Day 10 Demo IAAC + Deployment models

Demo Terraform - Release pipeline

1. We already saw the usage of terraform to deploy from my vscode to azure.
2. We already have the terraform config file.
3. My terraform is already having the IAM contributor role after I have registered it in the app registrations -Azure.
4. Make sure that the VS has the manifest file.
5. Push it to your Azure Repo.
6. Make sure you have a storage account on azure to store your terraform.tfstate file.
7. Make sure you have correct values in your terraform file.
8. Create a Build Pipeline now.
9. Create a Release Pipeline.
10. Deployed the resource on the WebApp.

Demo - CLI Release Pipeline

- It is basically a command-line tools on the microsoft azure portal.
- It is similar to powershell/cmd on windows/mac os.
- Tasks that we use Azure CLI for :
  - Creation, Deleting, Managing.
- Manual Deployment
  - Understanding how CLI works.
- Later we will use CLI in CI/CD.
  - Just make the change to the release pipeline.
  - Similar to terraform/ARM but you add AZURE CLI TASK to deploy your resources.

Demo - DSC

- Desired State Configuration
- Example - Your are hosting a website on a virtual machine [Not on Azure WebApp]
- You need to have IIS up and running all the time.

Catch - You have to ensure that IIS is always present and if by any chance someone deletes it. IIS should be automatically installed.

You want to make sure the configuration of your vm does not change.

Solution - Powershell Desired State Configuration

In order to implement DSC we will use a service called as Azure Automation. It is a inbuilt server which we can use to implement DSC.

Demo

- We will start with a VM Datacenter 2019/2022.
- Script

```
Configuration WebServerConfig [Configuration Block]
{
   Import-DscResource -ModuleName PSDesiredStateConfiguration
   Node localhost

    {
       WindowsFeature WebServer  [IIS windows feature]
       {
          Ensure = 'Present'
          Name   = 'Web-Server'
       }
    }
}
```

- Overall this script typically compile it into a configuration and apply it to the target vm - in our case the windows datacenter 2019.
- Create a Azure automation account.
- Used the above script in state configuration > upload the script.
- Then we compiled the script.
- And assigned it to a target node.

DSC FILE REOSURCE - https://learn.microsoft.com/en-us/powershell/dsc/reference/resources/windows/fileresource?view=dsc-1.1

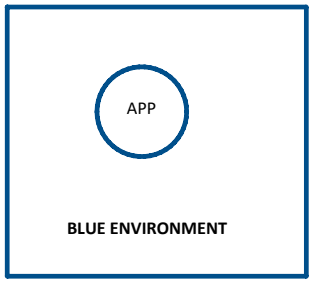
Deployment Models

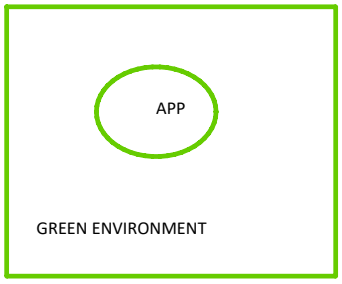
Why Did We Need Deployment Models?

- Fear of change.
- We needed a seamless process of deploying a new code.

Blue Green Deployment

This is also called as Red/Black





1. All the prod users are directed to my blue environment which Basically is my prod setup
2. If a software change has to be made, then a duplicate environment is setup The change is then applied to this duplicate environment [GREEN Environment].
3. Test is conducted on this setup
4. Testing is 100 completed. No bugs/errors reported
5. Once testing has been completed, users are then directed to the green environment.
6. Now the green environment becomes your prod environment.

At any given point in time only one environment is live and serving the production.
Other environment is just ideal.

Canary Deployment

- You gradually roll out the change to a small subset of users before making it available to the compete user base.
- Unlike in Blue/green you direct the 100% of traffic to the newer environment.

Rolling Deployment

- You gradually are updating the instances of the application to a new versions while ensuring that the application remains available and responsive.