DevOps Fundamentals - Infrastructure as Code (IaC)

[**🔍 1. Purpose** 1](#_Toc204938240)

[**📘 2. Theory** 1](#_Toc204938241)

[**🔧 3. Popular IaC Tools** 2](#_Toc204938242)

[**🧰 4. Prerequisites** 2](#_Toc204938243)

[**🧪 5. Code Example: ARM Template to Create Azure Storage Account** 2](#_Toc204938244)

[**🧪 Alternate: Bicep (Simpler Syntax)** 3](#_Toc204938245)

[**⚙️ 6. How to Deploy (Example with Azure CLI)** 3](#_Toc204938246)

[**📸 7. Snapshot: IaC Workflow** 3](#_Toc204938247)

[**✅ 8. Summary** 3](#_Toc204938248)

**🔍 1. Purpose**

To understand how **Infrastructure as Code (IaC)** automates the provisioning and management of infrastructure using **code** instead of manual processes.

**📘 2. Theory**

**🔹 What is Infrastructure as Code (IaC)?**

**Infrastructure as Code (IaC)** is the practice of **defining and managing infrastructure (servers, networks, databases, etc.) using code** rather than manual setup through a UI or CLI.

**🔹 Key Goals of IaC:**

| **🎯 Goal** | **💡 Description** |
| --- | --- |
| **Automation** | Eliminate manual setup and human error. |
| **Repeatability** | Ensure consistent environments across dev, test, and prod. |
| **Version Control** | Treat infra code like application code (use Git). |
| **Scalability** | Scale resources dynamically using scripts. |

**🔹 Benefits of IaC:**

| **✅ Benefit** | **💡 Explanation** |
| --- | --- |
| 💻 Faster Provisioning | Spin up infrastructure in minutes |
| 🛡️ Consistency | Same configuration every time |
| 🔁 Easy Rollback | Use versioned scripts to undo changes |
| 👥 Team Collaboration | Shareable, reviewable, and testable infra configs |

**🔧 3. Popular IaC Tools**

| **Tool** | **Platform** | **Language Style** |
| --- | --- | --- |
| **ARM Templates** | Azure | JSON |
| **Bicep** | Azure | DSL (simpler than JSON) |
| **Terraform** | Multi-cloud | HCL (HashiCorp Language) |
| **Pulumi** | Multi-cloud | TypeScript/C#/Python |
| **Ansible** | Multi-cloud | YAML |

**🧰 4. Prerequisites**

* Azure Subscription or Cloud Environment
* Basic knowledge of resources like VMs, App Services, etc.
* A code editor (VS Code)
* Installed CLI for the tool (e.g., az, terraform, etc.)

**🧪 5. Code Example: ARM Template to Create Azure Storage Account**

{

"contentVersion": "1.0.0.0",

"parameters": {

"storageAccountName": {

"type": "string"

}

},

"resources": [

{

"type": "Microsoft.Storage/storageAccounts",

"apiVersion": "2022-09-01",

"name": "[parameters('storageAccountName')]",

"location": "[resourceGroup().location]",

"sku": {

"name": "Standard\_LRS"

},

"kind": "StorageV2",

"properties": {}

}

]

}

This creates a **Storage Account** in Azure when deployed using az deployment group create.

**🧪 Alternate: Bicep (Simpler Syntax)**

param storageAccountName string

resource sa 'Microsoft.Storage/storageAccounts@2022-09-01' = {

name: storageAccountName

location: resourceGroup().location

sku: {

name: 'Standard\_LRS'

}

kind: 'StorageV2'

}

**⚙️ 6. How to Deploy (Example with Azure CLI)**

# Deploy ARM template

az deployment group create --resource-group my-rg --template-file storage.json --parameters storageAccountName=myuniquestorage

# OR deploy Bicep file

az deployment group create --resource-group my-rg --template-file storage.bicep --parameters storageAccountName=myuniquestorage

**📸 7. Snapshot: IaC Workflow**

Write Infra Code →

Store in Git →

Review/Approve →

Run Deployment →

Infra is Provisioned Automatically

**✅ 8. Summary**

| **Key Point** |
| --- |
| **Infrastructure as Code (IaC)** replaces manual infrastructure setup with **automated, version-controlled scripts**, ensuring consistent, reliable, and scalable environments across the SDLC. |