
PROLOGUE
Design and Develop a
Serverless Event-Driven
Microservice-Based Solution

Introduction to Infrastructure as Code and Terraform

Spells for Provisioning Cloud Resources

Conjuring Efficiency: DevOps and IaC

Terraform Magic: Spells for Provisioning Cloud Resources

DevOps

Set of practices, tools, and a cultural philosophy that **automates and integrates** the processes between software development and IT teams

Emphasizes **team empowerment**, **cross-team communication**, and **technology automation**.

DevOps Key Principles

Collaboration

DevOps Key Principles

Collaboration

Automation

DevOps Key Principles

Collaboration

Automation

Continuous Improvement

DevOps Key Principles

Collaboration

Automation

Continuous Improvement

Customer-Centric Action

DevOps Key Principles

Collaboration

Automation

Continuous Improvement

Customer-Centric Action

Create with End in Mind

DevOps Key Principles

- Collaboration
- Automation
- Continuous Improvement
- Customer-Centric Action
- Create with End in Mind

DevOps Impact on Software Quality

Automated Testing

DevOps Impact on Software Quality

Automated Testing

Faster Feedback

DevOps Impact on Software Quality

Automated Testing

Faster Feedback

Collaboration

DevOps Impact on Software Quality

Automated Testing

Faster Feedback

Collaboration

Infrastructure as Code (IaC)

DevOps Impact on Software Quality

Automated Testing

Faster Feedback

Collaboration

Infrastructure as Code (IaC)

Continuous Monitoring

DevOps Impact on Software Quality

- Automated Testing
- Faster Feedback
- Collaboration
- Infrastructure as Code (IaC)
- Continuous Monitoring
- **Security Integration**

DevOps Impact on Software Quality

- Automated Testing
- Faster Feedback
- Collaboration
- Infrastructure as Code (IaC)
- Continuous Monitoring
- Security Integration

Infrastructure as Code (IaC)

Practice of managing **infrastructure** – such as servers, networks, and storage – using **code**.

Infrastructure code is stored in repositories just like software allowing for better **collaboration** and **automation**.

IaC Usefulness

Consistency and Automation

IaC Usefulness

Consistency and Automation

Efficiency

IaC Usefulness

Consistency and Automation

Efficiency

Scalability

IaC Usefulness

Consistency and Automation

Efficiency

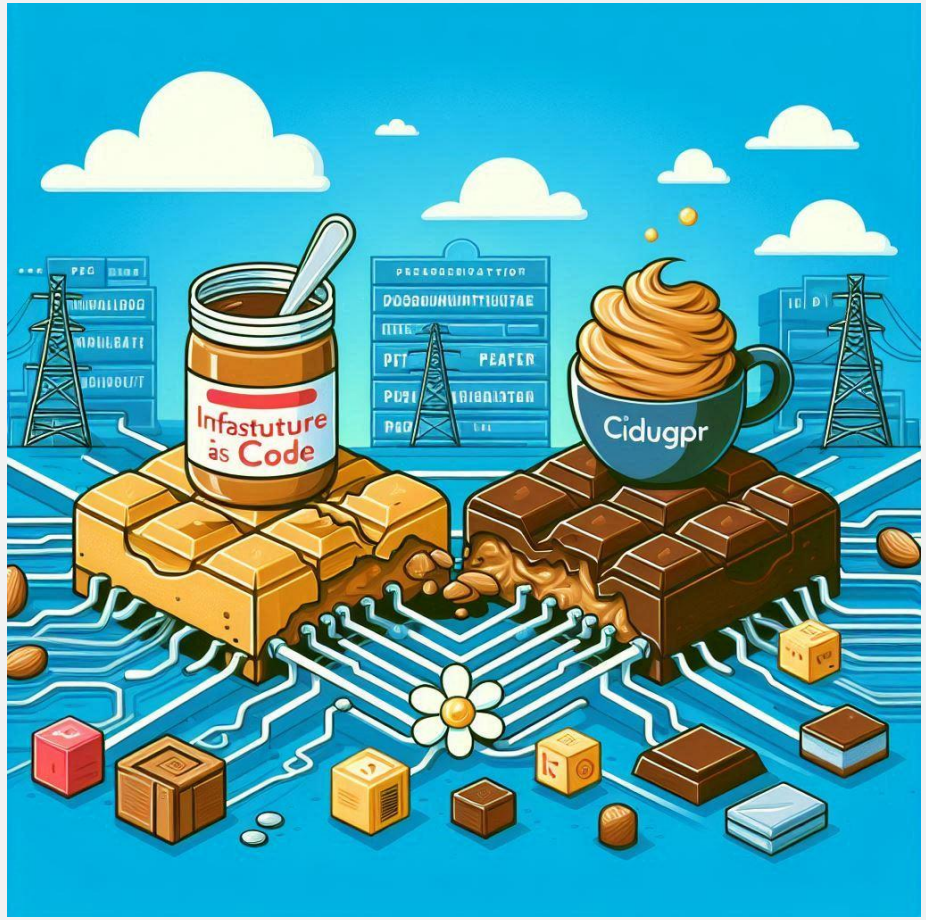
Scalability

Reduced Config Errors

IaC Usefulness

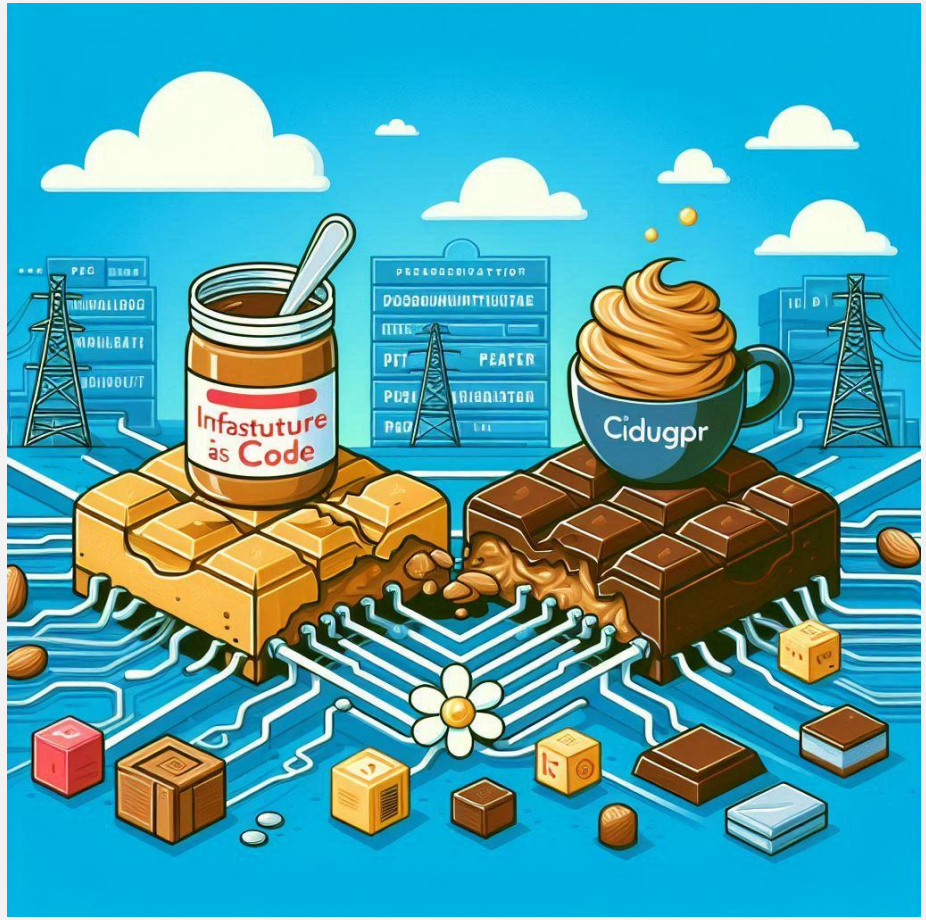
- Consistency and Automation
- Efficiency
- Scalability
- Reduced Config Errors

IaC and DevOps



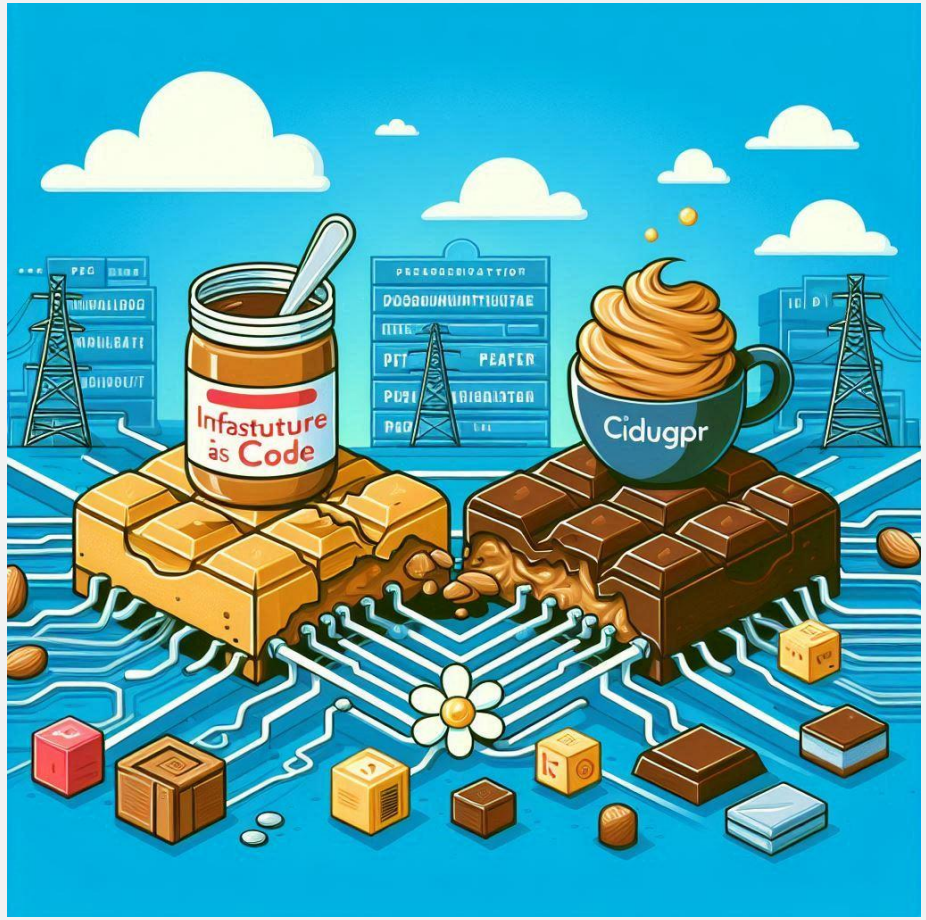
- Continuous Integration and Continuous Deployment (CI/CD)

IaC and DevOps



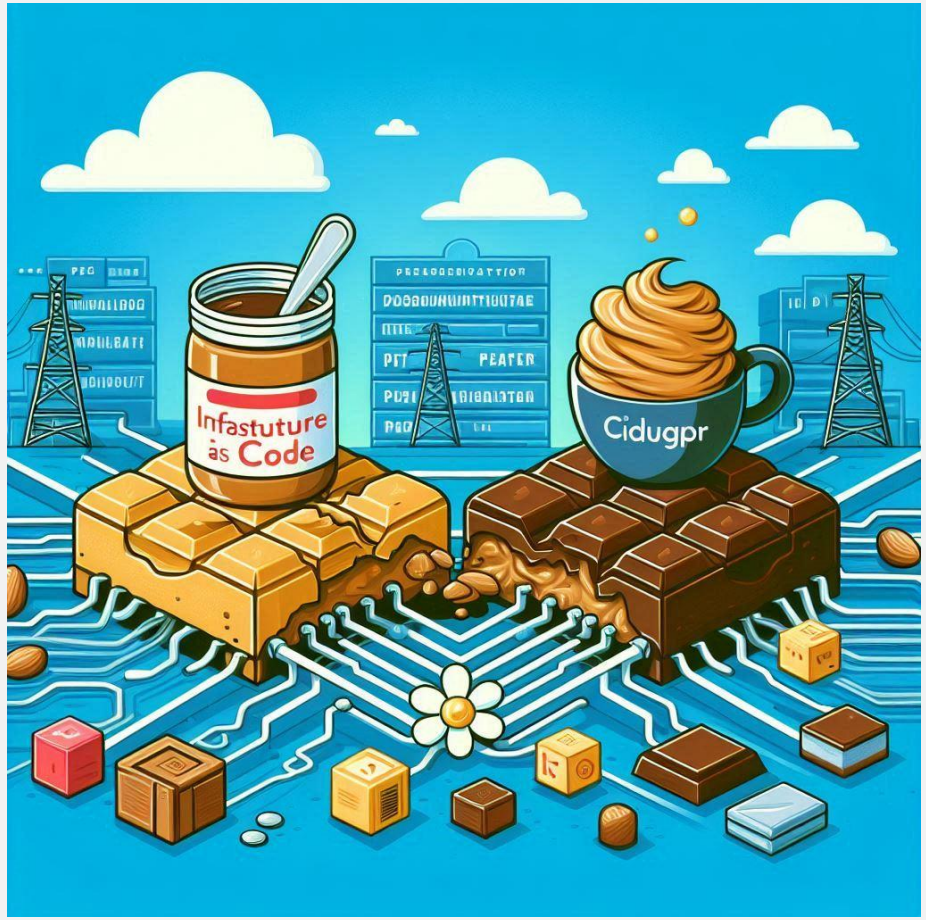
- Continuous Integration and Continuous Deployment (CI/CD)
- Version Control and Collaboration

IaC and DevOps



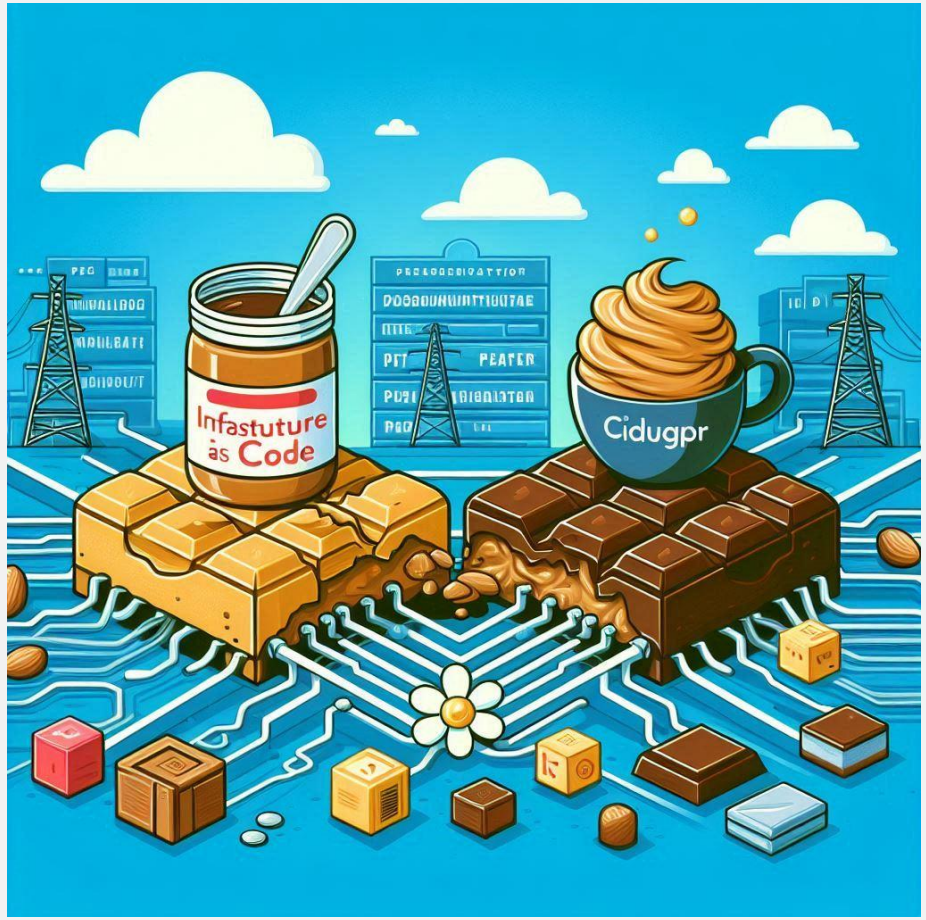
- Continuous Integration and Continuous Deployment (CI/CD)
- Version Control and Collaboration
- Automated Provisioning

IaC and DevOps



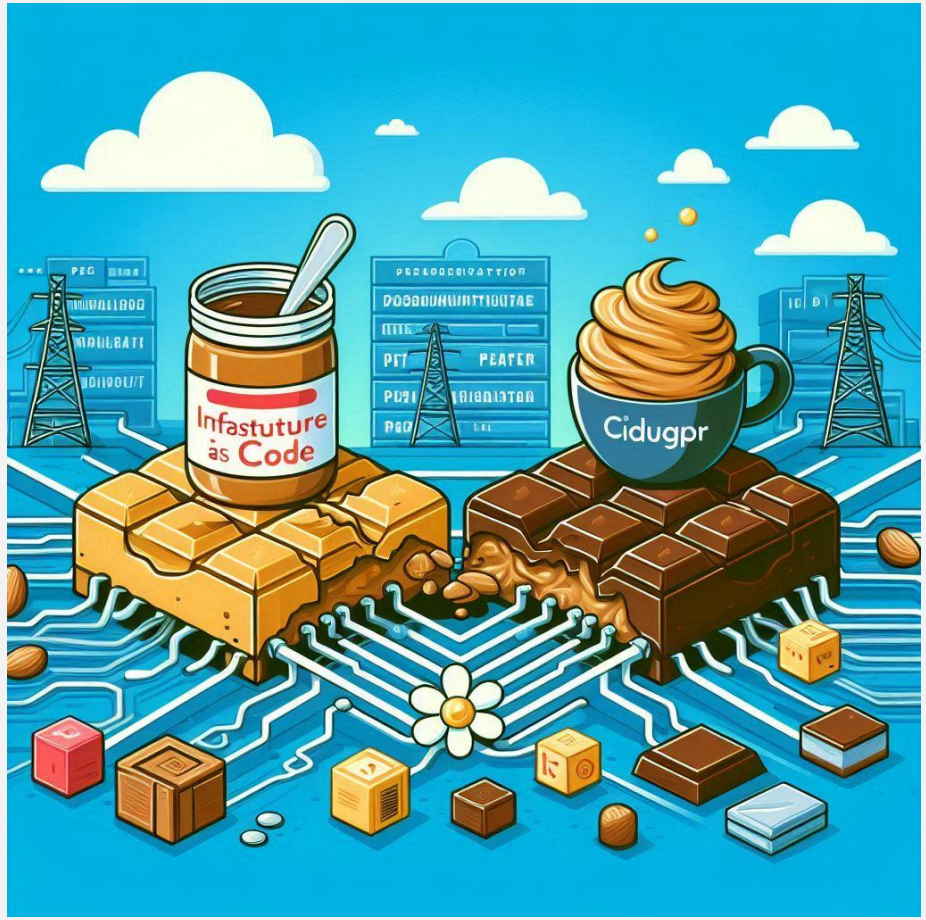
- Continuous Integration and Continuous Deployment (CI/CD)
- Version Control and Collaboration
- Automated Provisioning
- Immutable Infrastructure

IaC and DevOps



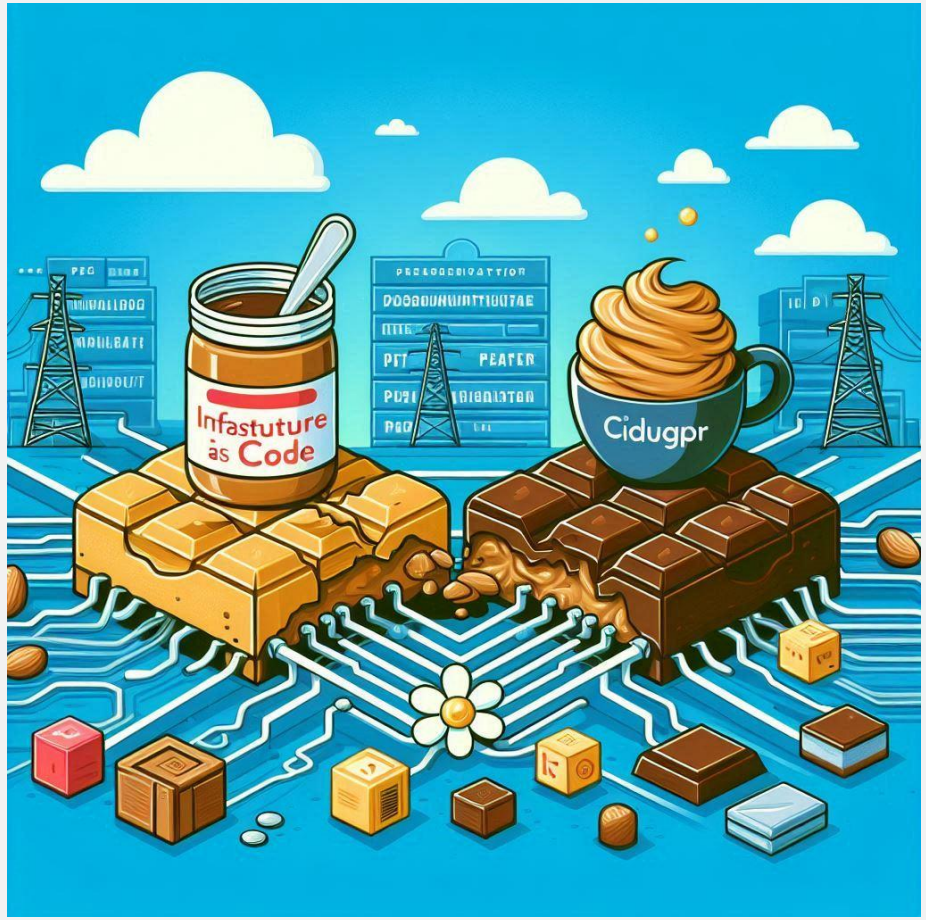
- Continuous Integration and Continuous Deployment (CI/CD)
- Version Control and Collaboration
- Automated Provisioning
- Immutable Infrastructure
- Testing and Validation

IaC and DevOps



- Continuous Integration and Continuous Deployment (CI/CD)
- Version Control and Collaboration
- Automated Provisioning
- Immutable Infrastructure
- Testing and Validation
- Monitoring and Alerts

IaC and DevOps



- Continuous Integration and Continuous Deployment (CI/CD)
- Version Control and Collaboration
- Automated Provisioning
- Immutable Infrastructure
- Testing and Validation
- Monitoring and Alerts

IaC Challenges

- Learning Curve
- Tool Proliferation
- Cloud Complexity
- Version Control
- Drift Management
- Security Concerns
- Enterprise Governance

Wizard of Provisioning: Terraform

Terraform Magic: Spells for Provisioning Cloud Resources

Terraform to the Rescue

Infrastructure as Code (IaC) tool that allows you to **build**, **modify**, and **version** cloud and on-premises resources efficiently.

With Terraform, you define infrastructure using **human-readable** configuration files, which can be **versioned**, **reused**, and **shared**.

How does Terraform work?

- Configuration Files
- Providers
- Workflow
- Immutable Infrastructure

Write

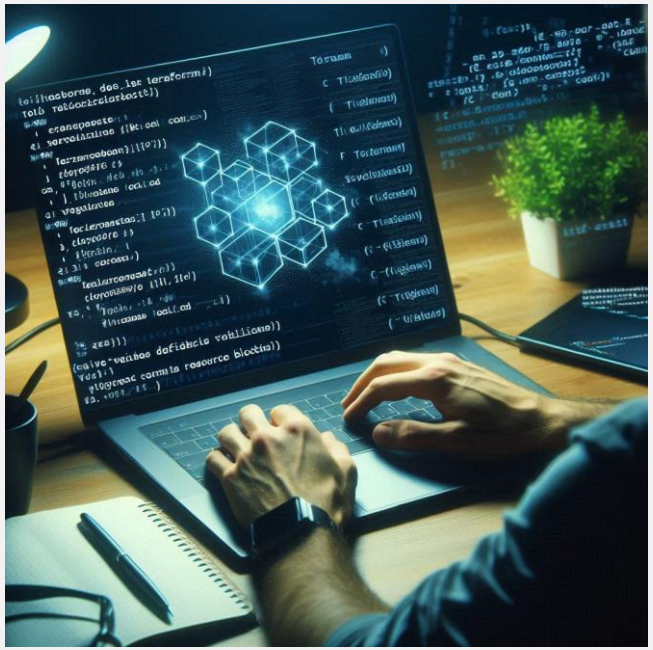
Plan

Apply

Terraform Project Structure

- .tf files
- .tfvars files
- terraform.tfstate
- .terraform directory

Terraform Elements

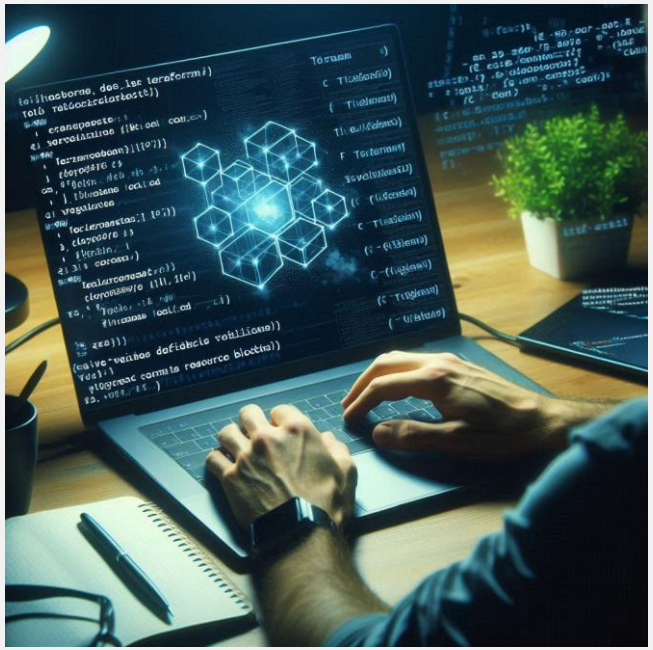


Resources

```
resource "azurerm_service_plan" "order_next_core" {
  name                = “asp-OrderNextCore-dev-usnc"
  resource_group_name = data.azurerm_resource_group.rg.name
  location            = data.azurerm_resource_group.rg.location
  os_type             = "Linux"
  sku_name            = "Y1"
  tags                = local.tags
}
```

Terraform Elements

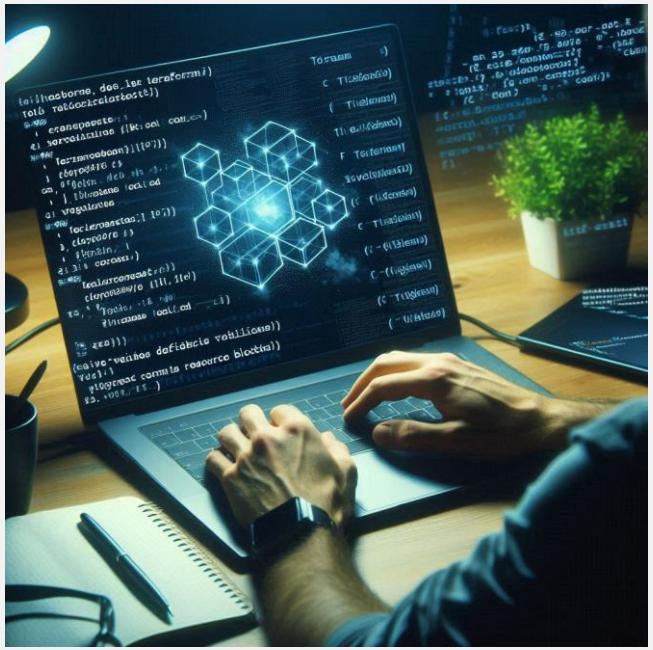
Data Sources



```
data "azurerm_resource_group" "rg" {
  name = var.resource_group_name
}
```

Terraform Elements

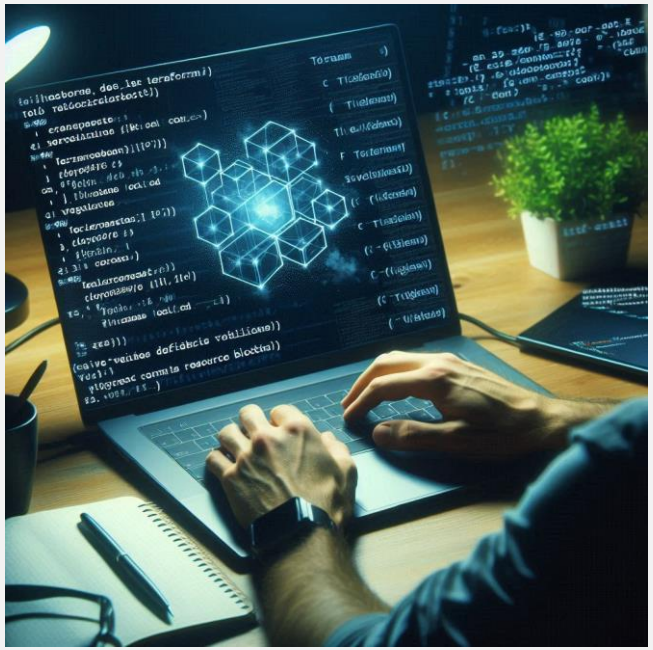
Variables



```
variable "tag_cost_center" {
  type        = string
  default     = "Remanufacturing"
  description = "Accounting cost center for the resource."
}
```

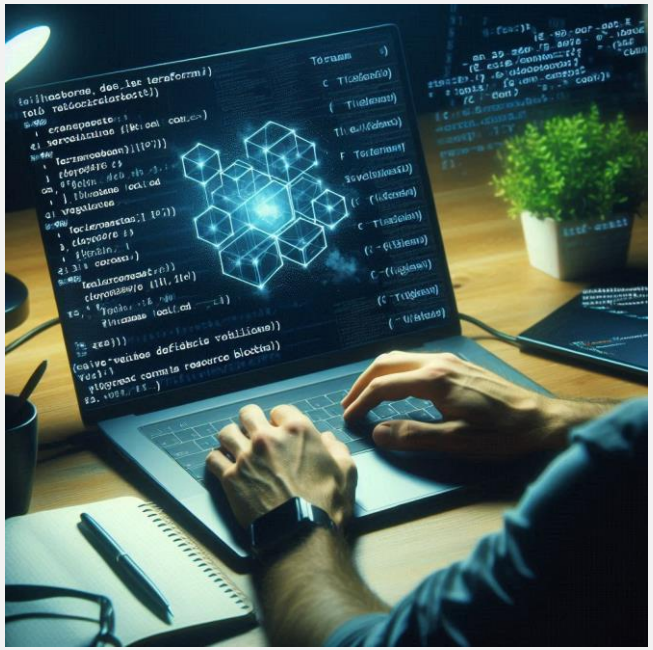
Terraform Elements

Modules



```
module "azure_regions" {
  source       = "git::https://github.com/TLC/azure-regions.git"
  azure_region = var.azure_region
}
```

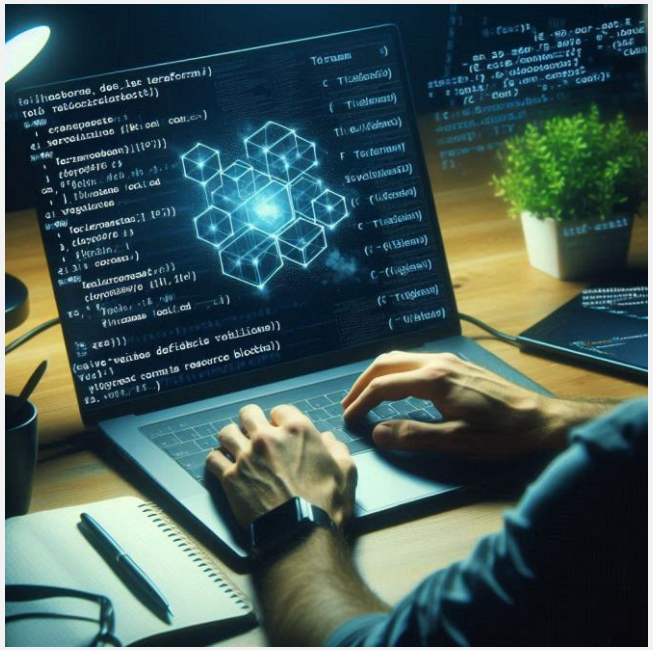
Terraform Elements



Locals

```
locals {
  tags = {
    Product      = var.tag_product
    Criticality = var.tag_criticality
    CostCenter   = var.tag_cost_center
    DR           = var.tag_disaster_recovery
    Env          = var.azure_environment
  }
}
```

Terraform Elements



Providers

```
provider "azurerm" {
  features {
    resource_group {
      prevent_deletion_if_contains_resources = true
    }
  }
}
```

Conclusion

- Infrastructure as Code and Terraform offers a modern approach to managing infrastructure, providing speed, consistency, and scalability.


EXPLORING SERVERLESS
DESIGN PATTERNS