Guide101: Building an Automated Zero Trust Hybrid Cloud Network

1. Introduction

Welcome! This guide provides a step-by-step walkthrough for building a complete, automated Zero Trust hybrid cloud network. You will simulate an on-premises corporate network using virtualization and connect it securely to a cloud environment hosted in Microsoft Azure.

The final architecture will feature:

- **Infrastructure as Code (IaC)** using Terraform for automated Azure deployment.
- **Hybrid Identity** with on-prem Active Directory synced to Microsoft Entra ID.
- **A secure network underlay** using a Site-to-Site VPN.
- **A Zero Trust Network Access (ZTNA)** overlay using Tailscale for identity-based access.

Architectural Overview:

2. Prerequisites

Before you begin, ensure you have the following:

- **Software:**
    - A hypervisor like Oracle VirtualBox or VMware Workstation.
    - Terraform installed on your local machine.
    - Azure CLI installed and configured.
- **Accounts & Access:**
    - A Microsoft Azure subscription with sufficient permissions to create resources.
    - A Tailscale account (a free personal account is sufficient).
- **ISO Files:**
    - pfSense Firewall ISO.
    - Windows Server 2022 ISO.

    - Windows 10/11 Enterprise ISO.
- **Knowledge:**
    - Basic understanding of IP networking (subnets, routing, DNS).
    - Familiarity with the command line.
    - Basic experience with virtualization.

3. Part 1: Build the Simulated On-Premises Network

In this section, you will use your hypervisor to create a virtual network that mimics a small corporate office.

Step 1: Configure the Hypervisor Network

1. In VirtualBox or VMware, create a new internal network or vSwitch.
2. Set the network address space to 192.168.50.0/24. This will be your internal LAN.

Step 2: Install and Configure pfSense Firewall

1. Create a new virtual machine for pfSense.
2. Assign two network interfaces: one connected to your home network (WAN) and one connected to the 192.168.50.0/24 LAN you just created.
3. Install pfSense from the ISO.
4. Configure the LAN interface with the static IP address 192.168.50.1.
5. Enable the DHCP server on the LAN interface to assign IP addresses to other VMs in the 192.168.50.0/24 range.

Step 3: Install and Configure Windows Server Domain Controller

1. Create a new VM for Windows Server 2022. Connect it to the 192.168.50.0/24 LAN.
2. Install Windows Server.
3. Once installed, add the "Active Directory Domain Services" role.
4. Promote the server to a new Domain Controller for a new forest. Name the domain something like mycorp.local.
5. Ensure the server's DNS points to itself (127.0.0.1) and that it can resolve public domains.

Step 4: Set Up the Windows Client

1. Create a final VM for your Windows 10/11 client. Connect it to the 192.168.50.0/24 LAN.
2. Install Windows.
3. Once installed, join the machine to the mycorp.local domain you created in the previous step.
4. Log in with a domain user account to verify the setup.

4. Part 2: Deploy Azure Infrastructure with Terraform

Now, you will define and deploy your entire cloud infrastructure using Terraform.

Step 1: Prepare Terraform and Authentication

1. Create a new directory for your Terraform project.
2. Open your terminal, navigate to the new directory, and log in to Azure by running az login.

Step 2: Define the Infrastructure

Create a main.tf file. This code will define all the necessary Azure resources.

```
# Configure the Azure Provider
provider "azurerm" {
  features {}
}

# 1. Create a Resource Group
resource "azurerm_resource_group" "rg" {
  name     = "ZeroTrust-Hybrid-RG"
  location = "East US" # Choose a region that works for you
}

# 2. Create a Virtual Network (VNet)
```

```
resource "azurerm_virtual_network" "vnet" {
  name                = "ZTVNet"
  address_space       = ["10.0.0.0/16"]
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
}

# 3. Create a Subnet
resource "azurerm_subnet" "subnet" {
  name                 = "WebServerSubnet"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.0.1.0/24"]
}

# 4. Create a Public IP for the VM
resource "azurerm_public_ip" "pip" {
  name                = "WebServer-pip"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  allocation_method   = "Static"
  sku                 = "Standard"
}

# 5. Create a Network Security Group (Firewall)
resource "azurerm_network_security_group" "nsg" {
  name                = "WebServer-nsg"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name

  security_rule {
    name                       = "AllowSSH"
    priority                   = 100
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    source_port_range          = "*"
    destination_port_range     = "22"
    source_address_prefix      = "YOUR_HOME_IP_ADDRESS" #
IMPORTANT: Lock this down
    destination_address_prefix = "*"
  }

  security_rule {
    name                       = "AllowHTTP"
    priority                   = 200
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    source_port_range          = "*"
    destination_port_range     = "80"
    source_address_prefix      = "*"
    destination_address_prefix = "*"
  }
```

```
}

# 6. Create a Network Interface for the VM
resource "azurerm_network_interface" "nic" {
  name                = "WebServer-nic"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name

  ip_configuration {
    name                          = "internal"
    subnet_id                     = azurerm_subnet.subnet.id
    private_ip_address_allocation = "Dynamic"
    public_ip_address_id          = azurerm_public_ip.pip.id
  }
}

# 7. Associate NSG with the NIC
resource "azurerm_network_interface_security_group_association"
"nsg_assoc" {
  network_interface_id      = azurerm_network_interface.nic.id
  network_security_group_id = azurerm_network_security_group.nsg.id
}

# 8. Create a cloud-init script to bootstrap the VM
resource "local_file" "cloud_init_script" {
  content = <<-EOT
    #!/bin/bash
    sudo apt-get update
    sudo apt-get install -y docker.io
    sudo systemctl start docker
    sudo systemctl enable docker
    sudo docker run --name nginx-server -d -p 80:80 nginx
  EOT
  filename = "${path.module}/install_docker_nginx.sh"
}

# 9. Create the Linux Virtual Machine
resource "azurerm_linux_virtual_machine" "vm" {
  name                = "WebServerVM"
  resource_group_name   = azurerm_resource_group.rg.name
  location            = azurerm_resource_group.rg.location
  size                = "Standard_B1s"
  admin_username        = "azureuser"
  admin_password        = "P@ssw0rd1234!" # Change to a secure password or
use SSH keys
  disable_password_authentication = false
  network_interface_ids = [azurerm_network_interface.nic.id]

  os_disk {
    caching              = "ReadWrite"
    storage_account_type = "Standard_LRS"
  }

  source_image_reference {
    publisher = "Canonical"
    offer     = "0001-com-ubuntu-server-focal"
    sku       = "20_04-lts-gen2"
    version   = "latest"
  }

  custom_data = base64encode(local_file.cloud_init_script.content)
}
```

Step 3: Deploy the Infrastructure

1. Run `terraform init` to initialize the project.
2. Run `terraform plan` to see what resources will be created.
3. Run `terraform apply` and type `yes` to deploy your Azure infrastructure.

5. Part 3: Establish Hybrid Connectivity

Connect your on-prem and cloud environments.

Step 1: Configure the Site-to-Site VPN (Underlay)

This creates a stable, private connection for network-level routing.

1. In Azure, create a Virtual Network Gateway and a Local Network Gateway that represents your on-prem pfSense firewall.
2. In pfSense, configure the IPsec connection to match the settings of your Azure VPN Gateway.
3. **Troubleshooting Tip:** If the connection fails, carefully check the logs on both Azure and pfSense. A common issue is a mismatch in the IKE Phase 2 encryption settings, specifically the Diffie-Hellman (DH) Group. Ensure they are identical on both ends.

Step 2: Configure Hybrid Identity

This syncs your on-prem user accounts to the cloud.

1. On your on-prem Windows Server DC, download and install **Microsoft Entra Connect**.
2. Follow the wizard using the "Express Settings" option.
3. Provide credentials for both your on-prem mycorp.local administrator and your Microsoft Entra ID (Azure) global administrator.
4. Once the sync is complete, you will see your on-prem users appear in the Microsoft Entra ID portal.

6. Part 4: Implement the Zero Trust Overlay

This is the core of the Zero Trust model, where access is based on identity, not network location.

Step 1: Install Tailscale

1. On your on-prem **Windows 10/11 client VM**, install the Tailscale client.
2. In Azure, SSH into your **Ubuntu WebServerVM** and install Tailscale using the command line instructions from their website.
3. On both machines, authenticate the Tailscale client using your Tailscale account.

Step 2: Test Secure Access

1. In the Tailscale admin console, you will now see both your on-prem client and your Azure VM listed with unique 100.x.x.x IP addresses.
2. From your on-prem Windows client, open a web browser and navigate to the **Tailscale IP address** of the Azure VM.
3. You should see the "Welcome to nginx!" page.

This connection is a direct, encrypted, point-to-point tunnel. You have successfully accessed the cloud resource without relying on the VPN or public IP addresses, proving access based on authenticated device identity.

7. Conclusion & Next Steps

Congratulations! You have successfully built a hybrid cloud network based on Zero Trust principles. You have automated the cloud deployment, established hybrid identity, and implemented a modern, identity-aware security overlay.

From here, you can explore several enhancements:

- **CI/CD Automation:** Set up a GitHub Actions pipeline to automatically run terraform plan and apply on code changes.
- **Secrets Management:** Use Azure Key Vault to store sensitive data like passwords and API keys instead of hardcoding them.
- **Advanced ZTNA:** Use Tailscale ACLs to create granular access rules, such as allowing only specific users to SSH into the server while allowing all users to access the web server.