Objective 1: Understand Infrastructure as Code (IaC) concepts

▾ Explain What IaC is?

Infrastructure is described using a high-level configuration syntax. This allows a blueprint of our data center to be versioned and treated as we would any other code. Additionally, infrastructure can be shared and re-used.

IaC makes it easy to provision and apply infrastructure configurations, saving time. It standardizes workflows across different infrastructure providers (e.g., VMware, AWS, Azure, GCP, etc.) by using a common syntax across all of them.

It is infrastructure (CPUs, memory, disk, firewalls, etc.) defined as code within definition files.

▾ Describe advantages of IaC patterns?

- **Can be applied throughout the infrastructure lifecycle**
  - Day 0 : Initial Build
  - Day 1 : OS and application config you apply after the initial build. Includes OS updates, patches, app config.
- **Saves time by making it easy to provision and apply infrastructure configuration.** Workflow is **standardized** across providers wether its VMWare, AWS, Azure, or GCP.
- **It's easy to understand** the intent of infrastructure changes.
- **Iac makes changes idempotent**:
  - The result will always be the same since the same code is being applied
- **Iac makes changes consistent**:
  - The manual work is removed with Iac no more need for system administrators to remotely connect to each machine by executing a series of commands or scripts which can cause inconsistencies based on who executes it
- **Iac makes changes predictable**:
  - code can be tested before applying it to production so results are always predictable
- **Iac allows for mutation in previously defined configurations, making for a more manageable system**

Objective 2: Understand Terraform's purpose (vs other IaC)

▾ Explain multi-cloud and provider-agnostic benefits

Multi-cloud deployment increases fault tolerance. This means in the event of failure there is a more graceful recovery of a region or provider.

The benefits of being provider-agnostic means there can be a single configuration that manages many providers.

▼ Explain the benefits of state

- **Mapping to the Real World**
  - Terraform requires a database to map Tf(Terraform) config to the real world. ex. With state mapping Tf knows resource `resource "aws_instance" "foo"` represents instance `i-abcd34233`.
- **Metadata**
  - Tf tracks metadata or resource dependencies
  - Tf keeps a copy of the most recent set of dependencies in state. So that correct order of operations can be executed even if an item is deleted from the configuration.
- **Performance**
  - besides basic mapping Tf also keeps a cache of attribute values for all resources in the state.
  - most optional feature of state, only used to improve performance.
  - small infra: for plan and apply Tf syncs all resources in state
  - large infra: cache state is used because of API rate limits and querying all resources is too slow. Large infra also make use of `-refresh=false` and `-target` flags
- **Syncing**
  - default syncing Tf stores state in a file in the current working directory
  - for teams remote state is used, remote locking is utilized to avoid multiple people running Tf at the same time.

▼ IaC with Terraform

At a high level, Terraform allows operators to use HCL to author files containing definitions of their desired resources on almost any provider (AWS, GCP, GitHub, Docker, etc) and automates the creation of those resources at the time of apply.

- **Workflows**
  - Scope: Establish resources that need to be created for the project
  - Author: Create the configuration based on the scoped parameters with HCL
  - Initialize: run `terraform init` to download the provider plug-ins for the project
  - Plan & Apply: run `terraform plan` to verify creation then `terraform apply` to create the resources and state files

- **Advantages of Terraform**
  - Platform Agnostic: allows for management of a mixed environment with the same workflow
  - State Management: State files are created when a project is initialized. state is used to create plans and update our infrastructure. State determines how configuration changes are measured. When a change is made, those changes are compared with the state file to determine resource creation or changes
  - Operator Confidence: `terraform apply` allows for review before changes are applied.

Objective 3 ⏩

🔙README