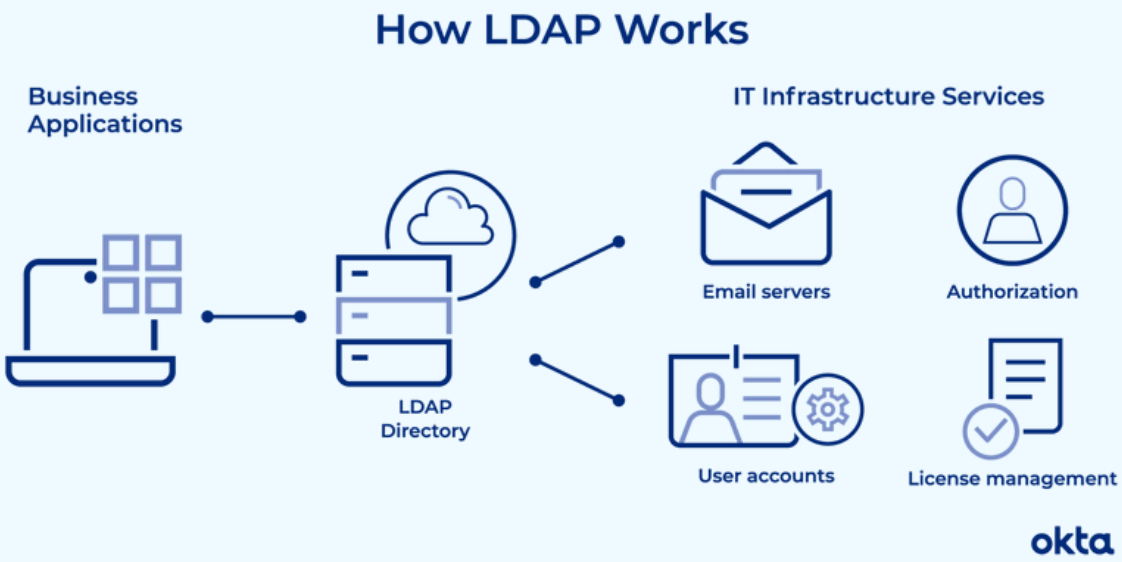
LDAP Configuration

Using ASP.net

Done by: Abdalhameed & Jehad

LDAP (Lightweight Directory Access Protocol) is a protocol used for accessing and maintaining distributed directory information services over an IP network. It is a lightweight, open, and widely used protocol designed to provide a standardized way to access directory services. **LDAP** is often used for centralized authentication, where user credentials and other information are stored in a central directory server and accessed by various applications and services.



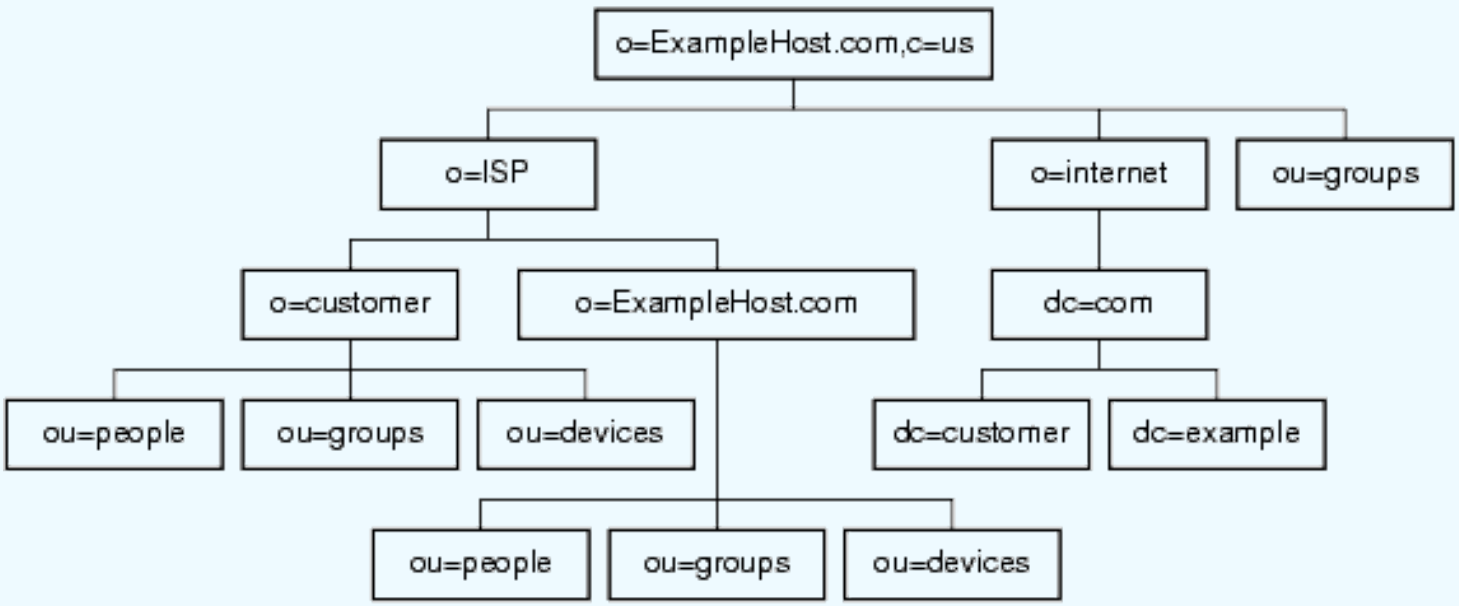
How LDAP Works?

LDAP works as a client-server protocol for accessing and managing directory information.
Here's a general overview of how LDAP works:

- Client-Server Model: **LDAP** operates on a client-server model. Clients initiate requests to the **LDAP** server, which processes these requests and returns the results.

- Directory Information Tree (DIT): **LDAP** organizes directory information in a hierarchical structure called the Directory Information Tree (DIT). The DIT starts at the root and branches out into various entries, each representing an object (such as a user, group, or device) and its attributes.

Directory Information Tree Example:



- **LDAP** Operations: LDAP defines several operations that clients can perform on the directory. These include:

1. Search: Clients can search for directory entries that match specified criteria.
2. Add: Clients can add new entries to the directory.
3. Modify: Clients can modify existing entries.
4. Delete: Clients can delete entries from the directory.
5. Bind: Clients authenticate themselves to the server using a bind operation.
6. Compare: Clients can compare an attribute value with a specified value in the directory.

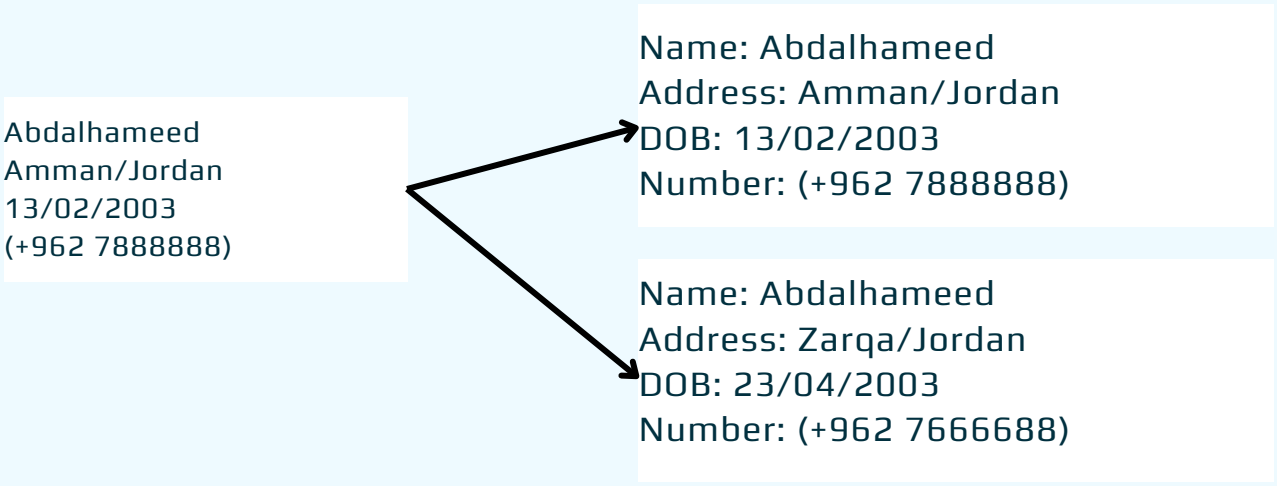
- Security: **LDAP** can be used with TLS (Transport Layer Security) to secure communication between clients and servers. This ensures that data exchanged over the network is encrypted and protected from unauthorized access.

Directory Structure

```
dn: CN=abd moh,CN=Users,DC=WIN,DC=jehad
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: abd moh
sn: moh
givenName: abd
distinguishedName: CN=abd moh,CN=Users,DC=WIN,DC=jehad
instanceType: 4
whenCreated: 20240406115118.0Z
whenChanged: 20240406115118.0Z
displayName: abd moh
uSNCreated: 32834
uSNChanged: 32839
name: abd moh
objectGUID:: iA9H0kVYJU+gn/9rktaNFQ==
userAccountControl: 66048
badPwdCount: 0
codePage: 0
countryCode: 0
badPasswordTime: 0
lastLogoff: 0
lastLogon: 0
pwdLastSet: 133568778788167556
primaryGroupID: 513
objectSid:: AQUAAAAAAAUVAAAAAeWh+1O66/6ismNyUgQAAA==
accountExpires: 9223372036854775807
logonCount: 0
sAMAccountName: abd123
sAMAccountType: 805306368
userPrincipalName: abd123@WIN.jehad
objectCategory: CN=Person,CN=Schema,CN=Configuration,DC=WIN,DC=jehad
dSCorePropagationData: 16010101000000.0Z
```

This is the directory structure of our **LDAP** server where u can see all the relevant information about the user such as the DN, Logs, Name, SAM, etc..

Directory Structure (CONT.)

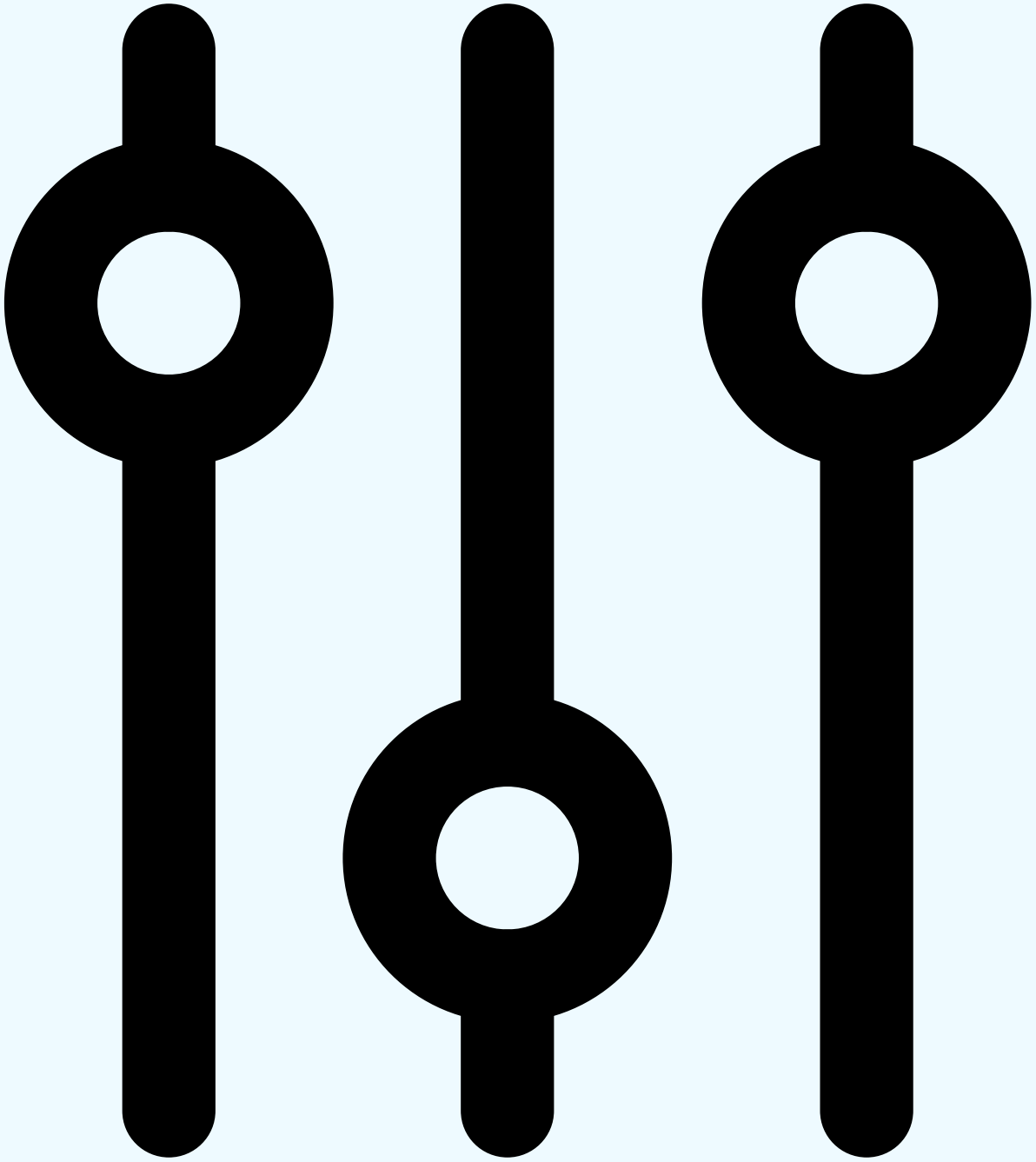


We have two users with the same name, one in Amman and one in Zarqa.
How can we distinguish between these users?

Distinguish Name(DN)

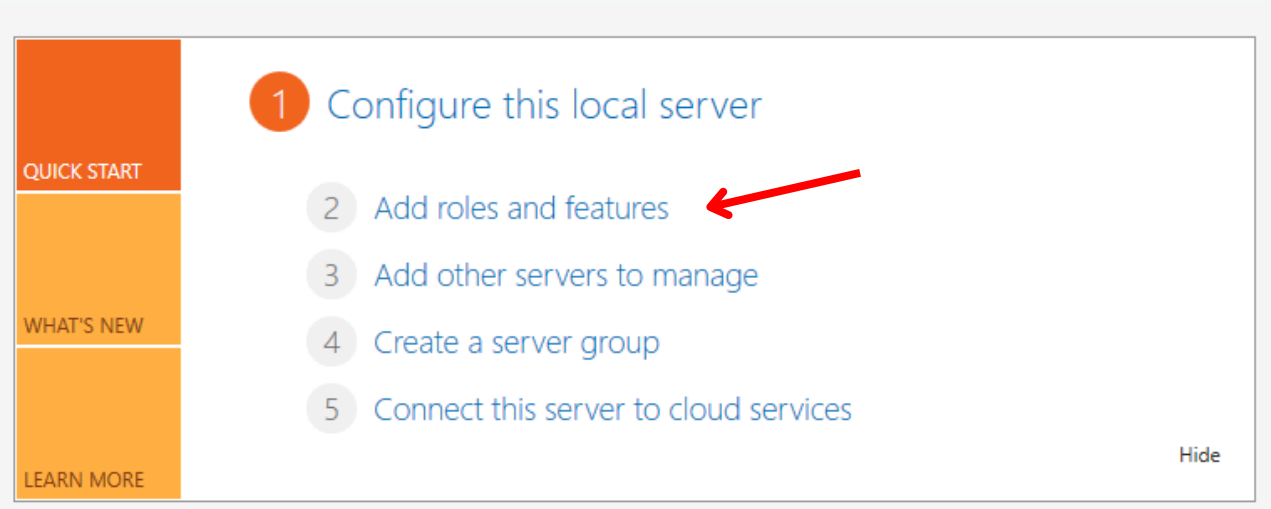
- One way of distinguishing between two very similar records is to create a unique name for each record in the directory.
- The DN is always indexed and will always be returned in any search.
- A DN is composed of a combination of directory information, and looks something like this :
  - cn=Abdalhameed,ou=Users,ou=Amman, ou=Jordan,dc=WIN,dc=jehad
  - cn=Abdalhameed,ou=Users,ou=Zarqa, ou=Jordan,dc=WIN,dc=jehad

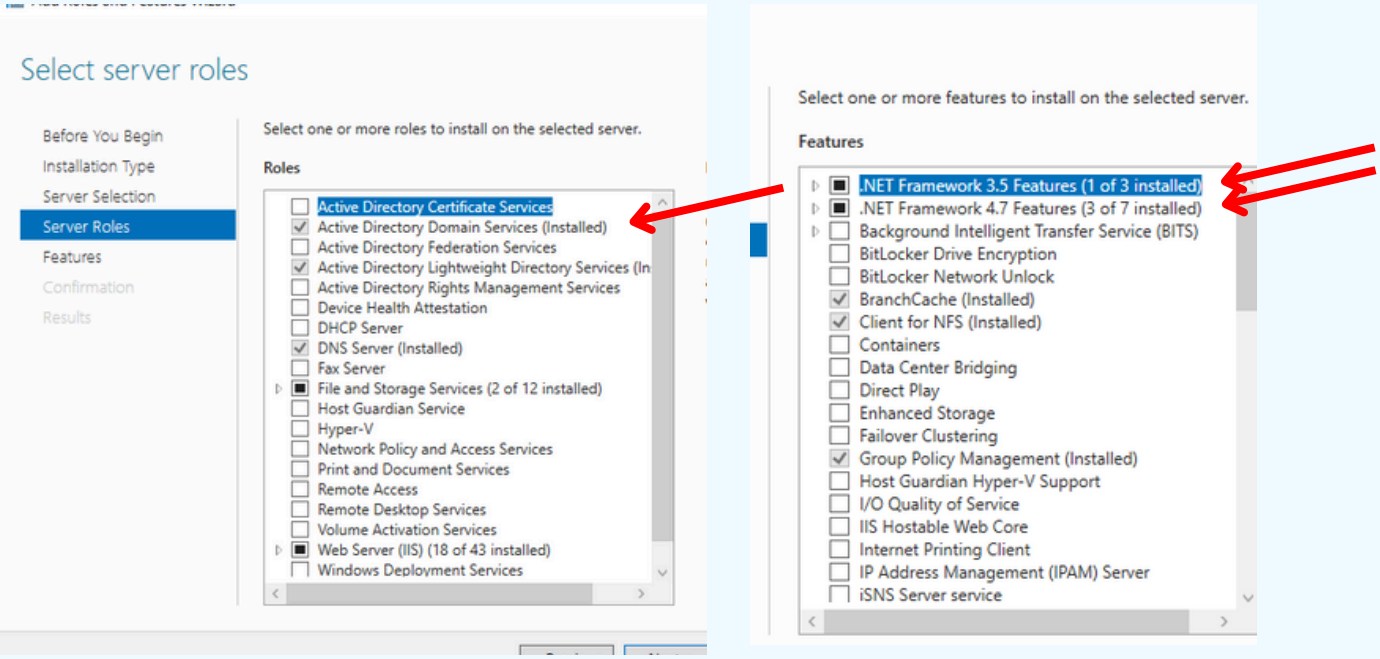
LDAP Configuration



LDAP Configuration

First step: you will need to configure your Active Directory:

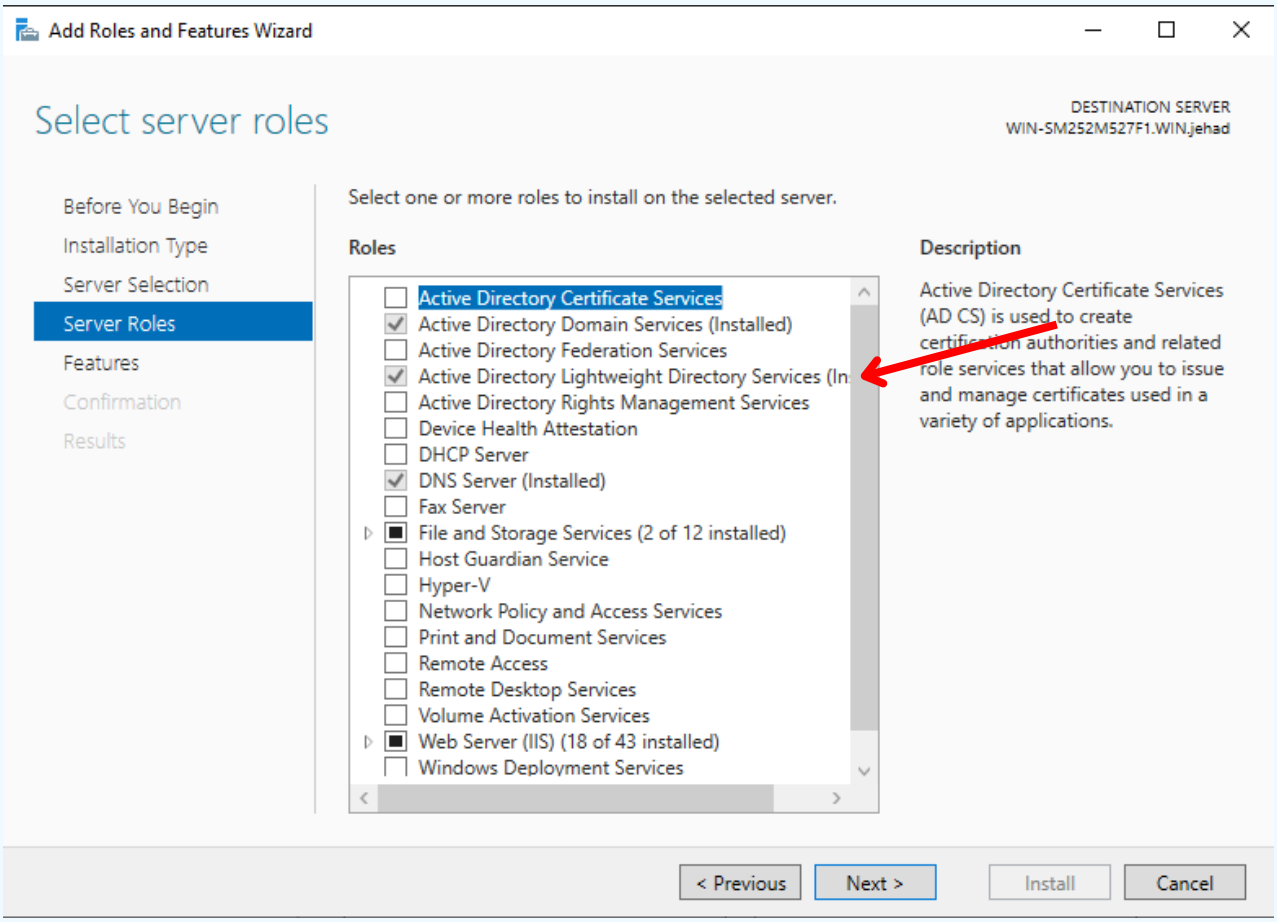




- Add Roles and Features: In Server Manager, click on "Add roles and features" from the Dashboard or the Manage menu.
- Server Roles: Scroll down and select "Active Directory Domain Services". This will open a popup window asking you to add features that are required for Active Directory Domain Services.
- Features: Check the boxes for ".NET Framework 3.5 Features" and ".NET Framework 4.7 Features" if they are not already checked.

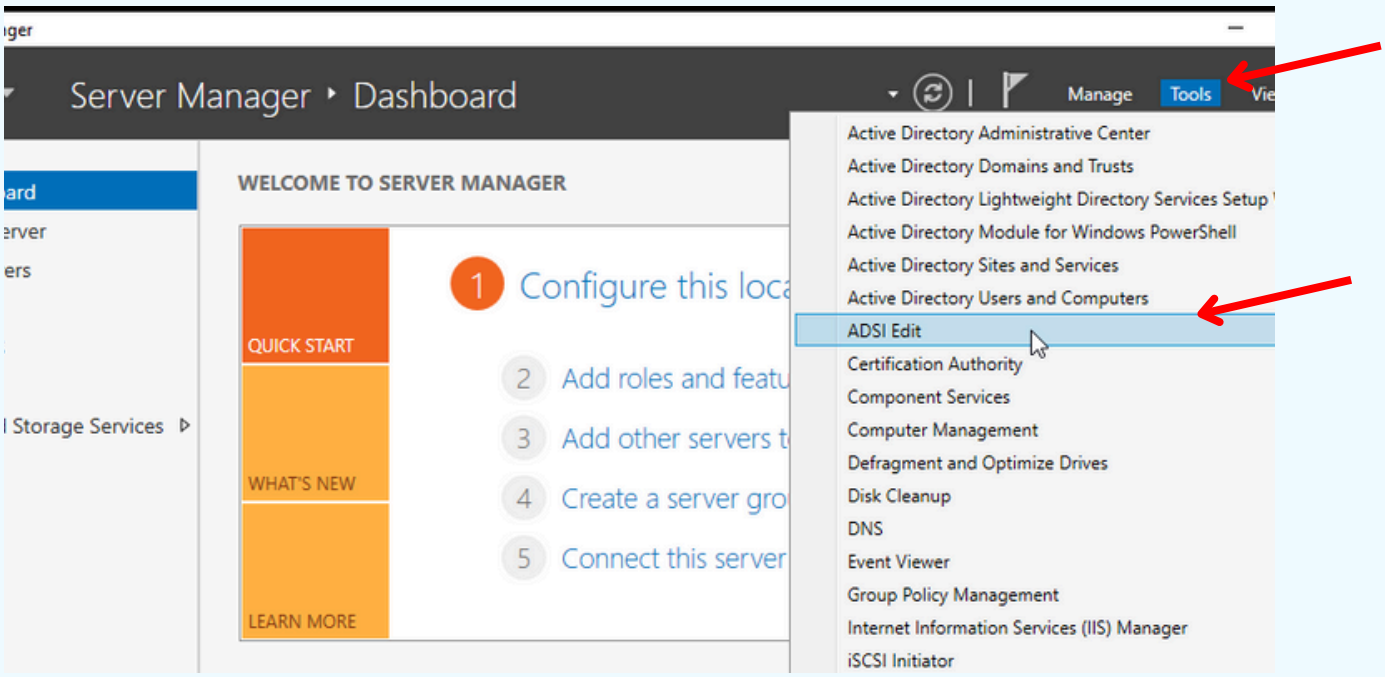
AD DS Configuration: Click Next through the AD DS configuration steps. You can leave the default settings unless you have specific requirements.

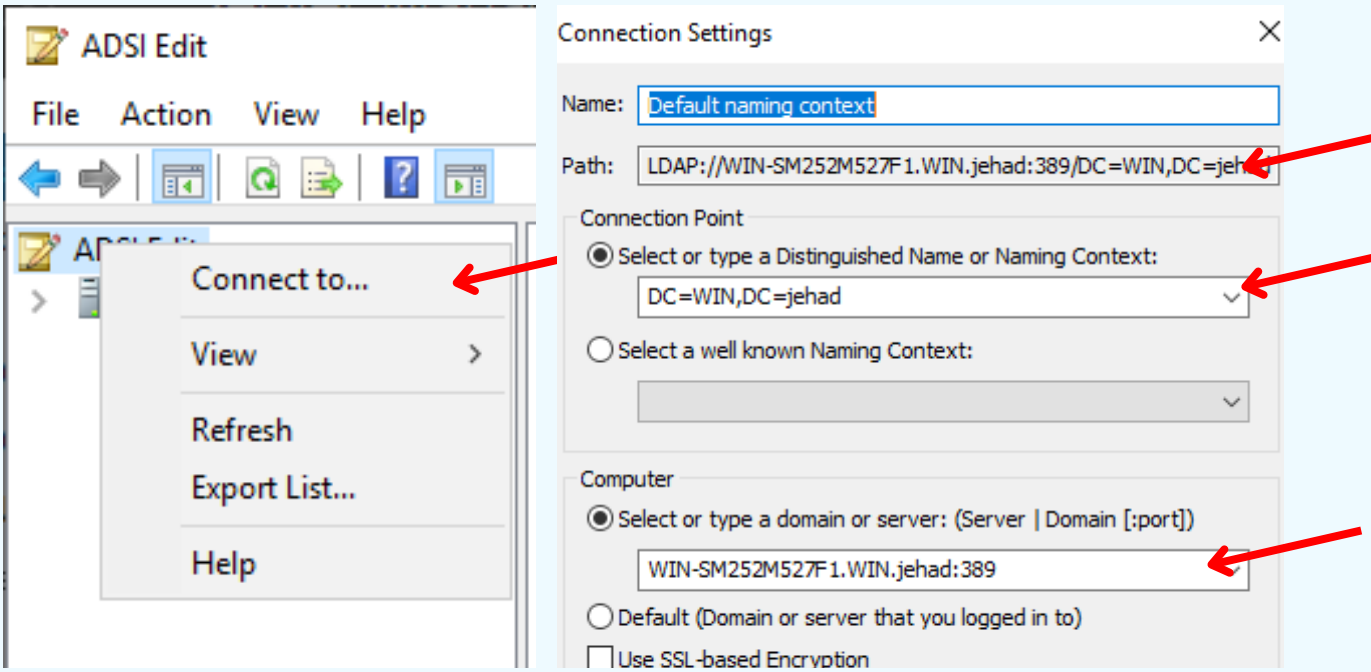
Second step: You will need to download the Active Directory Lightweight Directory Services (AD LDS).



- Add Roles and Features: In Server Manager, click on "Add roles and features" from the Dashboard or the Manage menu.
- Server Roles: Scroll down and select "Active Directory Lightweight Directory Services".
- Features: Ensure that the NET framework is installed
- After installation proceed to open ADSI Edit

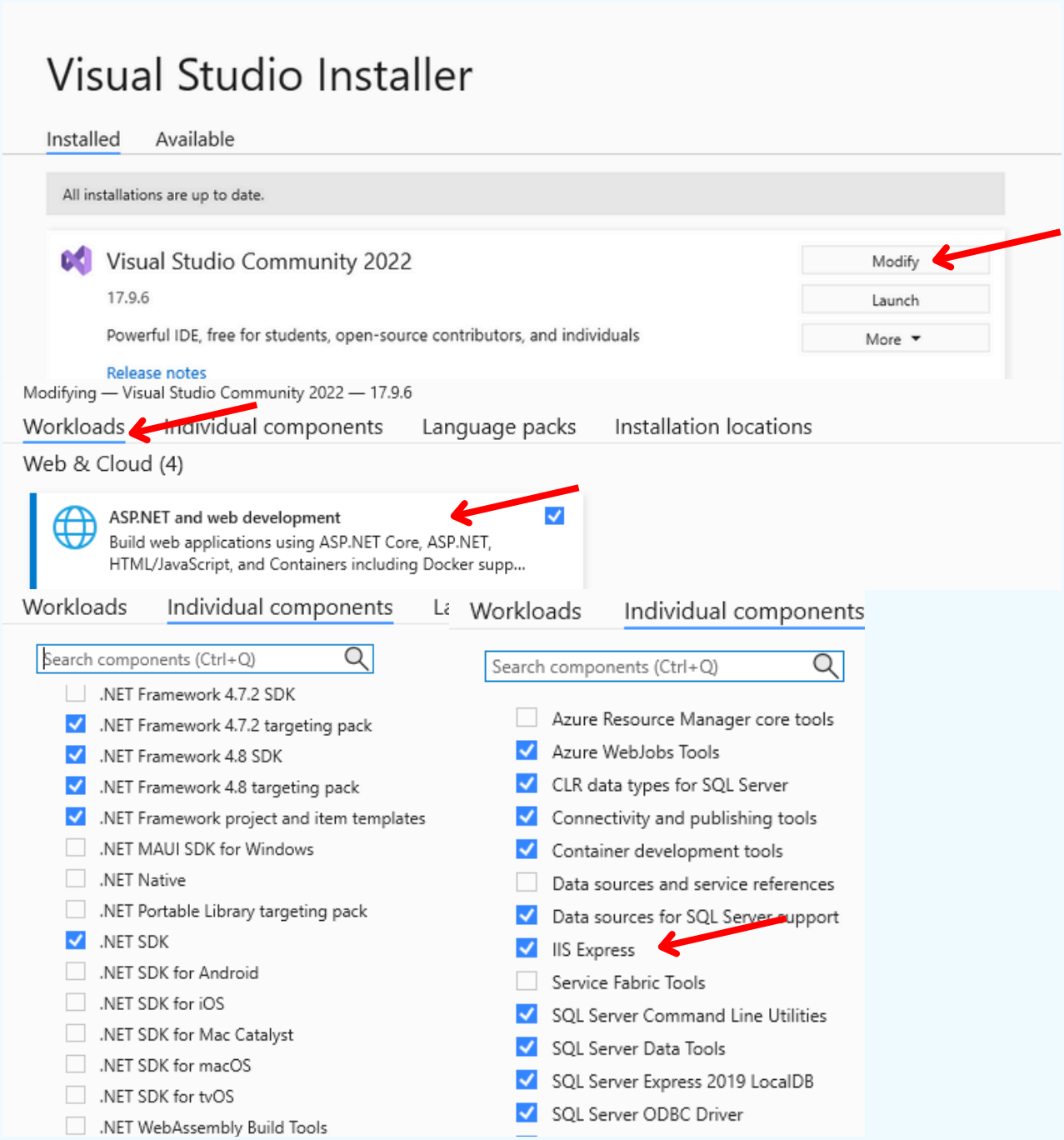
Third Step: Configure AD LDS





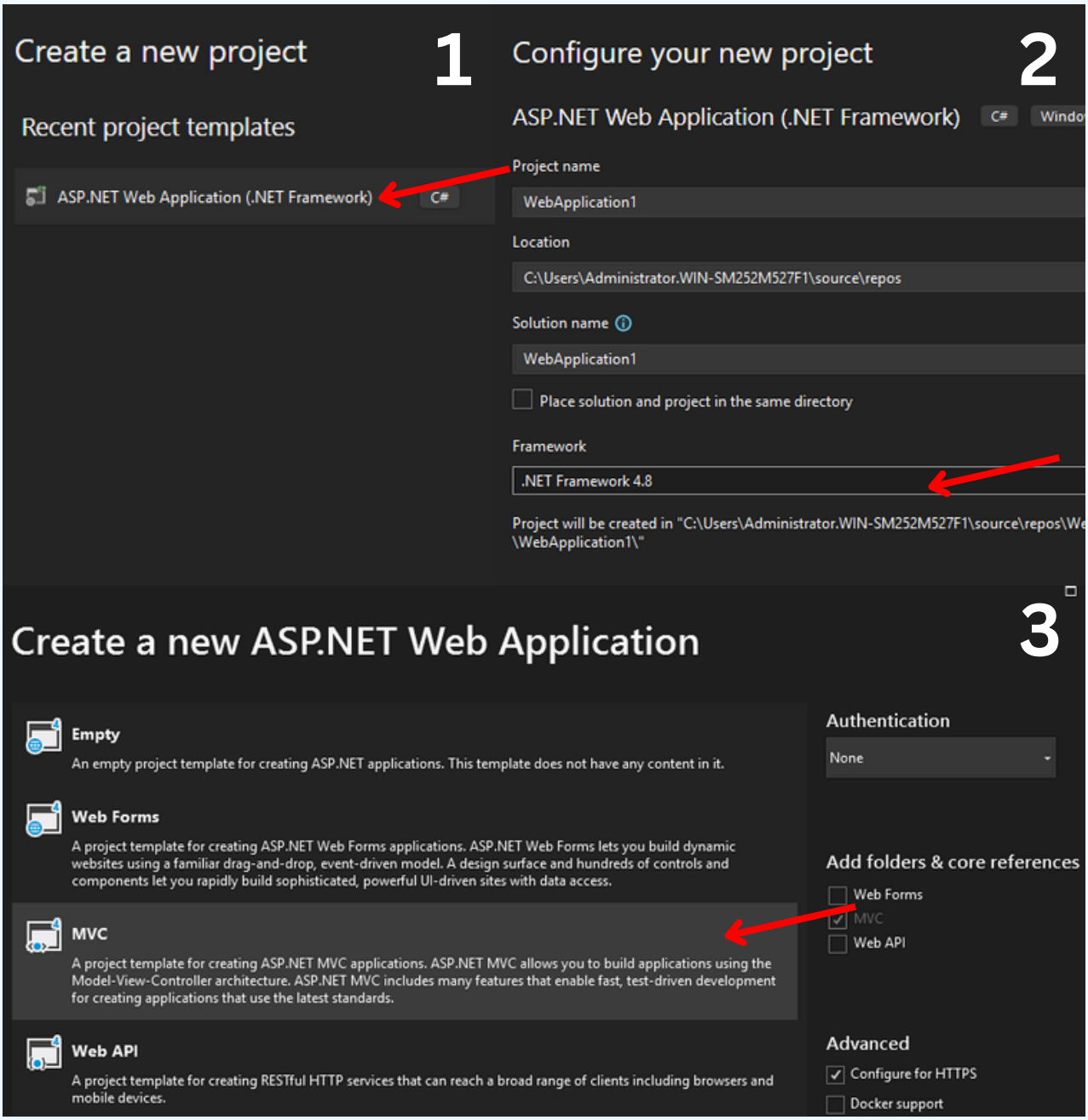
- After installing AD LDS, open the ADSI Edit tool from the Tools menu in Server Manager.
- Connect to the AD LDS instance by specifying the server name and the AD LDS instance name (e.g., "localhost:389" for the default instance).

Fourth Step: Install .NET Framework on Visual Studio



- Open Visual Studio installer and click Modify.
- Choose ASP.NET from Workloads and choose the latest version of .NET Framework SDK.
- Choose IIS Express if it is not installed.

Fifth Step: Create an ASP.NET Web Application project



- Open Visual Studio and choose new project.
- Choose ASP.NET Web Application.
- Select the "Web Application (Model-View-Controller)" template. This will set up your project with the MVC pattern. Click "Create".

The Main LDAP Code

Explanation is in the pictures

```
using System;
using System.Collections.Generic;
using System.DirectoryServices.AccountManagement;
using System.EnterpriseServices.Internal;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;

namespace ldap.Controllers
{
    public class HomeController : Controller
    {
        // GET: Home/Index
        public ActionResult Index()
        {
            // Set the message for the view
            ViewBag.Message = "Your contact page.";

            // Return the Index view
            return View();
        }

        // GET: Home/Panel
        public ActionResult Panel()
        {
            // Set the message for the view
            ViewBag.Message = "Your contact page.";

            // Return the Panel view
            return View();
        }
```

```
// Properties for username and password (used |for data binding)
[Required]
[Display(Name = "Username")]
public string Username { get; set; }

[Required]
[DataType(DataType.Password)]
[Display(Name = "Password")]
public string Password { get; set; }

// POST: Home/Index
[HttpPost]
public ActionResult Index(string username, string password)
{
    // Validate the user's credentials
    if (ValidateCredentials(username, password))
    {
        // If the credentials are valid, set up the forms authentication ticket
        FormsAuthentication.SetAuthCookie(username, false);

        // Redirect the user to the Panel action of the Home controller
        return RedirectToAction("Panel", "Home");
    }
    else
    {
        // If the credentials are not valid, add a model error and redirect to the Index a
        ModelState.AddModelError("", "The user name or password provided is incorrect.");
        return RedirectToAction("Index", "Home");
    }
}
```

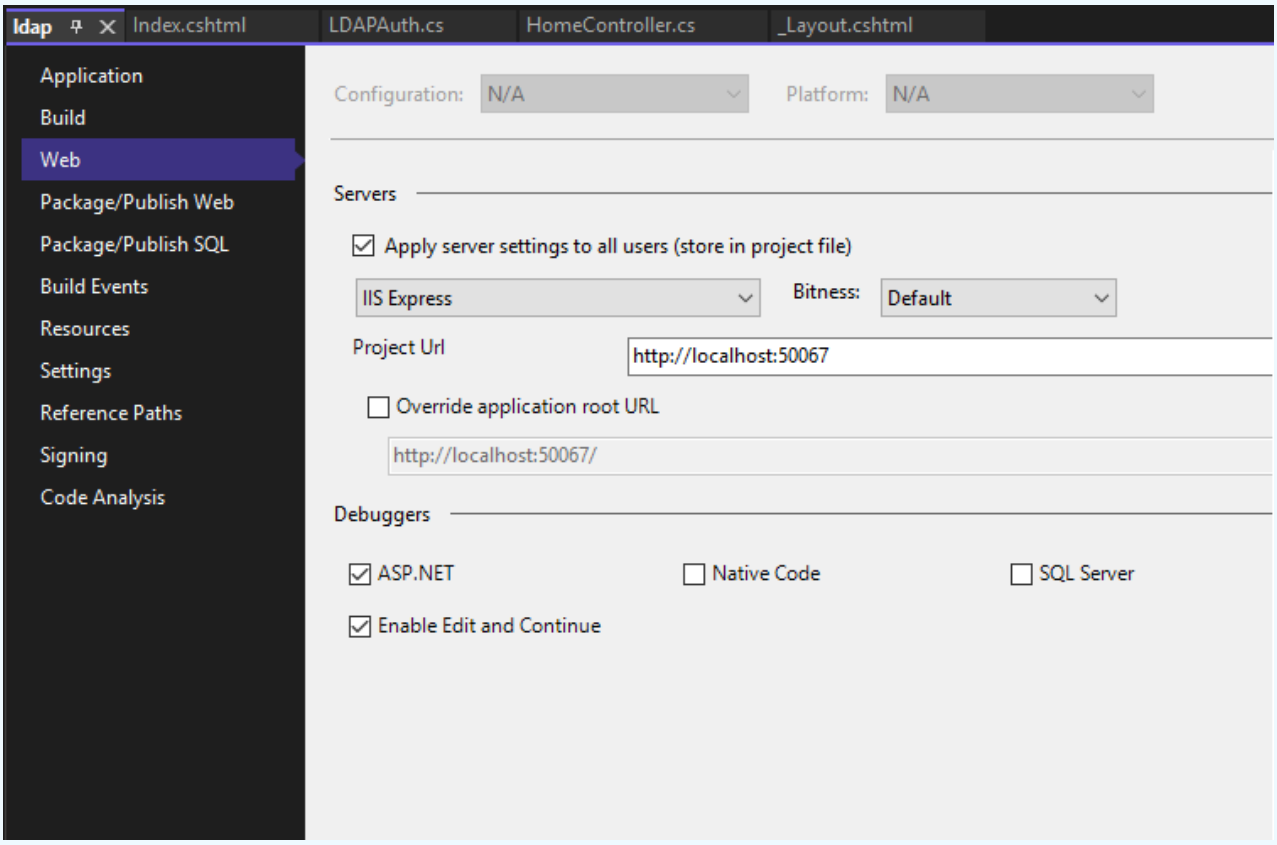
```
// Method to validate the user's credentials against an LDAP directory
private bool ValidateCredentials(string username, string password)
{
    // Assuming "LDAP://YourDomain" is the path to your LDAP directory
    using (var context = new PrincipalContext(ContextType.Domain, "WIN-SM252M527F1:389", "CN=jehad ahmad,CN=Users,DC=WIN,DC=jehad"))
    {
        // Perform actions, such as validating credentials
        return context.ValidateCredentials(username, password);
    }
}
```

LDAP PATH

Code explanation

This code snippet defines a HomeController class in an ASP.NET MVC application that handles user authentication against an LDAP directory. The Index action is used for displaying a login form, while the Panel action is a placeholder for a protected page. The Index action also includes a HttpPost version that receives the submitted username and password, validates them using the ValidateCredentials method, and sets an authentication cookie using FormsAuthentication if the credentials are valid. If the credentials are invalid, it adds a model error and redirects back to the login page. The ValidateCredentials method connects to the LDAP directory using a specified path and validates the provided credentials. Overall, this code provides a basic structure for implementing LDAP authentication in an ASP.NET MVC application.

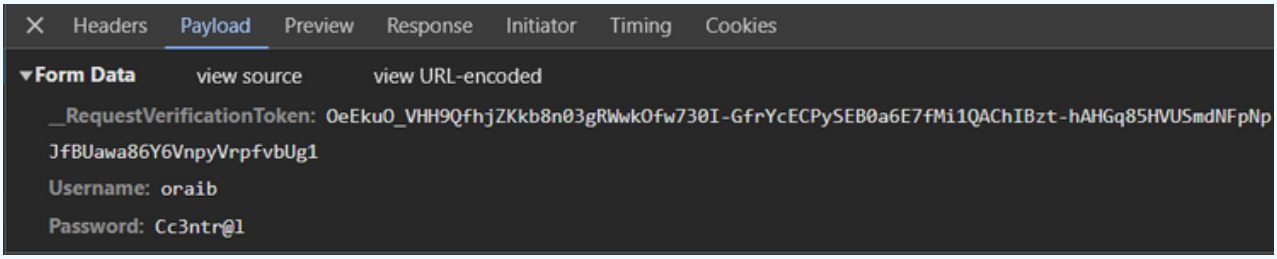
Configure the IIS express



To summarize the process of hosting an application on IIS Express locally and accessing it from another machine on the same network, follow these steps:

1. Configure IIS Express: Set up IIS Express to run your application locally on your machine. Make sure it is configured to listen on port 50067.
2. Find Your Machine's IP Address: Use the command prompt to find your machine's IP address. Use ipconfig on Windows or ifconfig on macOS/Linux to find the IPv4 address.
3. Allow Inbound Connections: Ensure that your machine's firewall settings allow inbound connections to port 50067. You may need to create an inbound rule to allow traffic on this port.
4. Access the Application from Another Machine: On another machine on the same network, open a web browser and navigate to http://[your_machine_ip]:50067 replacing [your_machine_ip] with the IP address of your machine.


Not secure 192.168.91.128:50067/Home/Index
Gmail
Google Drive
YouTube
Maps
Facebook
WhatsApp
Elearning
LDAP TESTER
Home
Login
Username
oraib
Password
Log in
In case of a unsuccessful login, user will be transmitted to the same page to retry.
LDAP TESTER
Success
LDAP TESTER
Home
Login
Username
Password
Log in



The payload is not encrypted because LDAP (Lightweight Directory Access Protocol) by default does not encrypt data in transit. LDAP typically operates over port 389 and sends data, including usernames, passwords, and other information, in plaintext. This lack of encryption makes it possible for the payload to be intercepted and read by unauthorized parties, which can be a security concern.

To encrypt the payload and protect the data in transit, LDAPS (LDAP over SSL/TLS) can be used. LDAPS operates over port 636 and encrypts the LDAP communication using SSL/TLS, providing a secure channel for transmitting sensitive information. Using LDAPS helps ensure the confidentiality and integrity of the data exchanged between the LDAP client and server.