## IaC on Azure - Introduction to IaC and Terraform

### What is IaC

* Infrastructure as Code (IaC) is a methodology that allows you to manage and provision your infrastructure using code, rather than manual configuration. This approach enables organizations to automate their infrastructure management, which can improve efficiency, consistency, and scalability. IaC is a crucial aspect of modern DevOps practices, as it allows teams to work in a more collaborative and efficient manner.

One of the most popular tools used for implementing IaC is Terraform. Terraform is an open-source tool that provides a simple and consistent way to define and provision infrastructure across multiple cloud providers, including Azure. It allows you to write your infrastructure in a high-level, human-readable language called HashiCorp Configuration Language (HCL) and provision it on Azure using the Azure Provider. With Terraform, you can define your infrastructure as code, version it, and collaborate with your team members, which makes it easy to manage, maintain and scale.

## IaC Methodologies, Approach, and Best Practices

### **Infrastructure as Code (IaC) methodologies**

Infrastructure as Code (IaC) methodologies are the different approaches and techniques used to define and provision infrastructure using code. These methodologies provide organizations with a simple and consistent way to manage and provision their infrastructure.

Here are some common IaC methodologies:

1. **Imperative IaC:** This methodology involves using code to specify how to provision and manage your infrastructure. This approach is similar to using a programming language to write instructions for a computer. It is commonly used with tools like Ansible, Puppet, and Chef.
2. **Declarative IaC:** This methodology involves using code to specify what your infrastructure should look like, rather than how to provision it. This approach is similar to using a markup language to describe a document. It is commonly used with tools like Terraform, CloudFormation, and ARM templates.
3. **Hybrid IaC:** This methodology involves using a combination of both imperative and declarative approaches. This approach allows organizations to take advantage of the benefits of both methodologies.

**GitOps IaC:** This methodology involves using Git as the single source of truth for infrastructure management. This allows teams to version their infrastructure, collaborate with their team members and automate the provisioning and management of their infrastructure.

### **IaC approach and best practices**

When implementing Infrastructure as Code (IaC), it is important to follow certain best practices to ensure that your infrastructure is managed in a consistent and efficient way. Here are some best practices to keep in mind when using IaC:

1. **Use version control:** Version control systems like Git allow you to version your infrastructure and collaborate with your team members. This makes it easy to manage, maintain, and scale your infrastructure.
2. **Use modular design:** Use a modular design approach to define your infrastructure. This allows you to easily manage, maintain, and scale your infrastructure.
3. **Automate testing:** Automate testing of your infrastructure to ensure that it is provisioned and configured correctly. This helps to reduce the risk of human error and ensures that your infrastructure is deployed consistently across multiple environments.
4. **Use infrastructure as code testing:** Use infrastructure as code testing to ensure that your infrastructure is provisioned and configured correctly. This helps to reduce the risk of human error and ensures that your infrastructure is deployed consistently across multiple environments.
5. **Use continuous integration and continuous deployment (CI/CD) pipeline:**Use a CI/CD pipeline to automate the provisioning and management of your infrastructure. This allows you to deploy new features and improvements to your customers more quickly and efficiently.
6. **Monitor and manage your infrastructure:** Monitor and manage your infrastructure to ensure that it is running optimally. This allows you to quickly identify and resolve any issues that may arise.

By following these best practices, organizations can ensure that their infrastructure is managed in a consistent and efficient way, and they can take advantage of the benefits of using IaC.

## Comparing Options for Azure

* When implementing Infrastructure as Code (IaC) on Azure, there are several options available, including Azure Resource Manager (ARM) templates, Azure CLI and PowerShell, and Terraform. Each option has its own set of benefits and challenges, and organizations should choose the option that best fits their needs and requirements.

### **Bicep**

Bicep is a domain-specific language (DSL) that uses declarative syntax to deploy Azure resources. It allows you to define your infrastructure in a high-level, human-readable language, and use it to provision and manage your infrastructure. Bicep provides concise syntax, reliable type safety, and support for code reuse. It offers a first-class authoring experience for your infrastructure-as-code solutions in Azure. Bicep provides support for all resource types and API versions, a simple syntax, repeatable results, orchestration, modularity, integration with Azure services, a preview option, and no state or state files to manage. Additionally, Bicep is completely free and open source, supported by Microsoft support.

* **Pros**:
  + Bicep provides concise syntax and reliable type safety.
  + It immediately supports all preview and GA versions for Azure services.
  + It is integrated with Azure services such as Azure Policy, template specs, and Blueprints.
  + It provides a preview option to get a preview of changes before deploying the Bicep file.
  + It is completely free and open-source.
* **Cons**:
  + It is only supported on Azure, so it is not suitable for multi-cloud solutions.

### **Terraform**

Terraform is an open-source tool that can be used to deploy resources on Azure. It is well-suited for organizations that are looking for a multi-cloud solution and want to use a consistent tool across multiple cloud providers.

One of the major benefits of using Terraform is that it is a multi-cloud solution. This means that you can use the same tool and configuration language to provision and manage infrastructure across multiple cloud providers, including Azure, AWS, and GCP.

With Terraform, you can write your infrastructure in a high-level, human-readable language called HashiCorp Configuration Language (HCL) and use the same configuration to provision and manage your infrastructure on different cloud providers. This allows you to standardize your infrastructure management across multiple clouds, which can improve efficiency, consistency, and scalability.

Additionally, Terraform allows you to use modules to reuse and share code across different cloud providers. This makes it easy to manage, maintain, and scale your infrastructure across multiple clouds.

Furthermore, Terraform also provides a powerful state management feature that allows you to track and manage the state of your infrastructure across multiple clouds. This makes it easy to keep track of changes to your infrastructure and ensure that it is deployed consistently across multiple clouds.

* **Pros**:
  + Terraform is an open-source tool that can be used to deploy resources on Azure and other cloud providers.
  + It provides a consistent syntax across different cloud providers, making it suitable for multi-cloud solutions.
  + It allows you to version your infrastructure and collaborate with your team members.
  + It provides a simple and consistent way to manage and provision your infrastructure.
* **Cons**:
  + It may have a steeper learning curve for those who are not familiar with the HashiCorp Configuration Language (HCL)
  + It requires additional setup and configuration compared to other options like Azure CLI or ARM templates.

## Conclusion

### **Benefits of using IaC and Terraform on Azure**

In conclusion, using Infrastructure as Code (IaC) and Terraform on Azure provides several benefits for organizations. By using IaC, organizations can manage their infrastructure in a consistent and efficient way, using version control systems like Git, and automate the provisioning and management of their infrastructure.

Terraform, in particular, is an open-source tool that can be used to deploy resources on Azure and other cloud providers, providing a consistent syntax across different cloud providers, making it suitable for multi-cloud solutions. It allows organizations to version their infrastructure and collaborate with their team members, providing a simple and consistent way to manage and provision their infrastructure.

Additionally, the integration of Azure services with Bicep provides an advantage over other options, as it allows a preview of changes before deploying the Bicep file, and eliminates the need to manage state. It’s also supported by Microsoft support.

In any case, organizations should carefully consider their specific needs and requirements when choosing the right IaC methodology and tool for their infrastructure management on Azure. Each option has its own set of benefits and challenges, and choosing the right one can help organizations improve their infrastructure management and deployment processes.

### **Future of IaC and Terraform in the cloud computing industry**

The future of Infrastructure as Code (IaC) and Terraform in the cloud computing industry looks bright. The increasing adoption of cloud computing and the need for efficient and consistent infrastructure management has led to a growing demand for IaC solutions.

As organizations continue to adopt cloud computing, the use of IaC will become even more important. Organizations will need to manage and provision their infrastructure in a consistent and efficient way, and IaC will play a critical role in achieving this.

Terraform, in particular, is well-positioned for the future. Its ability to be used across multiple cloud providers makes it a versatile tool that can help organizations manage their infrastructure in a consistent way, regardless of the cloud provider they are using.

Additionally, the growing adoption of Kubernetes and the need for infrastructure provisioning and management in the containerized environment will also drive the adoption of Terraform. As more organizations adopt Kubernetes, the need for tools that can provision and manage infrastructure in this environment will also increase.

In summary, the future of IaC and Terraform in the cloud computing industry looks bright. As organizations continue to adopt cloud computing and containerization, the use of IaC and Terraform will become even more important. Organizations will rely on these tools to manage and provision their infrastructure in a consistent and efficient way, and we can expect to see continued innovation and development in the field of IaC and Terraform.

## Modules for our purposes

## In module.tf file we need to use this modules to deploy Vnet, subnets:

## “azure\_region”

## “resource group”

## “azure\_network\_vnet”

## “azure\_network\_route\_table”

## “azure\_network\_security\_group”

## “azure\_network\_subnet”

## In main.tf file we describe terraform version and our provider:

## Terraform {required\_version = “>= example”

## Azurerm = source and version

In output.tf we store description of:

* Id of the created subnet
* Subnet list
* Subnet name
* Subnet IP