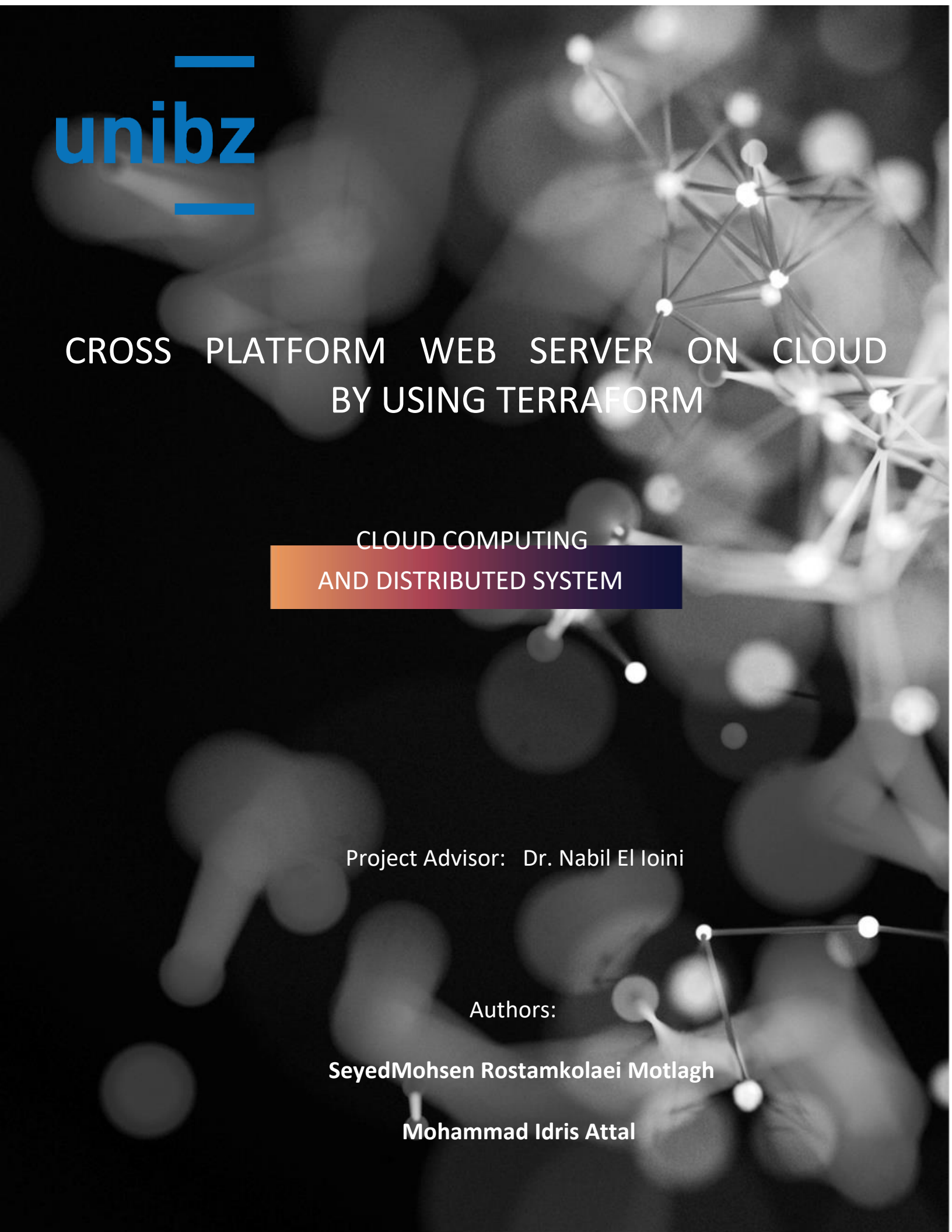


CROSS PLATFORM WEB SERVER ON CLOUD BY USING TERRAFORM

CLOUD COMPUTING
AND DISTRIBUTED SYSTEM

Project Advisor: Dr. Nabil El Ioini

Authors:

SeyedMohsen Rostamkolaei Motlagh

Mohammad Idris Attal

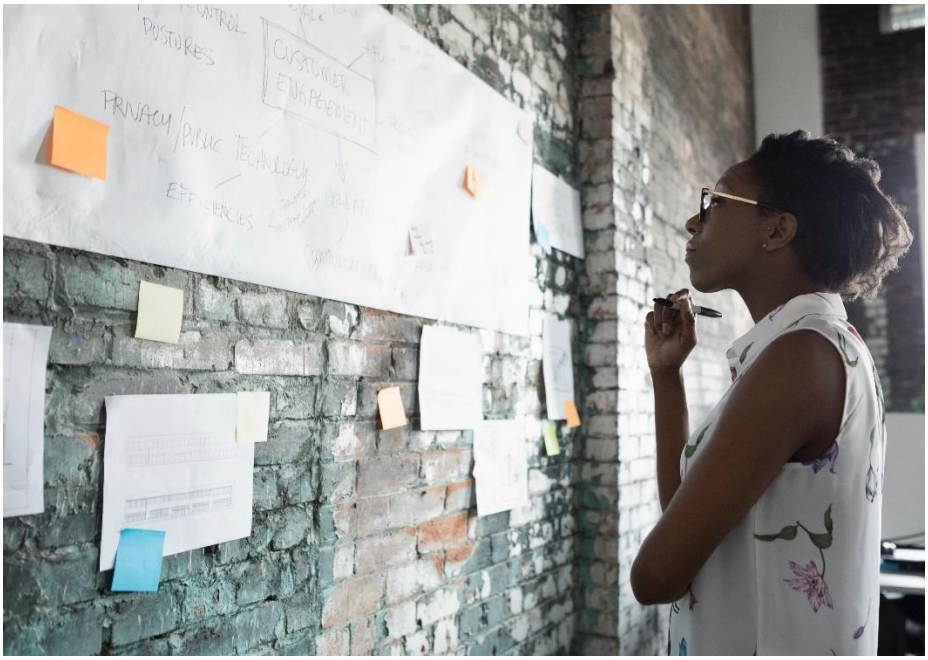
Table of Contents

PROBLEM DISCRIPTION

Cloud Configurations is a challenging process when it comes to implementation of it in large scales, especially when an individual is working with one of the famous cloud providers services such as AWS, GCP, Azure and so on. This implementation will bring challenges such as misconfiguration, human error or in some cases it will be time consuming as well.

We want to create a framework with the help of terraform which is an Infrastructure as code software tool. This framework will allow users to create their cloud infrastructure (web server) without any manual creation of the required resources for the web server, what the users' needs to do is, choose which platform (cloud provider) they want to use to deploy their web server and our solutions will exactly create their target resources in their chosen cloud provider.

It worth to mention that our solutions will bring accuracy, time saving and automation. This means that the user will take some small steps, and the required resources will be created for them in a minimum time span.



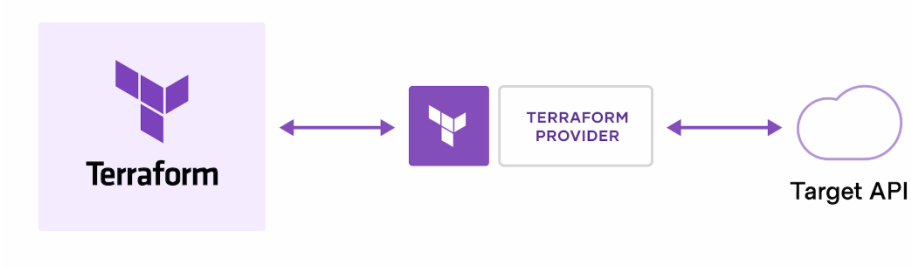
BASIC STRUCTURE AND TOOLS

TERRAFORM:

Terraform is an open-source infrastructure as code (IaC) software tool that allows DevOps engineers to programmatically provision the physical resources an application requires to run, with Terraform you are building your environment exactly as you wish with a pre-configured script to create a predefined end point.

Infrastructure as code is an IT practice that manages an application's underlying IT infrastructure through programming. This approach to resource allocation allows developers to logically manage, monitor and provision resources -- as opposed to requiring that an operations team manually configure each required resource.

Currently Terraform is available for integration into all the major public cloud providers (AWS, Azure, GCP, Oracle Cloud and Alibaba Cloud). It is also available for on-prem environments with VMware and Microsoft Hyper-V. Terraform creates and manages resources on cloud platforms and other services through their application programming interfaces (APIs). Providers enable Terraform to work with virtually any platform or service with an accessible API.



CLOUD PROVIDERS

AWS

A cloud provider is a company that delivers cloud computing based services and solutions to businesses and/or individuals. This service organization may provide rented and provider-managed virtual hardware, software, infrastructure and other related services. Cloud services are becoming increasingly desirable for companies because they offer advantages in terms of cost, scalability and accessibility.

GOOGLE CLOUD

Amazon web service is an online platform that provides scalable and cost-effective cloud computing solutions.

AWS is a broadly adopted cloud platform that offers several on-demand operations like compute power, database storage, content delivery, etc., to help corporates scale and grow.

AZURE

GCP is a public cloud vendor — like competitors Amazon Web Services (AWS) and Microsoft Azure. With GCP and other cloud vendors, customers are able to access computer resources housed in Google's data centers around the world for free or on a pay-per-use basis.

GCP offers a suite of computing services to do everything from GCP cost management to data management to delivering web and video over the web to AI and machine learning tools.

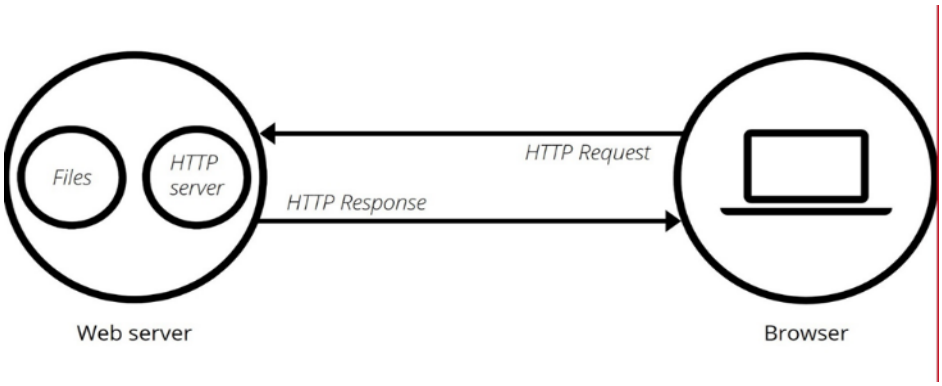
FINAL IMPLEMENTATION

The term web server can refer to hardware or software, or both of them working together.

1. On the hardware side, a web server is a computer that stores web server software and a website's component files (for example, HTML documents, images, CSS stylesheets, and JavaScript files). A web server connects to the Internet and supports physical data interchange with other devices connected to the web.

2. On the software side, a web server includes several parts that control how web users access hosted files. At a minimum, this is an HTTP server. An HTTP server is software that understands URLs (web addresses) and HTTP (the protocol your browser uses to view webpages). An HTTP server can be accessed through the domain names of the websites it stores, and it delivers the content of these hosted websites to the end user's device.

At the most basic level, whenever a browser needs a file that is hosted on a web server, the browser requests the file via HTTP. When the request reaches the correct (hardware) web server, the (software) HTTP server accepts the request, finds the requested document, and sends it back to the browser, also through HTTP. (If the server doesn't find the requested document, it returns a 404 response instead.)



To publish a website, you need either a static or a dynamic web server.

1. A static web server, or stack, consists of a computer (hardware) with an HTTP server (software). We call it "static" because the server sends its hosted files as-is to your browser.

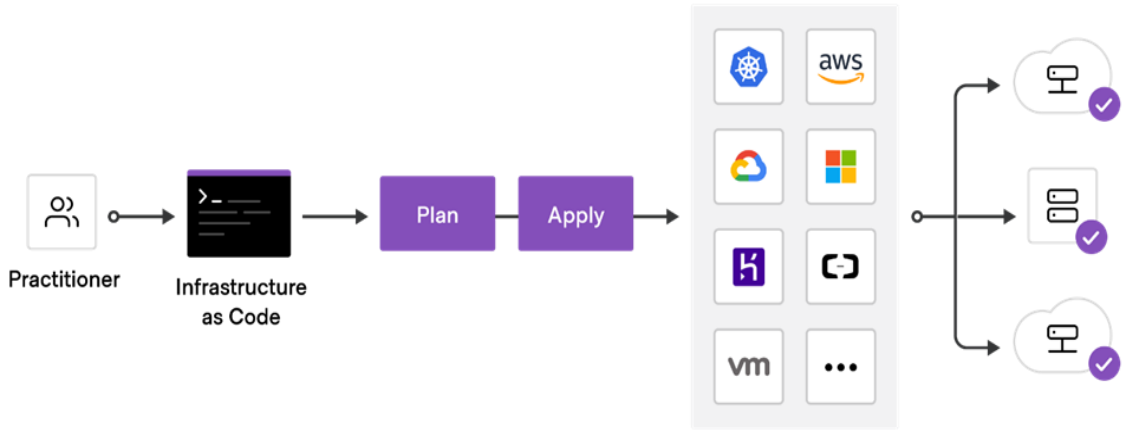
2. A dynamic web server consists of a static web server plus extra software, most commonly an application server and a database. We call it "dynamic" because the application server updates the hosted files before sending content to your browser via the HTTP server.

OUR WORKFLOW

The value of being able to write once and effectively deploy everywhere cannot be sniffed at, Simply by adding additional providers and changing a couple of lines, terraform can shift entire application stacks from one infrastructure to another, effectively making hybrid and multi-cloud that much more attainable.

In this project we want to use terraform for deploying a web server in 3 different cloud providers (AWS, Google Cloud and Azure), the goal is to use terraform infrastructure, to easily create the required resources for creating a web server and deploy our targeted code in it.

The workflow of the project is by running the bash command, the user will choose its targeted cloud provider choose between AZURE, AWS or Google Cloud then the framework will create the required resources on the selected cloud provider.



CHALLENGES

When it comes project implementation there is always challenges and the same was for us in this project as well.
The challenges were many for example, one of the main challenges was that there were no unique resources to follow to create our infrastructure as code. In addition to this, the required resources to create the web-server for each cloud providers were different, for example some of them needs less resources for creation and other needs more resources for creating the same web-server. Furthermore, user creation in cloud provider to do the experimentation with target cloud providers along with its credentials were other challenges that we faced during our solution implementation. Finally, lack of prior cloud deployment experience along with being new to work with Terraform were other challenges.
To sum up, challenges are part of each project and in fact they are the real reasons behind experiencing and learning new things, and the same was for us. During this project implementation period, we learn a lot which we can take it as precious take aways for our future working.

CONCLUSIONAND FUTUREENHANCEMENT

As conclusion, we would like to enhance our created solution or framework in several aspects, first, we would like to provide more cloud providers for our framework, second, we will try to enhance our already made solution to work more smoothly and also support whatever resources are needed for creating the web-servers. Finally, our intention would be to make our framework as fast and smooth as possible to handle all cases which are needed for building the required resources for cloud providers.
Finally, we would like to stress out that this is an experimentation project and there is always room for improvement and we would like to welcome each and every single feedback from your side. In addition to this, we would like to show our gratitude towards our respected Professor Nabil El Ioini, who was outstanding and cooperative with his help and inputs during working in this project.