STORAGE DEVELOPER CONFERENCE

SDC 21

BY Developers FOR Developers

Virtual Conference
September 28-29, 2021

A SNIA Event

Managing Cloud infrastructure by using Terraform - HCL

Kannadasan Palani

MSys Technologies, LLC

ABOUT
MSYS TECHNOLOGIES

PRODUCT ENGINEERING SERVICES AND DIGITAL TRANSFORMATION PARTNER

BAY AREA UNICORNS

rubrik PURESTORAGE® COHESITY Acronis

ENTERPRISES

amazon flex Google bakkt

AWARDS

DOIS

BRONZE 2021 STEVIE WINNER

IAOP
2021 Global Outsourcing 100

RED HERRING 100 WINNER ASIA

GoodFirms

IE20 INDIA EMERGING TWENTY

'TOP FASTEST GROWING STORAGE COMPANIES' FOR TWO CONSECUTIVE YEARS

Storage.com Newsletter

OUR WW STRENGTH 1000+ AND GROWING

MSYS OFFICES - USA | INDIA | VIETNAM

STORAGE DEVELOPER CONFERENCE SDC 21

Agenda

- Cloud Computing
- Cloud Infrastructure
- Terraform - Infra as a Code
- Deploy Infrastructure
- Managing Cloud infrastructure

Cloud Computing

Cloud Computing

- Cloud computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the Internet (“the cloud”)
- Faster innovation, flexible resources, and economies of scale.
- Pay only for cloud services you use, helping lower your operating costs, run your infrastructure more efficiently and scale as your business needs change
- Benefits of Cloud computing - Cost, Speed, Global scale, Performance, Reliability and Security

Cloud Infrastructure

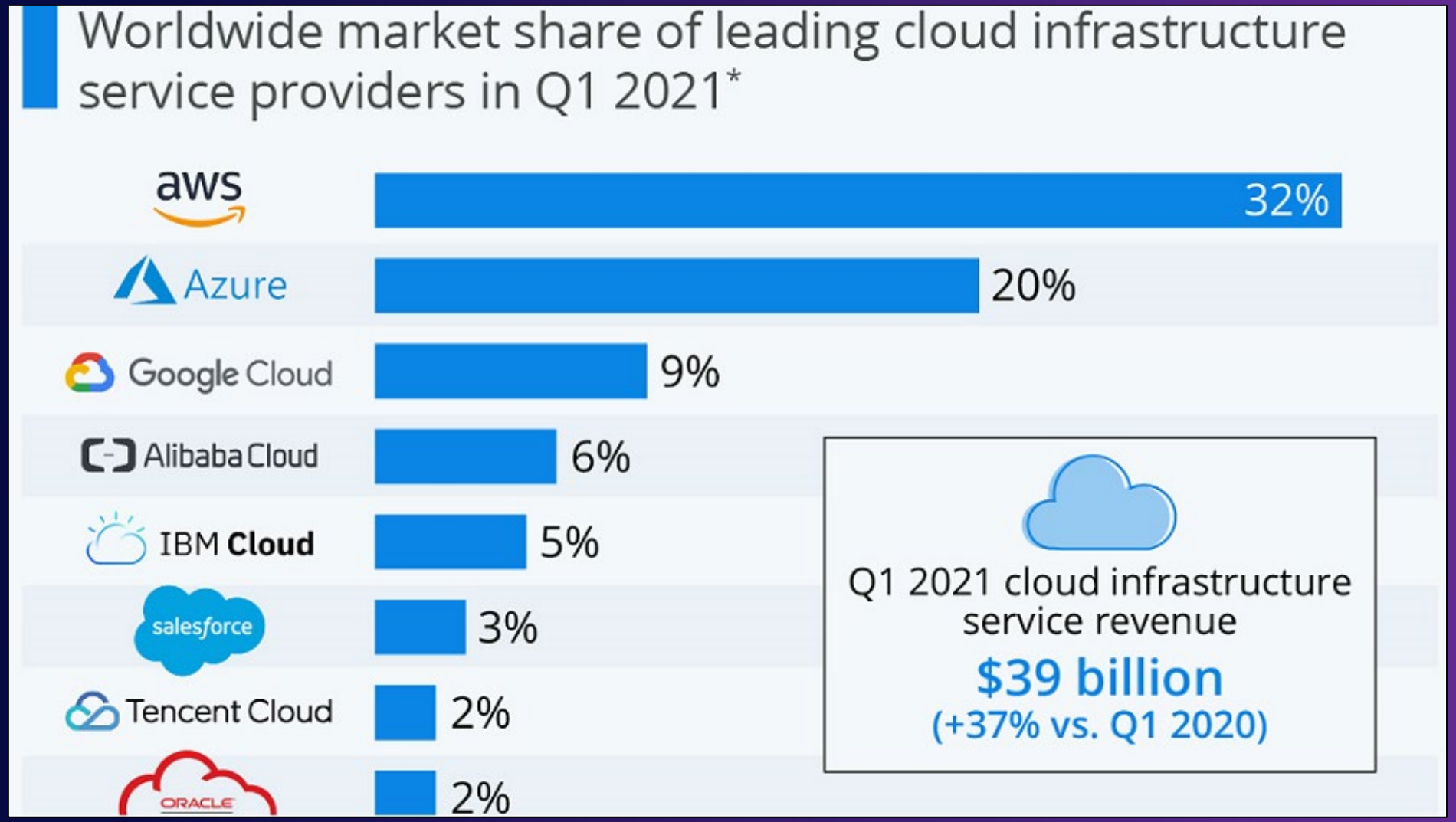
Cloud Infra

Cloud service providing by various vendors like

- AWS
- Azure
- GCP
- OCI
- Alibaba etc...

- Each having its own advanatage and disadvanatages

Cloud Market share



Cloud comparison

	Microsoft	Google	aws
Maturity	• Proven platform • Second largest provider	• Up-and-coming • Rapport from brand	• Proven platform • Market leader
Services	• Simplified services • Microsoft slant	• Open source slant • Primarily serverless	• Large number of services • Complex architecture
Administration	• Easy setup & provisioning • Microsoft-centric	• API-centric setup • Limited IaaS options	• Challenge to navigate • Lot of optionality
Integration	• Supports hybrid architecture • Interoperability is a priority	• APIs drive adoption • Less mature ETL tools	• Favors public cloud • Interoperability is a priority
AI & ML	• Proven platform • Second largest provider	• Pioneer in AI/ML space • Best APIs available	• Mature big data services • DS workbench is low traction
Cost	• Manageable cost • Light-weight options	• Minimal cost to start • Free credits emphasized	• Cost grows quickly • Cost management is a challenge
Overall	• No-brainer for MFST shop • Stable and cost-effective	• Potential to leap frog • Strength in Advanced Analytics	• Leader but costly • Significant training required

Terraform - Infra as a Code

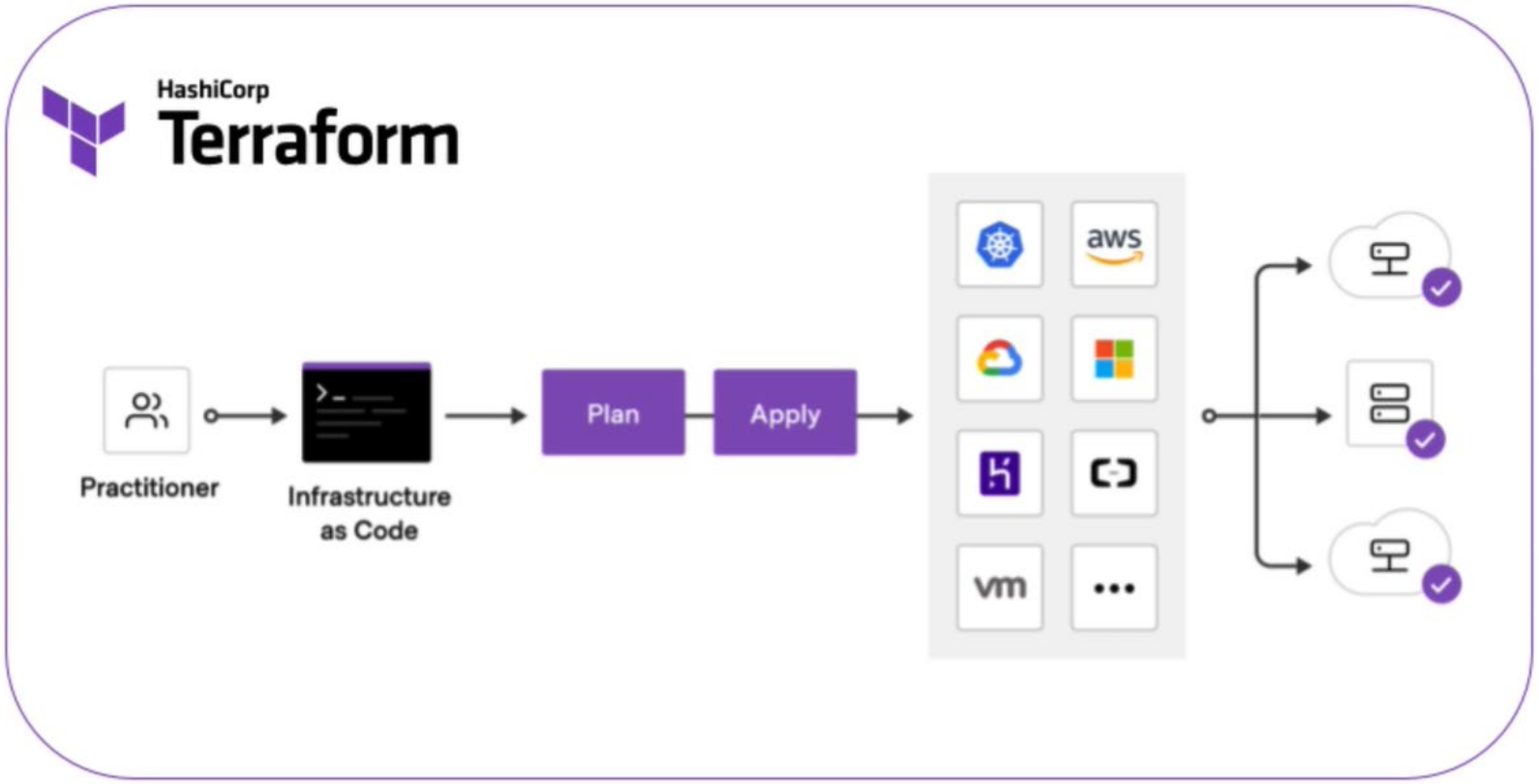
Terraform

- Terraform - infrastructure as code tool from HashiCorp for building, changing, and managing infrastructure.
- We can use it to manage Multi Cloud environments with a configuration language called the HashiCorp Configuration Language (HCL).
- It codifies cloud APIs into declarative configuration files.
- It reads configuration files and provides an execution plan of changes, which can be reviewed for safety and then applied and provisioned.

Terraform Plugins

- Terraform plugins called providers.
- Terraform interact with cloud platforms and other services via their application programming interfaces (APIs).
- HashiCorp and the Terraform community have written over 1,000 providers to manage resources on Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, and DataDog etc....
- Find providers for many of the platforms and services in the Terraform Registry.

Terraform



Terraform - Deploy infrastrucutre

- Scope - Identify the infrastructure for your project.
- Author - Write the configuration for your infrastructure.
- Initialize - Install the plugins Terraform needs to manage the infrastructure.
- Plan - Preview the changes Terraform will make to match your configuration.
- Apply - Make the planned changes.

Manage Cloud infrastructure

Manage Cloud infra

- Install Terraform
- Build Infrastructure
- Change Infrastructure
- Destroy Infrastructure

Install Terraform - CentOS/RHEL

Install `yum-config-manager` to manage your repositories.

```
$ sudo yum install -y yum-utils
```

Use `yum-config-manager` to add the official HashiCorp Linux repository.

```
$ sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo
```

Install.

```
$ sudo yum -y install terraform
```


Build infrastructure

Users > rnorwood > learn-terraform-aws-instance > main.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.27"
    }
  }
}

provider "aws" {
  profile = "default"
  region  = "us-west-2"
}

resource "aws_instance" "app_server" {
  ami           = "ami-830c94e3"
  instance_type = "t2.micro"

  tags = {
    Name = "ExampleAppServerInstance"
  }
}
```


Change infrastructure

```
 resource "aws_instance" "app_server" {
-  ami           = "ami-830c94e3"
+  ami           = "ami-08d70e59c07c61a3a"
   instance_type = "t2.micro"
 }
```

This update changes the AMI to an Ubuntu 16.04 AMI. The AWS provider knows that it cannot change the AMI of an instance after it has been created, so Terraform will destroy the old instance and create a new one.

Change infrastructure

```
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # aws_instance.app_server must be replaced
-/+ resource "aws_instance" "app_server" {
      ~ ami                          = "ami-830c94e3" -> "ami-08d70e59c07c61a3a" # forces replacement
      ~ arn                          = "arn:aws:ec2:us-west-2:561656980159:instance/i-01e03375ba238b384" ->
##...

Plan: 1 to add, 0 to change, 1 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```


Change infrastructure

```
  Enter a value: yes

aws_instance.app_server: Destroying... [id=i-01e03375ba238b384]
aws_instance.app_server: Still destroying... [id=i-01e03375ba238b384, 10s elapsed]
aws_instance.app_server: Still destroying... [id=i-01e03375ba238b384, 20s elapsed]
aws_instance.app_server: Still destroying... [id=i-01e03375ba238b384, 30s elapsed]
aws_instance.app_server: Still destroying... [id=i-01e03375ba238b384, 40s elapsed]
aws_instance.app_server: Destruction complete after 42s
aws_instance.app_server: Creating...
aws_instance.app_server: Still creating... [10s elapsed]
aws_instance.app_server: Still creating... [20s elapsed]
aws_instance.app_server: Still creating... [30s elapsed]
aws_instance.app_server: Still creating... [40s elapsed]
aws_instance.app_server: Creation complete after 50s [id=i-0fd4a35969bd21710]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
```


Destroy infrastructure

```
$ terraform destroy                                                    Copy
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.app_server will be destroyed
  - resource "aws_instance" "app_server" {
      - ami                          = "ami-08d70e59c07c61a3a" -> null
      - arn                          = "arn:aws:ec2:us-west-2:561656980159:instance/i-0fd4a35969bd21710" ->
##...

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value:
```


Destroy infrastructure

Answer `yes` to execute this plan and destroy the infrastructure.

```
  Enter a value: yes


aws_instance.app_server: Destroying... [id=i-0fd4a35969bd21710]
aws_instance.app_server: Still destroying... [id=i-0fd4a35969bd21710, 10s elapsed]
aws_instance.app_server: Still destroying... [id=i-0fd4a35969bd21710, 20s elapsed]
aws_instance.app_server: Still destroying... [id=i-0fd4a35969bd21710, 30s elapsed]
aws_instance.app_server: Destruction complete after 31s


Destroy complete! Resources: 1 destroyed.
```


Terraform - Advantages

Terraform advantages over manually managing your infrastructure:

- Terraform can manage infrastructure on multiple cloud platforms.
- The human-readable configuration language helps you write infrastructure code quickly.
- Terraform's state allows you to track resource changes throughout your deployments.
- You can commit your configurations to version control to safely collaborate on infrastructure.

About Author

- I am Kanna working as a Senior test manager with 14+ yrs of Testing experience in the Storage domain. Having very good experience in testing multiple storage vendor products
- Currently working on Cloud Storage with AI/ML projects
- Expertise in Ceph Storage, Kubernetes, and managing multi-cloud infrastructure by using Terraform.
- Also, I am an Oracle Cloud Certified practitioner and Certified Scrum master.
- www.linkedin.com/in/kdasan
- kannadasan@msystechnologies.com

Please take a moment to rate this session.

Your feedback is important to us.

Thank You

