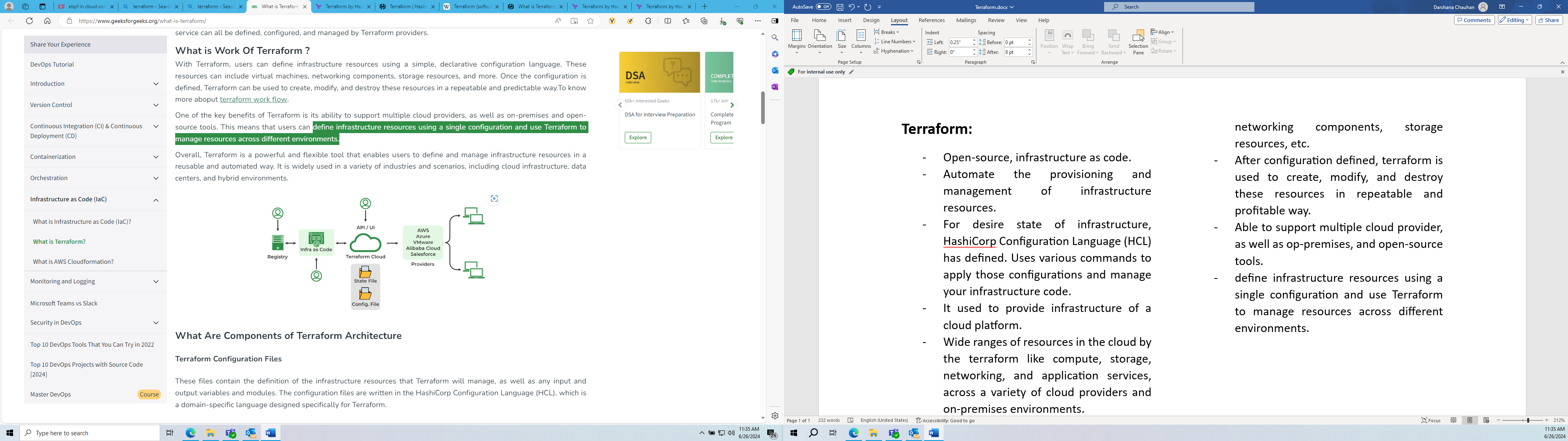
**Terraform:**

* Open-source, infrastructure as code.
* Automate the provisioning and management of infrastructure resources.
* For desire state of infrastructure, HashiCorp Configuration Language (HCL) has defined. Uses various commands to apply those configurations and manage your infrastructure code.
* It used to provide infrastructure of a cloud platform.
* Wide ranges of resources in the cloud by the terraform like compute, storage, networking, and application services, across a variety of cloud providers and on-premises environments.
* Language is declarative, just need to declare the desire state.
* Allows to define and provide data center infrastructure using a declarative configuration language.
* Key features:
  + Infrastructure as code
  + Platform Agnostic
  + State Management
  + Plan and Apply
* Use cases:
  + Provisioning Cloud Resources
  + Multi-cloud Management
  + Infrastructure versioning and Collaboration
  + Automation and CI/CD:
* Terraform Provider enables terraform to communicate with a particular infrastructure platform.
* The resource kinds and data sources that terraform handle must be implemented by providers.
* Define infrastructure resources using a simple, declarative configuration language. Which includes VMs, networking components, storage resources, etc.
* After configuration defined, terraform is used to create, modify, and destroy these resources in repeatable and profitable way.
* Able to support multiple cloud provider, as well as op-premises, and open-source tools.
* define infrastructure resources using a single configuration and use Terraform to manage resources across different environments.



* Tool for infrastructure provisioning:
  + Private network space
  + Ec2 instances
  + Install docker and other tools.
  + Security
  + Networking

All the above part will be coming under terraform in the formation of infrastructure for deploying the application.

* By replicating the initial stages of infrastructure, we can build another set up of the cluster or we can modify the existing cluster set.
* Terraform Architecture: terraform core and providers.
* Terraform core: where the user creates and configures the infrastructure.
  + Terraform State keeps up-to-date state of how the current set up of the infrastructure looks like.
  + So, core will take input and figures out the plan of what needs to be done so, it compares the current state and what is the desire configuration and as the end result it compare both result and if there is any difference then it figures out that what needs to be done to get the desire state in the configuration file.
* Providers: provides specific technologies, which could be cloud providers like AWS, Azure for IaaS platforms, Kubernetes for PaaS, Fastly for SaaS.
* Why to use terraform:
  + Multi-cloud and multi-provider support
  + Terraform is declarative management technique.
  + Mutable and immutable infrastructure
  + State management
* Advantages:
  + Declarative configuration
  + Support for multiple cloud providers
  + Reusable Infrastructure as code
  + Collaboration and version control
  + Efficient resource management
* Disadvantages:
  + Complexity
  + State management
  + Performance
  + Limited error handling
  + Limited rollback capabilities
* The core Terraform workflow consists of three stages:
  + Write: define resources, may be across multiple cloud providers and services.
  + Plan: creates an execution plan describing the infrastructure it will create, update, or destroy based on the existing infrastructure and configuration.
  + Apply: performs the proposed operations in the correct order, respecting any resource dependencies.