. In the next section, we are going to cover how you can add connected organizations.

Adding a connected organization

If you frequently collaborate with guest users from an organization who also have an Azure AD directory or domain, you can add it as a connected organization. The users from that domain or directory can then ask for access to your organization's resources and applications. By connecting an organization, you are establishing a relationship between these directories in Azure.

For example, let's say that our organization is collaborating with an organization where users have a user principal name that ends with sjoukjezaal.com. In this case, we can create a connected organization for this collaboration. If we then add this connected organization to an access package, all users with a principal name that matches the policy can request access.

To add a connected organization, we have to take the following steps:

1. Again, navigate to the Azure portal by opening https://portal.azure.com/.
2. In the left menu, select **Azure Active Directory** > **Identity Governance**.
3. In the Identity Governance overview blade, click **Connected organizations**.
4. Click **+ Add connected organization**:

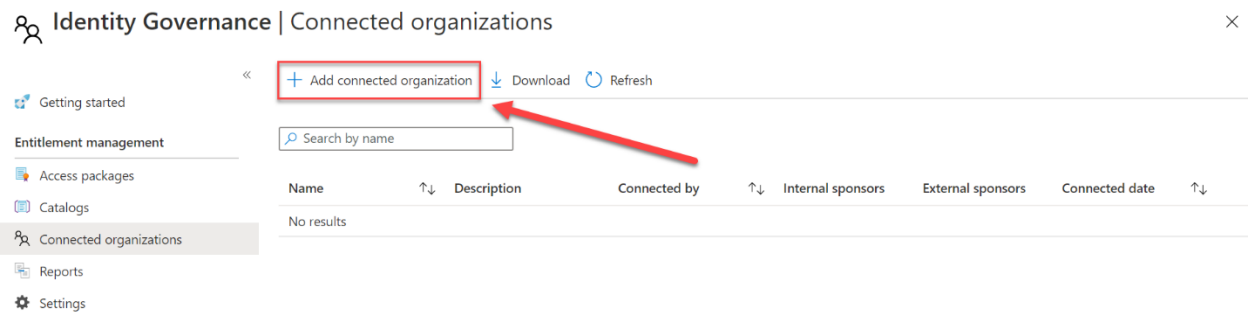

Figure 70: Add new connected organization

5. In the first page of the wizard, specify a name and description for the organization:

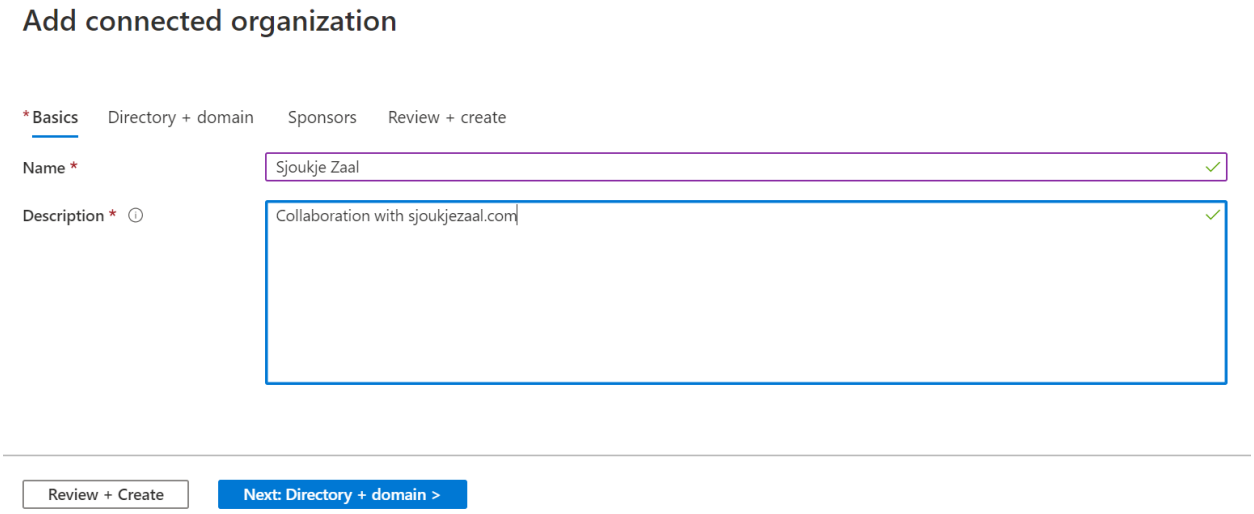

Figure 71: Specify organization name and description

6. Click **Next: Directory + domain**.
7. Click **Add directory + domain**. Fill in the domain name of the organization and click **Add**:

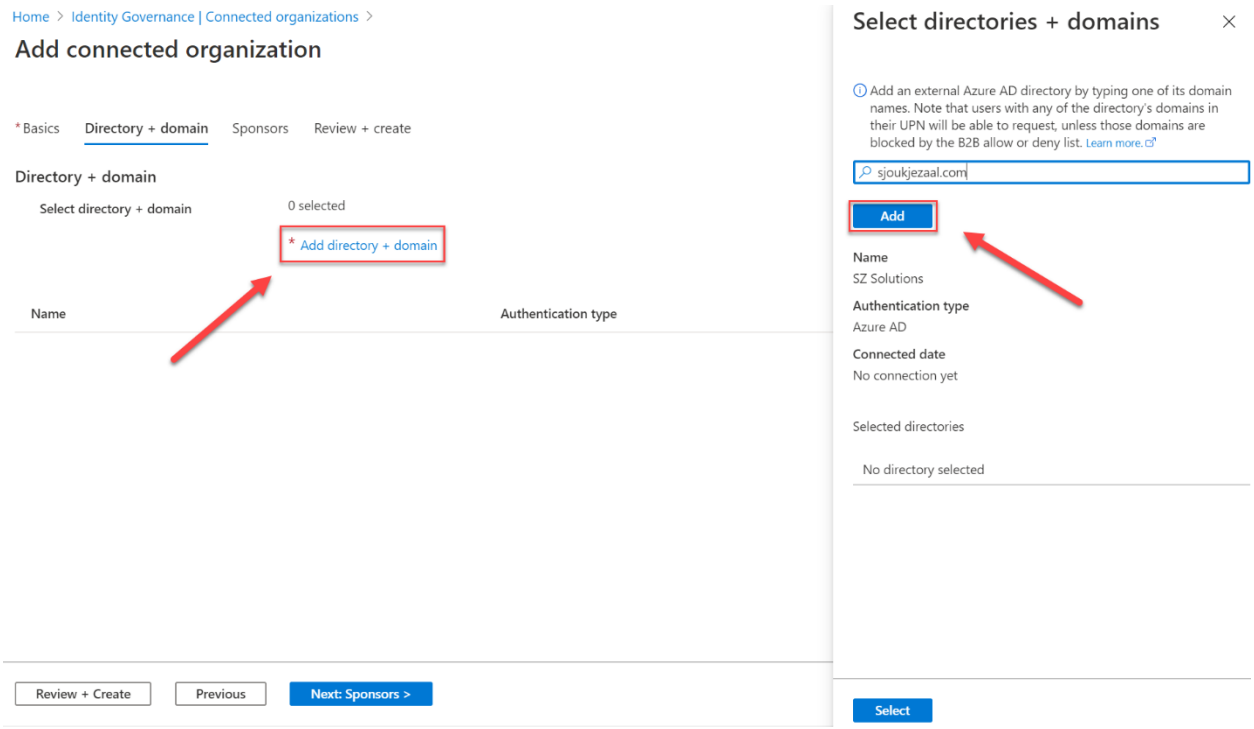

Figure 72: Add the domain name

8. Click **Select**.

Note: All users from the Azure AD directory or domain will be able to request this access package. This includes users in Azure AD from all subdomains associated with the directory, unless those domains are blocked by the Azure AD business-to-business (B2B) allow or deny list.

9. Click **Next: Sponsors**. Sponsors are internal or external users in your Azure AD tenant who are the point of contact for the relationship with the connected organization. External users already need to be added to the directory as a guest user before they can be added as a sponsor. You can select a sponsor or skip this step:

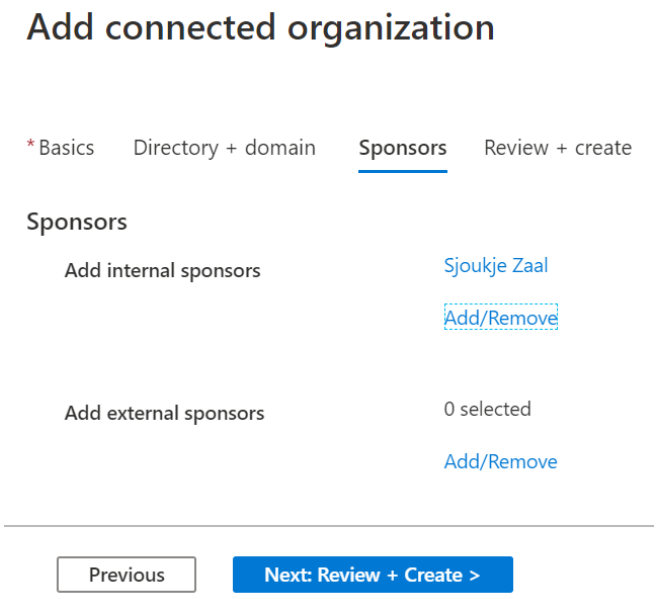

Figure 73: Adding sponsors

10. Click **Next: Review + Create**. A summary of all the settings will be displayed:

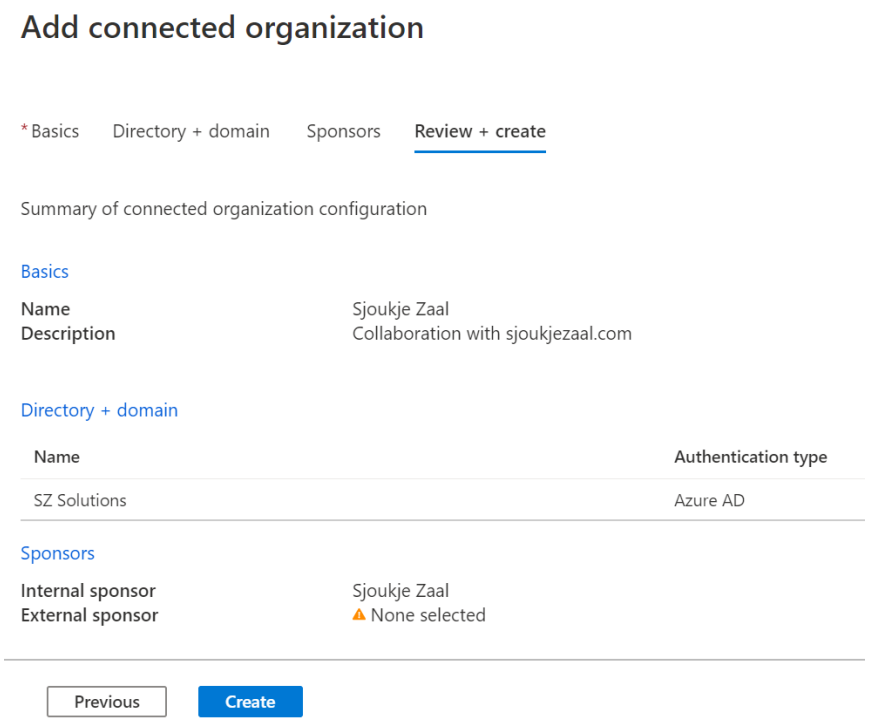

Figure 74: Connected organization summary

11. Click **Create** to create the connected organization in Azure AD.

Now that we added this domain as a connected organization, it can easily be added to all the access packages that are created.

Thus concludes this chapter. We have covered Identity Governance and entitlement management in Azure AD and Azure AD B2B. We created an access package and shared it with a guest user. In the next chapter, we are going to cover how Azure AD is integrated in Office 365.

Chapter 7 Azure AD B2B and Office 365

In this final chapter, we are going to cover how Azure AD B2B is integrated in Office 365. We are going to cover the differences between Azure AD B2B and the default sharing mechanism for SharePoint Online and OneDrive. And last, we are going to create a guest user from a PowerApp that is hosted in SharePoint Online. This PowerApp is calling a Microsoft Flow that will call the function app that we created in Chapter 5 to create the guest user in Azure.

Azure AD B2B integration in Office 365

In Office 365, users are authenticated using Azure AD B2B as well. However, there is a difference for SharePoint Online and OneDrive. These two services have a separate invitation manager by default. Support for external sharing was added to SharePoint Online and OneDrive before Azure AD B2B even existed. Nowadays, millions of users are already using the default sharing mechanism in SharePoint Online and OneDrive.

There are some differences between Azure AD B2B and the default sharing mechanism in SharePoint Online and OneDrive:

- In OneDrive and SharePoint Online, users are added to the directory *after* they have accepted their invitation. When you use Azure AD B2B, users are added immediately after the invitation is sent to the user.
- Azure AD B2B users can be picked from the SharePoint Online and OneDrive sharing dialog boxes immediately after they are added to the directory. For SharePoint Online and OneDrive, this can only be done after the invitation is redeemed.
- The redemption experience in SharePoint Online and OneDrive is different than their default sharing experiences. This means that if organizations are using more Office 365 services for guest users as well, the guest user will have different experiences across all the different products.
- Licensing works differently in the invitation managers. The SharePoint Online and OneDrive invitation managers don't have the 1:5 ratio limit that counts for Azure AD B2B. You can add unlimited guest users to SharePoint Online and OneDrive.

To enable Azure AD B2B sharing for your guest users in SharePoint Online and OneDrive, you need to change the setting in the SharePoint admin center. You need to set the external sharing value to **Existing guests**:

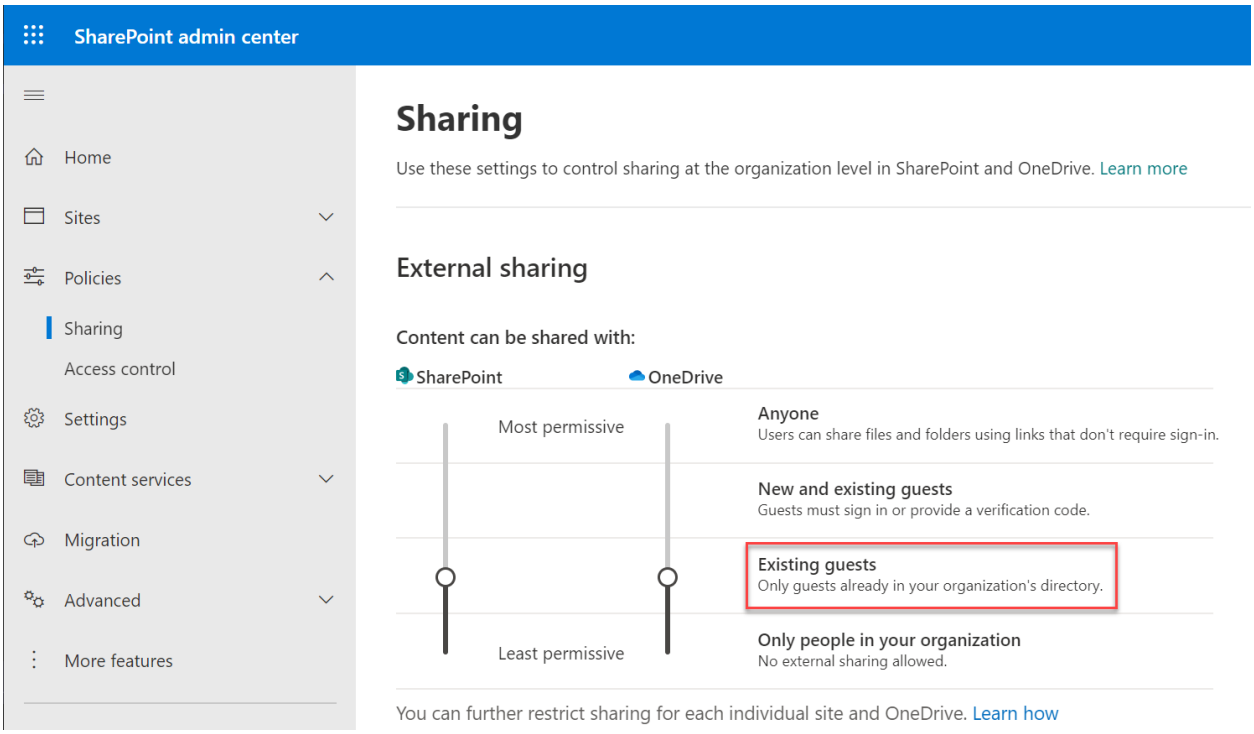

Figure 75: Enable Azure AD B2B in SharePoint Online and OneDrive

In the next section, we are going to create a sample application in PowerApps to add guest users in SharePoint Online.

Creating a guest user from a PowerApp in SharePoint Online

In this demonstration, we are going to create a guest user from a PowerApp in SharePoint Online. The PowerApp is calling Power Automate, which in turn is calling our Azure function that we created in Chapter 5. However, it is possible to call an API directly from PowerApps. I want to use Power Automate to make it clearer how to combine different services for creating robust integration solutions.

Note: For this demonstration, you need a valid Office 365 license to create the complete flow of adding the guest user.

We are first going to create the part that calls the Azure function in Power Automate.

Set up Power Automate

To set up the flow in Power Automate, you have to take the following steps:

1. Open Power Automate from Office 365 and select the **PowerApp** template:

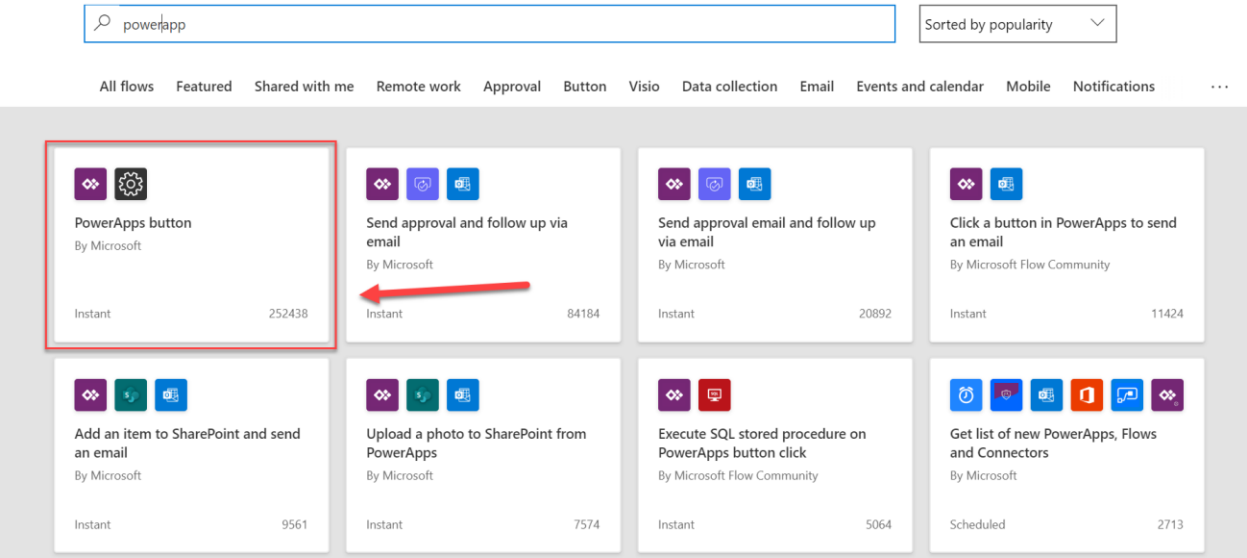

Figure 76: Create a new flow using the PowerApps button

2. Log in to the Azure portal, navigate to the Azure function that we deployed in Chapter 5, and copy the full URL to Notepad. From the Azure function resource group, select the App Service:

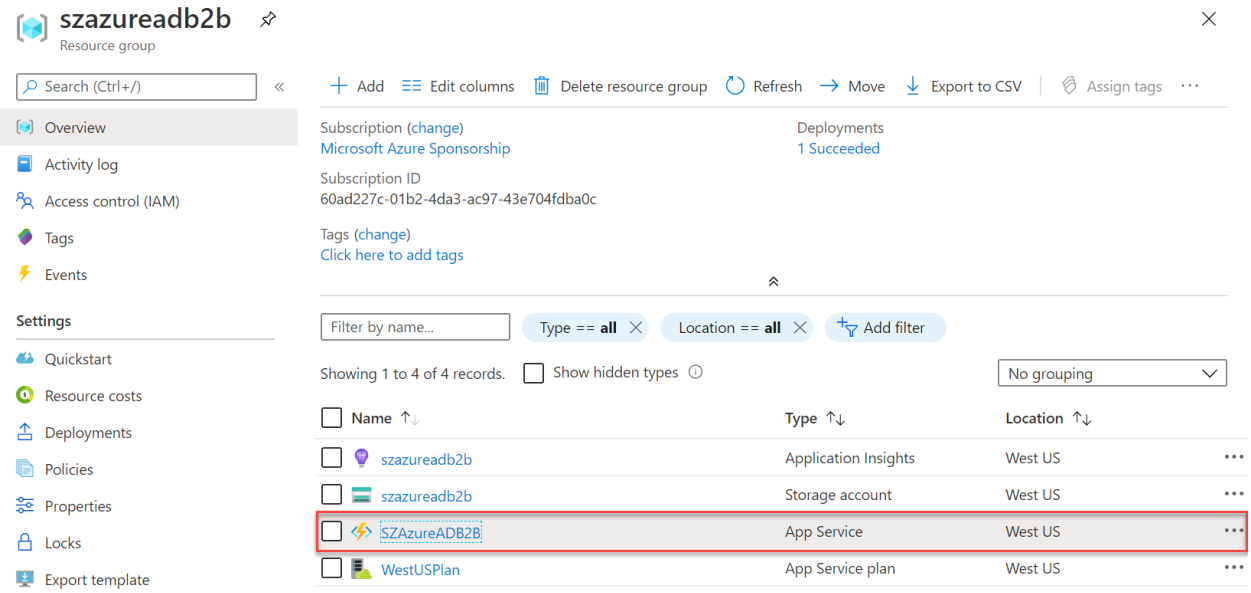

Figure 77: Navigate to the App Service

3. Then, in the App Service overview, from the top right, select the URL:

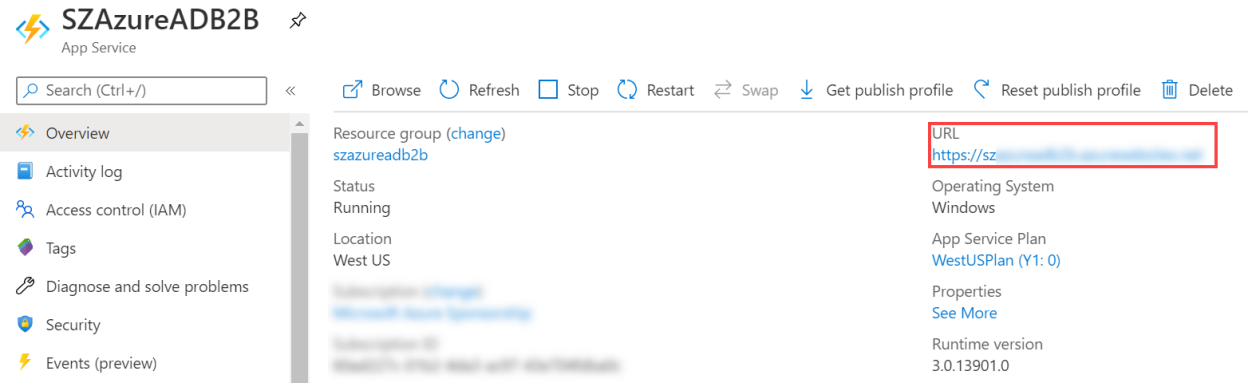

Figure 78: Copy the URL for the function

4. Click **+ New step** > **Add an action**.

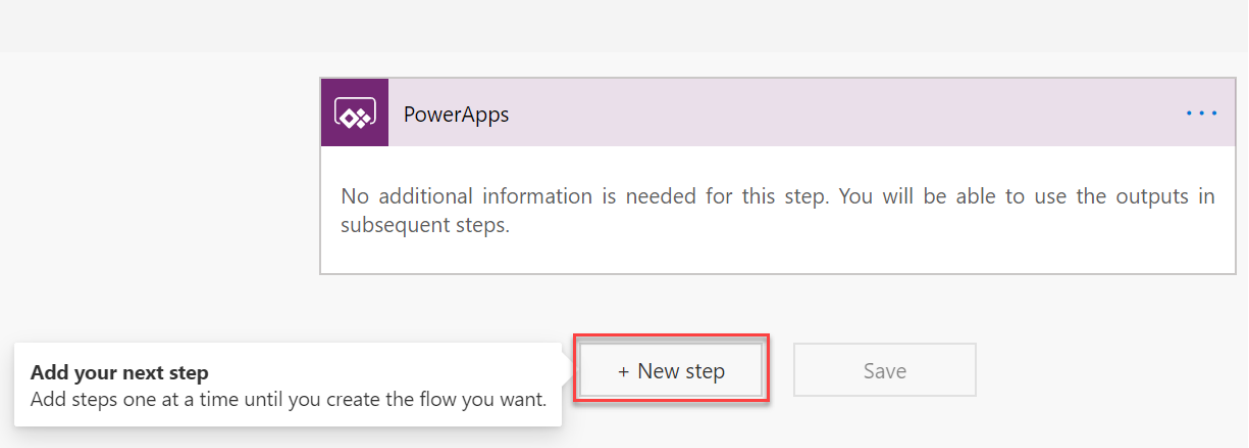

Figure 79: Add a new step to the Power Automate flow

5. To pass on the parameters from the PowerApp, add three **Initialize Variable** actions below the PowerApps action. Give them the following values:
    a. Rename the action to **FirstNameParameter**.
        i. **Name:** FirstName
        ii. **Type:** String
        iii. **Value:** Select **Ask in PowerApps**, and then a new parameter is created.
    b. Rename the action to **LastNameParameter**.
        i. **Name:** LastName
        ii. **Type:** String
        iii. **Value:** Select **Ask in PowerApps**, and then a new parameter is created.
    c. Rename the action to **EmailAddressParameter**.
        i. **Name:** EmailAddress
        ii. **Type:** String
        iii. **Value:** Select **Ask in PowerApps**, and then a new parameter is created.

    This will look like the following image:

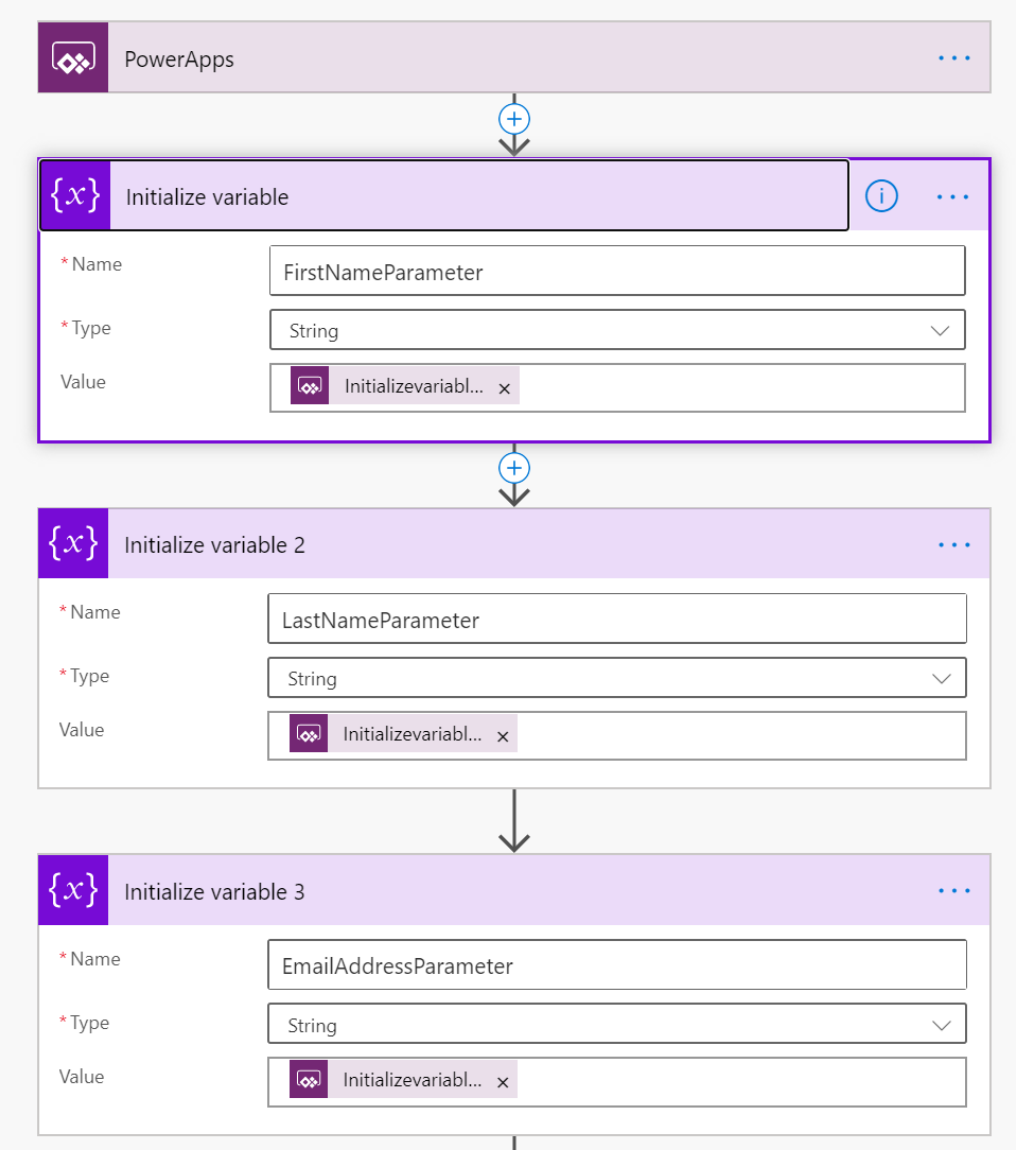

Figure 80: Create three PowerApp variables

6. Add a new action and select the **HTTP Action**. Rename the action as **Create Guest User**. Add the following values:
    a. **Method:** Post
    b. **Uri:** https://*<your-function-url>*/ HttpTriggerCSharp1
    c. **Headers:** Content-Type, application/x-www-form-urlencoded
    d. **Body:**

Code Listing 8

```
{
      "firstname": @{variables('FirstNameParameter')},
      "lastname": @{variables('LastNameParameter')},
      "emailaddress": @{variables('LastNameParameter')}
}
```

7. Click **Save** in the top right corner.

We have finished the Power Automate flow. In the next step, we can create our PowerApp for the input fields.

Create the PowerApp

In this part of the demonstration, we are going to create the PowerApp. Therefore, we have to take the following steps:

1. Go to https://web.powerapps.com and log in with your Office 365 credentials. Select the **Canvas app from blank** template, give the app a name, and select the **Phone** format:

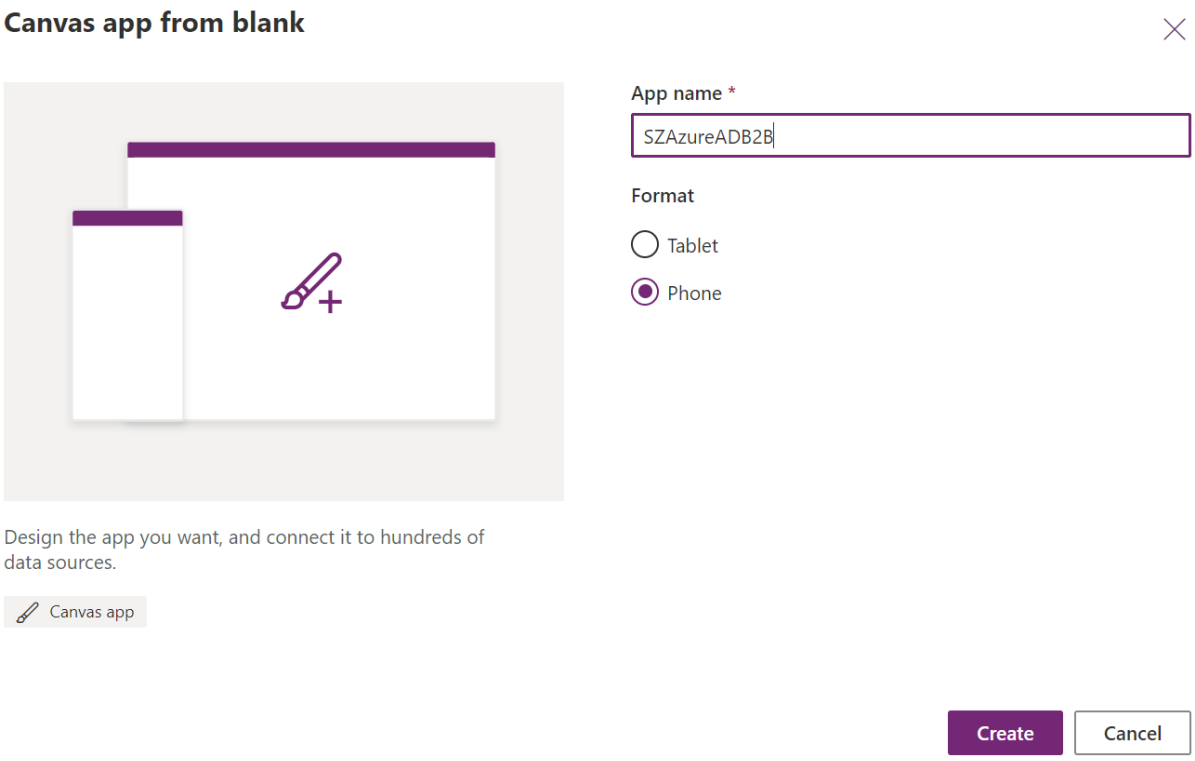

Figure 81: Create a new PowerApp

2. Click **Create**.
3. To change the screen size, go to **File** > **Settings** > **Screen size + orientation**, and set the size to **500** × **600**:

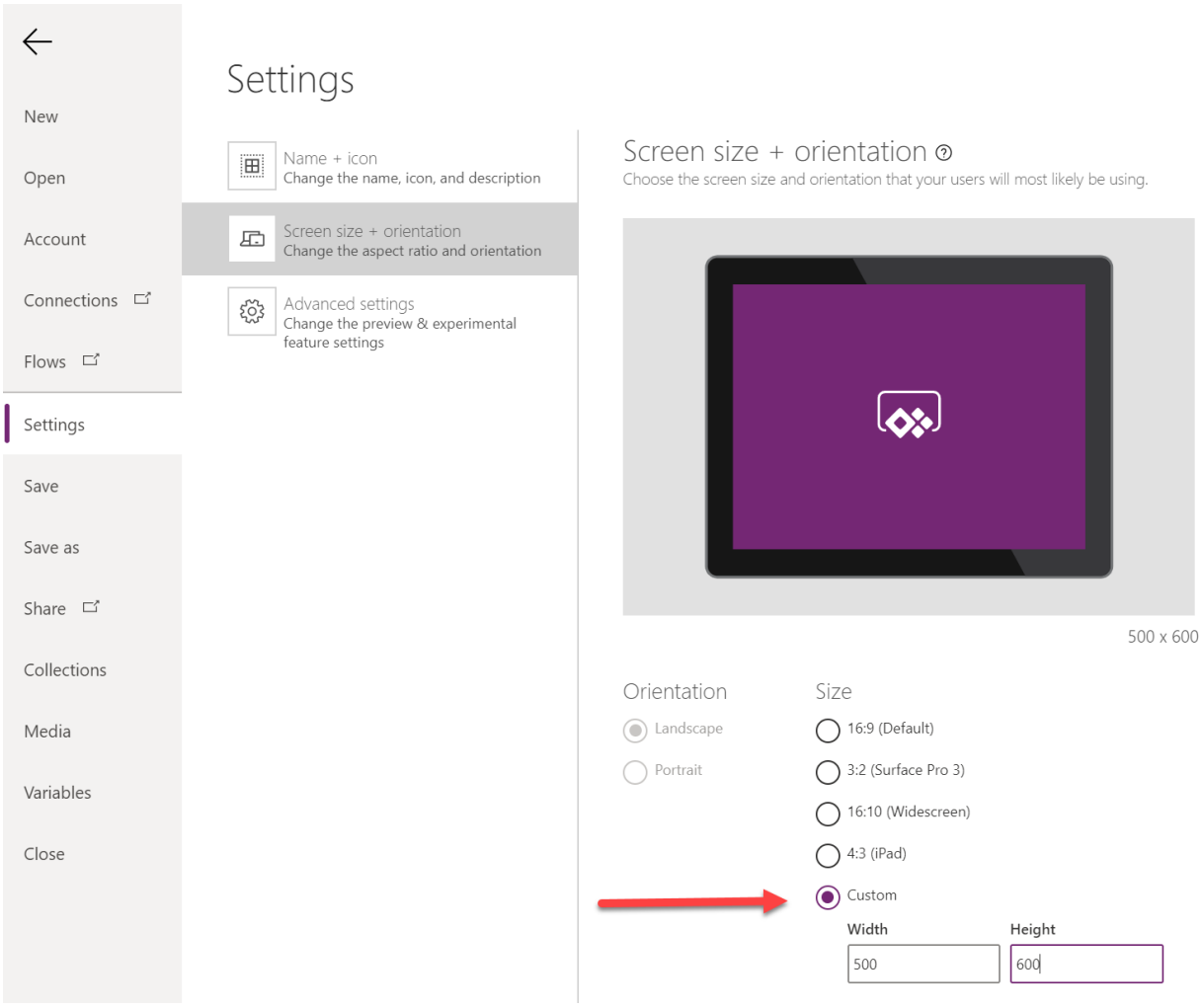

Figure 82: Adjust the screen size

4. The Edit screen is displayed. Add the following labels, fields, and button to the canvas:
    a. **Label:** Name: LabelFirstname, Text: Firstname *
    b. **TextInput:** Name: TextFirstName, clear Text Input
    c. **Label:** Name: LabelLastname, Text: Lastname *
    d. **TextInput:** Name: TextLastName, clear Text Input
    e. **Label:** Name: LabelEmailAddress, Text: Email Address *
    f. **TextInput:** Name: TextEmailAddress, clear Text Input
    g. **Button**: Name: ButtonAdd, Text: Add Guest User

    This will look like the following image:

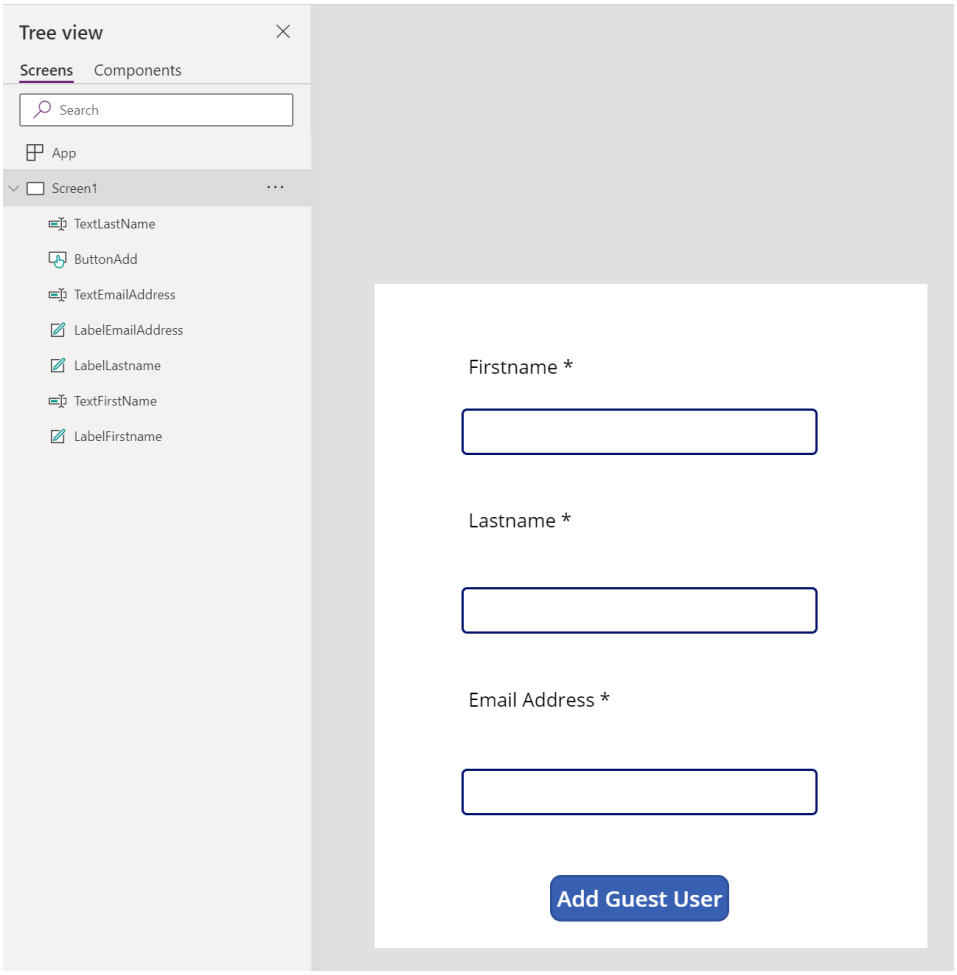

Figure 83: PowerApp

5. Click **ButtonAdd** and then click **Action**. Select the flow created in the previous section and associate it with the button:

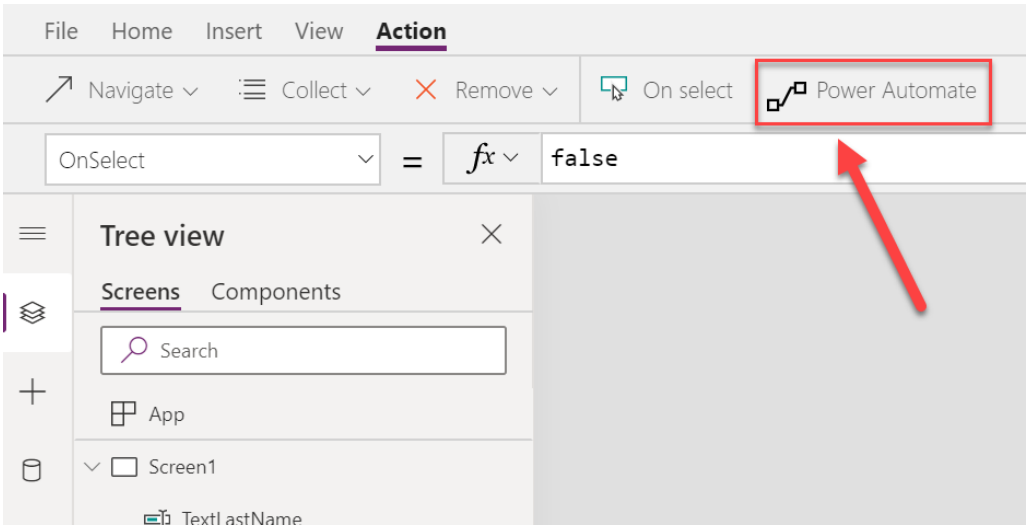

Figure 84: Associate the button with Power Automate

6. Select the **Power Automate** flow that we created in the previous section to associate it with the button.
7. In the formula bar, add the following variables to the **`Run()`** method to pass the values to the flow:

```
PowerAppsbutton.Run(TextFirstName.Text, TextLastName.Text,
TextEmailAddress.Text)
```

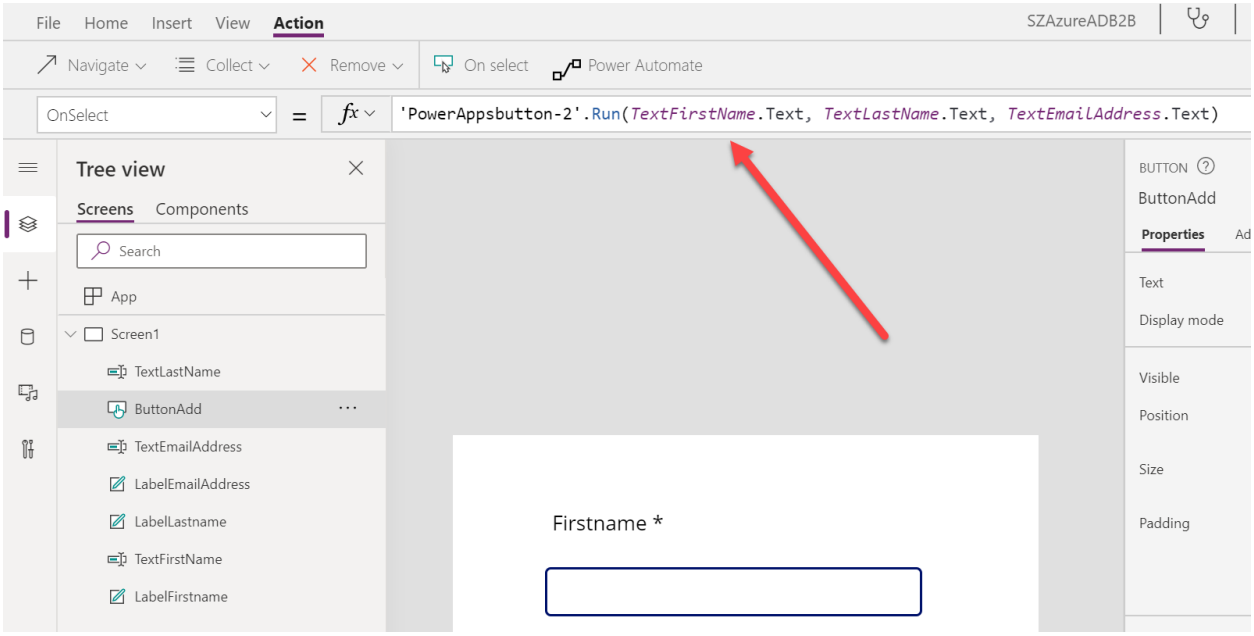

Figure 85: Add the formula for passing on the parameters

8. Save the PowerApp. Then copy the **Web link** or **App ID** to Notepad:

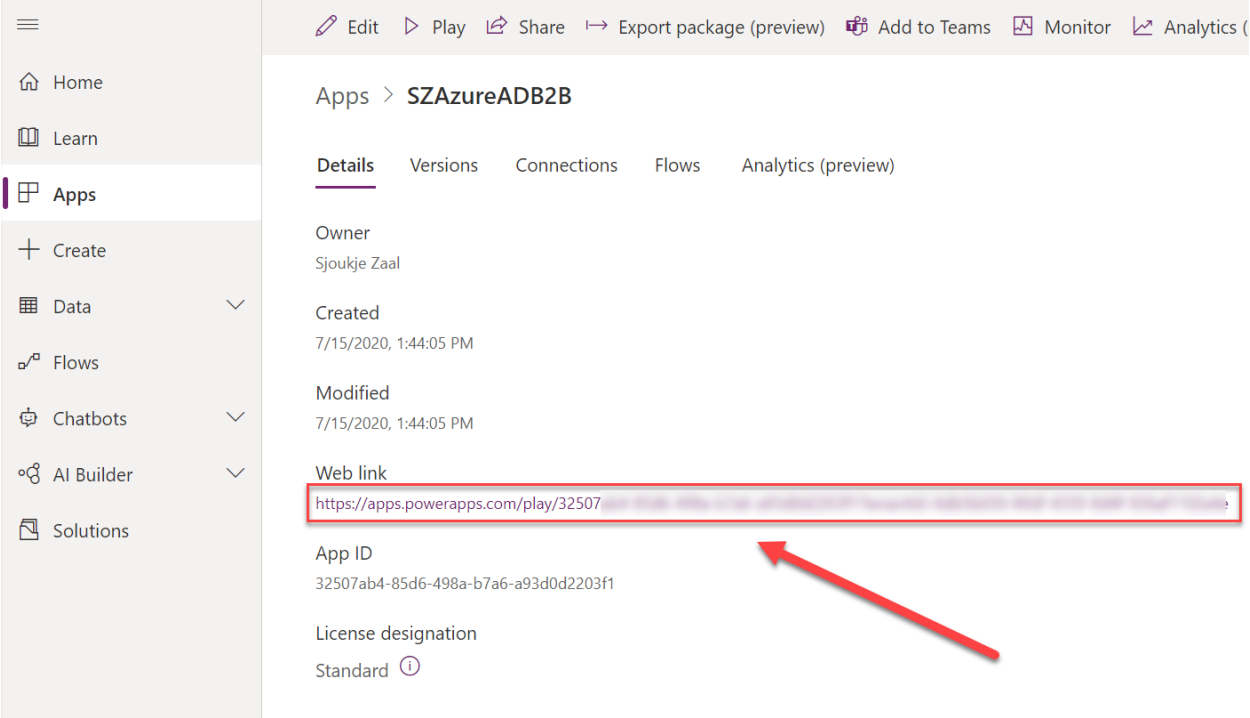

Figure 86: Copy the Web link

We have now created the PowerApp. The next and final step is to add it to a SharePoint Online site.

Adding the PowerApp to a SharePoint Site

The last step is to add the PowerApp to SharePoint so that you can start using it to invite guest users.

1. Navigate to a SharePoint Online site.
2. Add a PowerApp web part to a SharePoint page. In the settings, add the web link or the app ID, which you copied to Notepad in the previous step. Publish the page. This will look like the following image:

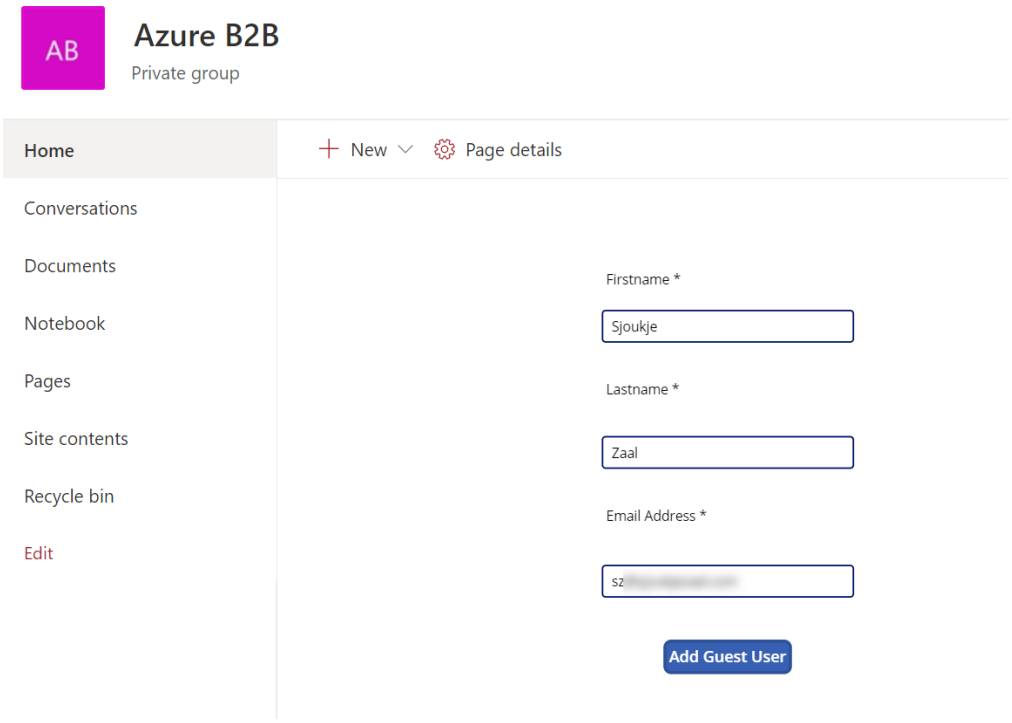

Figure 87: Add the web part to a page

3. Fill in the first name, last name, and email address. This will create the guest user in Azure AD.
4. Navigate back to Azure AD and go to **Users** in the left menu. You will see that the guest user is created.

In this last chapter, we have covered how Azure AD is integrated in Office 365. We created a sample application to add guest users from a SharePoint Online site to Azure AD using a PowerApp and a Power Automate flow.