**Moving a Website to Azure Using Terraform: An Author’s Journey**

**The Project**

As an author and coder, I set out to revamp my 5-year-old author website and migrate it to Azure. This journey allowed me to use Terraform for provisioning cloud resources, making infrastructure management more efficient. While I appreciated WordPress, my coding skills offered new opportunities for customization, control, and security.

This project served a dual purpose: enhancing my author website and creating a strong portfolio project showcasing Terraform and cloud computing skills.

### **Why Have an Author Website?**

1. **Linking Your Work:** An author website provides a central hub where readers can find links to your books, preorder options, or additional content.
2. **Reader Magnets:** Offering free downloadable content, such as a prequel novella, maps, or character sheets, can attract readers. These "reader magnets" help showcase your work and engage your audience.
3. **Mailing List Development:** Your website can collect email addresses in exchange for reader magnets. With tools like Azure Event Grid, you can automate email responses, including delivering newsletters or opting out unsubscribers seamlessly.

**Preparing Newsletters**

Years ago, my newsletters were inaccessible for screen readers and Braille devices. Transitioning to accessible formats using tools like Google Docs or Word ensures inclusivity.

**Steps to Create an Accessible Newsletter**:

* Use a template from Google Docs or Word.
* Enable accessibility features (e.g., **Tools → Accessibility** in Google Docs or **Accessibility Checker** in Word).
* Add links to your books or recommendations.
* Save the document as a PDF after verifying all links.
* Store newsletters in a dedicated folder, such as newsletters, for deployment.

### **Why Terraform?**

Terraform is an open-source Infrastructure as Code (IaC) tool. It simplifies resource provisioning for platforms like Azure, AWS, and Google Cloud. With its JSON- and Python-like syntax, it’s approachable for developers familiar with these formats.

I used Terraform for:

* Resource grouping for simplified billing and deletion.
* Secure storage using Azure Key Vault.
* Automating event-driven processes, such as mailing list management.

**Terraform Course**

I enrolled in the Zero to Mastery’s extremely thorough [Terraform course taught by Andrei Dumitrescu](https://zerotomastery.io/courses/learn-terraform-certification/). Although the course focused on AWS, I adapted the exercises for Azure, leveraging tools like ChatGPT and GitHub Copilot to troubleshoot. This dual approach enabled me to list both AWS and Azure on my resume, expanding my employability.

**Using a Free Website**

I used a free website, [Helios by HTML5UP](https://html5up.net/helios) as my website. It featured sections I could use for book links, series overviews, and newsletters. While custom coding was an option, my priority was functionality over aesthetic perfection.

### **Terraform Implementation**

Key components for this project included:

1. **Resource Group**: A container for all related resources for streamlined billing and management.
2. **Azure Key Vault**: For securely storing sensitive information like secrets and credentials.
3. **Azure Blob Storage**: For hosting the static website and newsletters.
4. **Cosmos DB**: For managing mailing list data.
5. **Azure Event Grid**: For triggering automated workflows, such as newsletter distribution and handling opt-outs.
6. **Docker**: To containerize the website, simplifying version control and deployment.

**Example Code**

**Key Vault Setup**

# Create a Key Vault for Secrets

resource "azurerm\_key\_vault" "author\_example\_main" {

name = var.key\_vault\_name

location = azurerm\_resource\_group.author\_example\_main.location

resource\_group\_name = azurerm\_resource\_group.author\_example\_main.name

tenant\_id = data.azurerm\_client\_config.current.tenant\_id

sku\_name = "standard"

}

**Container Setup**

# Create an Azure Container Instance

resource "azurerm\_container\_group" "author\_example\_main" {

name = var.container\_group\_name

location = azurerm\_resource\_group.author\_example\_main.location

resource\_group\_name = azurerm\_resource\_group.author\_example\_main.name

os\_type = "Linux"

container {

name = var.container\_name

image = var.container\_image

cpu = "0.5"

memory = "1.5"

ports {

port = 80

protocol = "TCP"

}

ports {

port = 443

protocol = "TCP"

}

environment\_variables = {

"ENV\_VAR\_NAME" = "value"

}

}

tags = {

environment = "testing"

}

}

**Event Listeners: Python**

main.py

import os

import azure*.functions* as func

from azure.communication.email import EmailClient, EmailContent, EmailAddress, EmailMessage

*def* main(*event*: func*.EventGridEvent*):

email\_client = EmailClient*.from\_connection\_string(os.environ["COMMUNICATION\_SERVICES\_CONNECTION\_STRING"])*

# Load the HTML content from the welcome\_email.html file

with open("your\_folder\welcome\_email.html", "r") as file:

html\_content = file*.read()*

content = EmailContent(

*subject*="Welcome to Our Newsletter!",

*plain\_text*="Thank you for subscribing!",

*html*=html\_content

)

message = EmailMessage(

*sender*="author@example.com",

*content*=content,

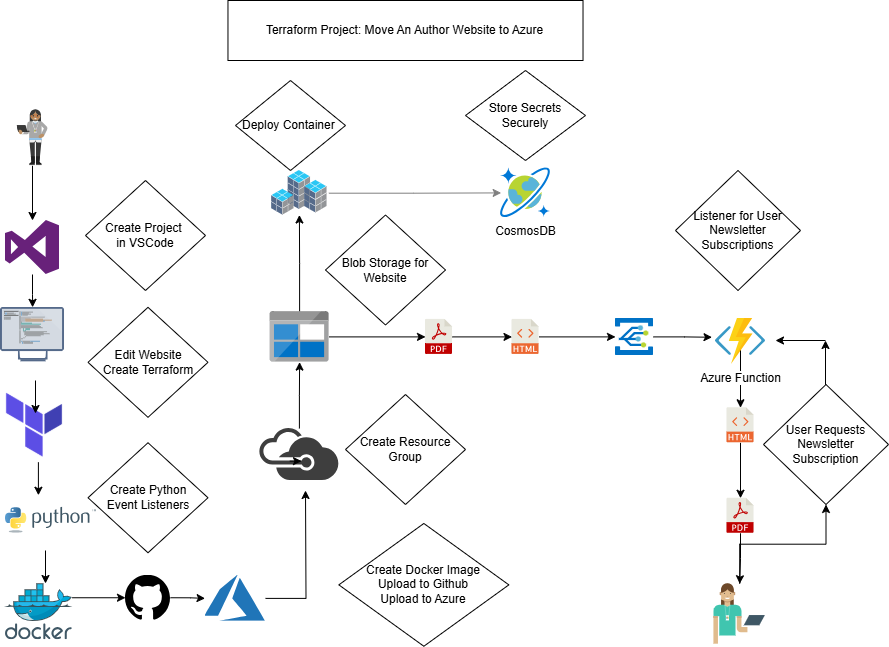
*recipients*=[EmailAddress(*email*="recipient@example.com")]

)

response = email\_client*.send(message)*

print(*f*"Email sent with message ID: *{*response*.message\_id}*")

**Deployment Diagram**

****

**Advanced File Information**

**Terraform Files**

Terraform takes several files.

* The **main.tf** specifies the provider and all the resources being provisioned.
* The **variables.tf** file specifies the names of all these resources.
* The **terraform.tfvars** file is optional, and is used for managing configuration variables. This is helpful if there are multiple environments (such as development and production), or one wants to keep variable values separate from the main configuration files.
* The **outputs.tf** will output certain things into the command line.

For this project, **the main.tf** file must have:

* A defined resource group
* A storage account. In my case, a storage account and blob containers for newsletters and templates.
* Static website hosting and uploading website files or Docker image(s).
* The Cosmos DB database for my Cosmos DB account, database, and container.
* An Azure Function App with settings for Cosmos DB.
* An Event Grid topic and subscription for newsletters and opt-out events.
* A Key Vault to store secrets for the email service and Cosmos DB.
* A container registry and container instance for the Docker image.

The **variables.tf** file must have:

* A resource group name
* A location
* A storage account name
* A Cosmos DB account name
* A Function App name
* A Key Vault name
* A container registry name
* A container group name
* A container name
* A container image

The **terraform.tfvars** file has all the same resources as the variables.tf file with their names.

The **outputs.tf** file provides outputs for:

* The resource group name
* The storage account name
* The Cosmos DB account name
* The function app default hostname
* The Key Vault URI (which will allow applications to securely access a specific version of a secret)

**Docker File**

The Docker file needs:

* A Python runtime
* A working directory in the container
* Set the working directory in the container
* Needed packages installation
* Expose ports 80 for HTTP and 443 for HTTPS incoming traffic
* Expose outgoing ports
* Define the environment variable
* Run app.py when the container launches

**Other Files**

* I also needed a few short Python and JSON files for my event listeners.
* I also coded my welcome email in HTML.

### **Cost Efficiency**

Azure services like Blob Storage and Cosmos DB are inexpensive, costing between $5 and $10 per month. Combined with a $35 annual domain fee, this setup was significantly cheaper than my previous $150 annual hosting expense.

### **Lessons Learned**

This project wasn’t just about moving a website; it was about mastering tools and techniques to secure, scale, and automate cloud deployments. I deepened my understanding of Terraform, Azure services, and Docker while building a dynamic portfolio project.

For aspiring coders and authors, this journey exemplifies the possibilities of blending creativity and technology.

### **Final Thoughts**

Whether you’re a writer, coder, or both, embracing modern tools can redefine what you can achieve. My author website now stands as a testament to the power of learning and implementing new skills.

### **Explore the Code**

The complete example code for this project is available on my GitHub: <https://github.com/lauralstephenson/pseudocode_terraform_examples_for_website>. Feel free to explore, fork, or contribute!