Azure Web App is part of Azure App Service. It is a PAAS service as you don't manage the underlying Infrastructure. It provides a fully managed platform for deploying web applications developed in various programming languages, including .NET, Java, Node.js, Python, and PHP.

Our terraform template will create a resource group, an app service plan and a linux web app.

```
terraform {
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "3.0.0"
    }
  }
}

###If you don't specify provider configuration, TF will use CLI as the default provider
###If you want to use a different provider, you can specify it in the provider block

provider "azurerm" {
  features {}
}

####Create a resource group
resource "azurerm_resource_group" "rg" {
  name     = var.resource_group_name
  location = "East US"
}


####Create a service plan.
####asp => azurerm_service_plan
resource "azurerm_service_plan" "asp" {
  name                = var.service_plan_name
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  os_type             = "Linux"
  sku_name = "P1v3"
}

####Create a web app
####alwebapp=azurerm_linux_web_app
resource "azurerm_linux_web_app" "linux_webapp" {
  name                = var.web_app_name
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
  service_plan_id = azurerm_service_plan.asp.id
  public_network_access_enabled = true
  tags = var.custom_tags

  site_config {
    application_stack {
```

```
    java_version = 8
    java_server = "JBOSSEAP"
    java_server_version = "7"
    #To list all the available stacks, run az webapp list-runtimes --linux
   }
  }
}

output "resource_group_name" {
  value = azurerm_resource_group.rg.name
}

output "service_plan_name" {
  value = azurerm_service_plan.asp.name
}

output "web_app_name" {
  value = azurerm_linux_web_app.linux_webapp.name
}
```

Here's a brief summary of what it does:

1. Provider Configuration: It specifies that the Terraform configuration requires the Azure provider version 3.74.0.
2. Azure Provider Block: It configures the Azure provider with default settings. I am using CLI credentials.
3. Resource Group Creation: It creates an Azure Resource Group, Resource groups are containers for Azure resources.
4. Service Plan Creation: It sets up an Azure Service Plan named with Linux as the operating system and "P1v3" as the SKU (Service Level). A service plan defines the region, operating system, and capacity of the servers that will host the web app.
5. Linux Web App Creation: It creates an Azure Linux Web App within the previously created resource group. This web app uses the service plan defined earlier. It's configured to have public network access enabled and is tagged with the owner's name.

- Inside the web app configuration, it specifies that the application will use Java 8 and run on the "JBOSSEAP" stack with version 7.

To deploy the terraform template, Clone the repository.

git clone https://github.com/Abhimanyu9988/azure-maven-java-web-app.git

Let's deploy the infrastructure on Azure, We will run

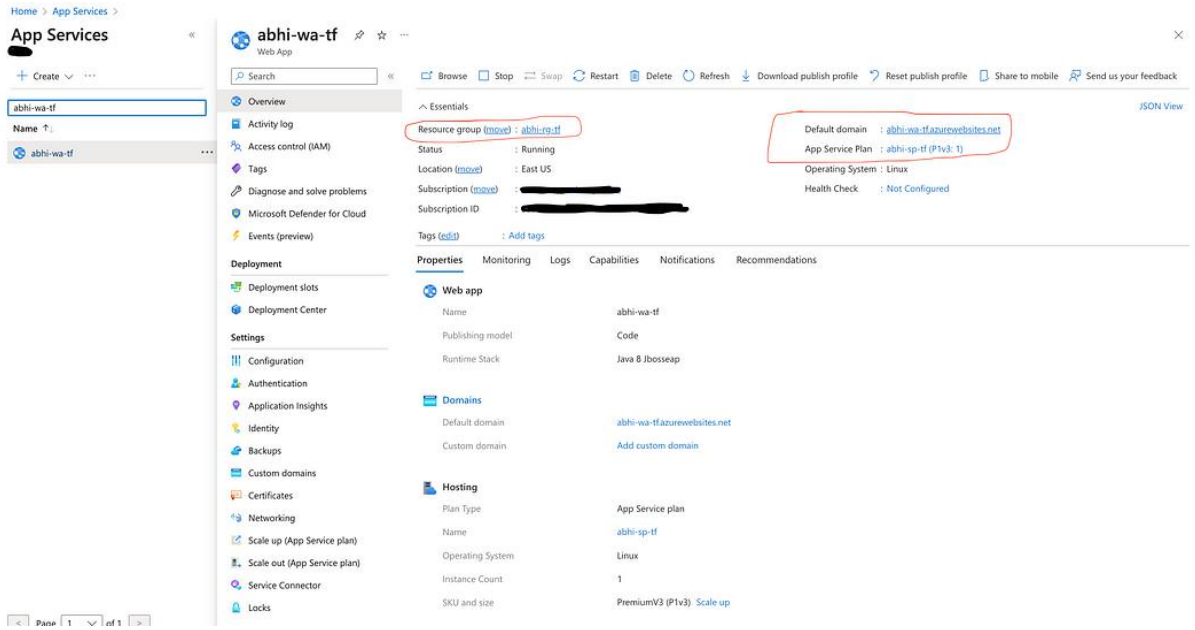
```
cd azure-maven-java-web-app
./deploytf.sh
```

```
var.resource_group_name
  Name of the Azure resource group

  Enter a value: abhi-rg-tf

var.service_plan_name
  Name of the Azure service plan

  Enter a value: abhi-sp-tf

var.web_app_name
  Name of the Azure Linux web app

  Enter a value: abhi-wa-tf
```

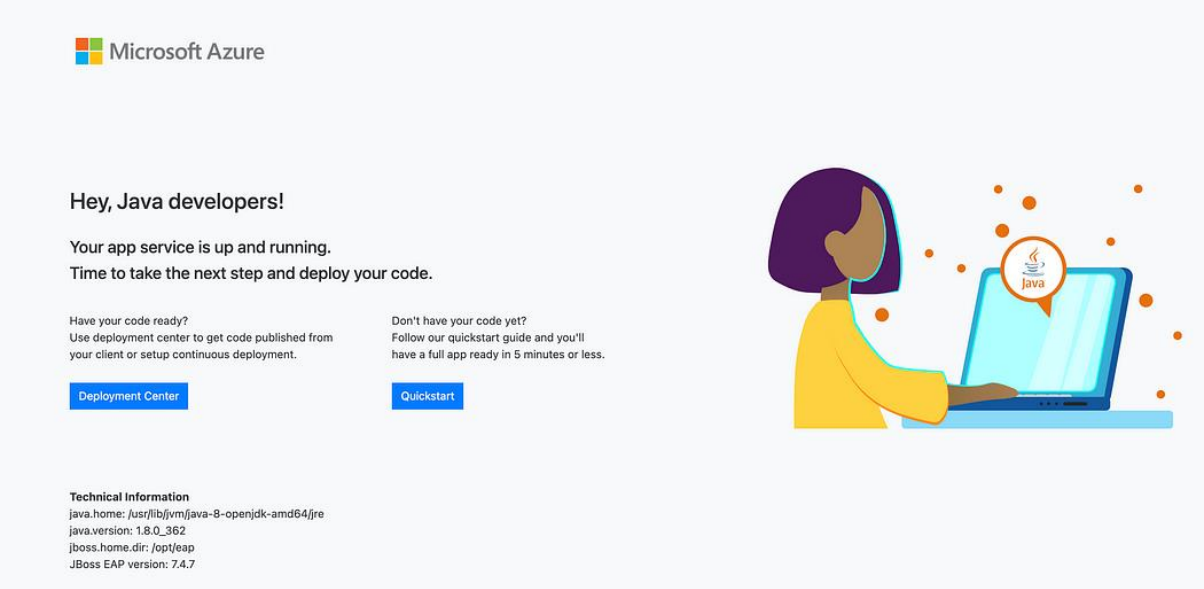
You would need to pass your resource_group, service_plan and web_app_name. For us we are passing abhi-rg-tf, abhi-sp-tf, abhi-wa-tf

```
Outputs:

resource_group_name = "abhi-rg-tf"
service_plan_name = "abhi-sp-tf"
web_app_name = "abhi-wa-tf"
```

Once done, You can go to Azure App Services -> Search for the name you typed before. I am searching for abhi-wa-tf. You can see the default domain, Resource group this Web app is created inder and App Service plan.



If you click on Default Domain which is normally
https://<web-app-name>.azurewebsites.net/



Beautiful!! Let's deploy your application now. Make sure you took a note of your web_app_name = "abhi-wa-tf"

To deploy the application, We will run->

./deployapp.sh

You will be prompted to choose
Please choose a Web Container Web App [<create>]:

Choose the WebApp you created. You would need to choose numerical number, The default value is to create a new web app. We have already configured the runtime and stack so in the next Confirm pop up, Select yes.

Once done, You can visit the default domain (https://<web-app-name>.azurewebsites.net/) listed on WebApp screen and you will see->

Hello World!

To destroy everything, From the same directory (azure-maven-java-web-app), Run->

terraform destroy

You will be prompted to enter Resource group, Service plan and Web app name you wish to destroy. To automate , you can edit the variables.tf file and add->

```
variable "resource_group_name" {
  description = "Name of the Azure resource group"
  type        = string
  default     = "<Resource_group_name>"
}
```

```
variable "service_plan_name" {
  description = "Name of the Azure service plan"
  type        = string
  default     = "<Service_plan_name>"
}

variable "web_app_name" {
  description = "Name of the Azure Linux web app"
  type        = string
  default     = "<Web_app_name>
}
```