**Q1. Terraform Basics**

A DevOps team is discussing whether Terraform overlaps with tools like Ansible and Chef. One engineer claims Terraform handles application configuration, while another argues it is mainly for provisioning infrastructure across multiple providers.

Which statement is correct about Terraform?

**Option A** – Terraform is a configuration management tool like Chef or Puppet  
**Option B** – Terraform is an Infrastructure as Code (IaC) tool that provisions resources across multiple providers  
**Option C** – Terraform is limited to AWS resources only  
**Option D** – Terraform cannot perform provisioning

✅ **Correct Answer: B**  
**Explanation:** Terraform is an IaC tool used for provisioning infrastructure across multiple providers (AWS, Azure, GCP, VMware, etc.). Unlike Chef/Ansible, its primary role is resource provisioning, not configuration management.

**Q2. Terraform Commands**

An engineer runs terraform plan to preview infrastructure changes. Once reviewed, he needs to apply those changes to provision resources in production. His colleague mistakenly suggests re-running terraform init.

Which command should the engineer use?

**Option A** – terraform plan  
**Option B** – terraform init  
**Option C** – terraform apply  
**Option D** – terraform refresh

✅ **Correct Answer: C**  
**Explanation:** terraform apply applies the planned configuration to create or update resources. init only prepares the working directory, plan previews changes, and refresh updates the state file.

**Q3. Terraform State File**

A developer accidentally deletes the terraform.tfstate file from the project directory. The deployed cloud resources are still running, but Terraform seems unable to track them.

What happens next when Terraform is used?

**Option A** – All infrastructure will be destroyed immediately  
**Option B** – Terraform will re-create all resources on the next apply  
**Option C** – Terraform loses track of existing resources, but the infrastructure still exists  
**Option D** – Nothing changes; Terraform continues normally

✅ **Correct Answer: C**  
**Explanation:** Deleting terraform.tfstate does not remove infrastructure. However, Terraform loses the mapping between resources and configuration, causing drift. Running apply may create duplicate resources unless they are re-imported.

**Q4. Missing State File Impact**

A team loses its Terraform state file due to accidental deletion. They still have the .tf configuration files but no state data. They wonder how this impacts future deployments.

What will Terraform do in this case?

**Option A** – Automatically destroy existing infrastructure  
**Option B** – Treat all resources as new and attempt to recreate them  
**Option C** – Keep infrastructure running but lose the ability to manage it  
**Option D** – Block all operations until state is restored

✅ **Correct Answer: C**  
**Explanation:** Without the state file, Terraform cannot map configuration to existing resources. The infrastructure continues running, but Terraform cannot manage it unless the resources are re-imported.

**Q5. Remote State Management**

A company wants multiple engineers to collaborate on Terraform projects. They decide to use a shared backend for state management to avoid conflicts.

Which backend is commonly used for storing state securely with locking?

**Option A** – Local backend  
**Option B** – AWS S3 with DynamoDB for locking  
**Option C** – GitHub repository  
**Option D** – Jenkins workspace

✅ **Correct Answer: B**  
**Explanation:** S3 is a popular backend for storing state files, while DynamoDB provides state locking to prevent race conditions in team environments.

**Q6. Moving Resources in State**

During refactoring, a Terraform engineer renames a resource in the configuration. Terraform detects it as a new resource, but the team wants to move the existing resource into the new name without re-creating it.

Which command should be used?

**Option A** – terraform refresh  
**Option B** – terraform state mv  
**Option C** – terraform import  
**Option D** – terraform taint

✅ **Correct Answer: B**  
**Explanation:** terraform state mv moves resources between states or changes addresses, avoiding unnecessary destruction and re-creation.

**Q7. State File Security**

A security audit reveals that Terraform state files may contain passwords and access tokens. The DevOps lead wants to secure state files in remote storage.

Which is the best practice?

**Option A** – Commit .tfstate to GitHub for versioning  
**Option B** – Encrypt state files both at rest and in transit  
**Option C** – Share .tfstate files via email for collaboration  
**Option D** – Store .tfstate only on a local machine for team use

✅ **Correct Answer: B**  
**Explanation:** Since state files may store sensitive data, they must be encrypted at rest and in transit, and stored in secure backends like S3, Azure Blob, or GCS.

**Q8. Sensitive Variables**

A Terraform pipeline manages database credentials. The team wants to ensure these credentials do not appear in CLI output or logs during execution.

How can they achieve this?

**Option A** – sensitive = true  
**Option B** – hidden = true  
**Option C** – encrypt = true  
**Option D** – mask = true

✅ **Correct Answer: A**  
**Explanation:** Terraform allows marking variables and outputs as sensitive = true, preventing them from being displayed in logs or CLI output.

**Q9. Secrets Management**

A company wants to integrate Terraform with a secure system for managing dynamic secrets like database passwords and API keys.

Which Terraform provider should they use?

**Option A** – AWS IAM  
**Option B** – Azure AD  
**Option C** – HashiCorp Vault  
**Option D** – Terraform Registry

✅ **Correct Answer: C**  
**Explanation:** The HashiCorp Vault provider enables secure secrets management in Terraform workflows, supporting dynamic secrets and integration into IaC.

**Q10. Importing Resources**

A cloud engineer manually created an S3 bucket. Later, the team decides to manage it using Terraform without destroying and re-creating it.

Which command allows linking the existing bucket to Terraform state?

**Option A** – Imports variables from another .tfvars file  
**Option B** – Imports an existing resource into Terraform state  
**Option C** – Imports Terraform modules from registry  
**Option D** – Imports provider plugins

✅ **Correct Answer: B**  
**Explanation:** terraform import connects an existing infrastructure resource to Terraform state, making it manageable without re-creation.

**Q11. Removing Resources from State**

An engineer wants to stop Terraform from managing a resource but does not want to delete the actual infrastructure.

Which command should be used?

**Option A** – terraform destroy  
**Option B** – terraform state rm  
**Option C** – terraform plan -destroy  
**Option D** – terraform refresh

✅ **Correct Answer: B**  
**Explanation:** terraform state rm removes a resource from state while leaving the infrastructure intact.

**Q12. Provisioners**

During provisioning, a Terraform configuration needs to run a script on a newly created virtual machine.

Which block should be used for this?

**Option A** – connection  
**Option B** – provisioner  
**Option C** – lifecycle  
**Option D** – backend

✅ **Correct Answer: B**  
**Explanation:** Terraform provisioners (e.g., remote-exec, local-exec, file) run scripts or transfer files after resource creation.

**Q13. Lifecycle Management**

A team is updating a resource but wants to ensure that the new version is created before destroying the old one to prevent downtime.

Which lifecycle meta-argument should they use?

**Option A** – prevent\_destroy  
**Option B** – ignore\_changes  
**Option C** – create\_before\_destroy  
**Option D** – depends\_on

✅ **Correct Answer: C**  
**Explanation:** create\_before\_destroy ensures replacement resources are provisioned before the old ones are destroyed, avoiding downtime.

**Q14. Reusable Infrastructure**

A company has standardized VPC configurations that need to be reused across multiple environments. Instead of duplicating code, they want to package the setup.

Which Terraform feature should they use?

**Option A** – Variables  
**Option B** – Providers  
**Option C** – Modules  
**Option D** – Workspaces

✅ **Correct Answer: C**  
**Explanation:** Terraform modules allow packaging reusable infrastructure components for consistent deployment across projects.

**Q15. Secrets Handling Best Practice**

A team is storing AWS access keys in their Terraform .tf files for convenience. The security lead flags this as risky and asks for a secure alternative.

What is the recommended approach?

**Option A** – Hardcode secrets in .tf files  
**Option B** – Use environment variables or a secrets manager  
**Option C** – Store secrets in terraform.tfvars without encryption  
**Option D** – Share secrets with team members in chat

✅ **Correct Answer: B**  
**Explanation:** Best practice is to store secrets in secret managers (Vault, AWS Secrets Manager, Azure Key Vault) or use environment variables. Hardcoding or storing unencrypted secrets is insecure.