TERRAFORM INFRASTRUCTURE ENGINEER PROFESSIONAL (002) · CHEAT SHEET · CERTIFYDECK

HashiCorp Certified: Terraform Infrastructure Engineer Professional (002)

Dominios, pesos oficiales y habilidades clave del temario de Terraform Infrastructure Engineer Professional (002), para repasar antes del examen.

Nivel: Advanced (Professional) Duración: 90 min Fuente: guía de estudio oficial de Microsoft Learn

1. Scalable Terraform Configuration Design

1a Translate business requirements into infrastructure code

- Traducir requisitos a código es elegir la construcción adecuada: variables con tipo, valores opcionales con optional() y validation para las reglas que deben detener el plan; for_each sobre un mapa cuando cada instancia tiene ajustes propios y count condicional para crear o no un recurso. En módulos compartidos, el prefijo de nombres debe ser una variable que decida quien llama.

1b Predictable and secure Terraform configuration

- Una configuración predecible refactoriza con bloques moved, que se conservan porque quitarlos es un cambio incompatible, y mantiene los bloques provider fuera de los módulos compartidos. La guía de estilo pide tipo y descripción en cada variable, separar entornos en estados distintos y no subir el estado ni los planes guardados; sensitive no saca un valor del estado.

1c Trace input variables, local values, and outputs to identify the source of a configuration issue

- Para encontrar el origen de un valor erróneo, sigue la cadena variable -> local -> output: la precedencia (-var y -var-file, luego *.auto.tfvars en orden léxico, terraform.tfvars, TF_VAR_ y default), la conversión de tipos (los atributos extra de un object se descartan, list(string) convierte números y bools) y el ámbito: los locales solo existen en su módulo y el padre solo ve los outputs del hijo.

1d Trace dynamic resources created with for_each

- Con for_each, cada instancia se identifica por la clave del mapa o el miembro del conjunto (recurso["clave"], o module.nombre["clave"].recurso si está en el módulo); con un conjunto, each.value es igual que each.key. Quitar una clave solo destruye esa instancia, un recurso con for_each se usa como mapa de objetos y las claves deben conocerse en el plan.

1e Predict the value of an expression that uses a Terraform function

- Para predecir una función, evalúa de dentro hacia fuera: merge da prioridad al argumento posterior, keys() y values() siguen el orden lexicográfico de las claves, zipmap usa el índice más alto para una clave repetida, slice incluye el inicio y excluye el final, element da la vuelta con el módulo, coalesce salta null y "" y try devuelve el primer argumento que no falla.

1f Implement advanced configurations

- Los bloques dynamic generan bloques anidados repetibles (nunca lifecycle ni provisioner) y las expresiones for y splat transforman colecciones; un splat sobre null da una tupla vacía. Las validaciones de variables, las precondiciones y las postcondiciones detienen la operación, mientras que los bloques check solo avisan. replace_triggered_by solo admite recursos gestionados, por eso se usa terraform_data para valores simples.

1g Configure Terraform providers

- Las configuraciones de proveedor (provider) se definen en el módulo raíz; los módulos hijos heredan solo la configuración por defecto, y las configuraciones con alias se pasan con el argumento providers y se declaran con configuration_aliases. Cada módulo declara su propio required_providers con source y versión mínima, y el archivo .terraform.lock.hcl fija solo proveedores; terraform providers lock -platform añade checksums de otras plataformas.

1h Use the Terraform MCP server to enhance development with AI coding agents

- Terraform MCP server es el servidor Model Context Protocol de HashiCorp: da a los agentes de IA documentación actual de proveedores, módulos y políticas del registro en lugar de datos de entrenamiento antiguos. Los toolsets registry-private y terraform necesitan un TFE_TOKEN válido y las operaciones destructivas, además, ENABLE_TF_OPERATIONS=true. Usa stdio en local y streamable HTTP en despliegues remotos; revisa siempre el código generado.

2. Terraform State and Backend Operations

2a Create and manage Remote Backend and Remote State Configuration

- El bloque backend no admite variables: los valores dinámicos se pasan con configuración parcial y terraform init -backend-config, y las credenciales mejor por variables de entorno. Al cambiar de backend, -migrate-state copia el estado y -reconfigure lo descarta. En S3 el bloqueo se activa con use_lockfile = true. terraform_remote_state solo expone outputs raíz y exige acceso a todo el estado; en HCP Terraform se recomienda tfe_outputs.

2b Manage state and remediate drift

- La deriva (drift) aparece cuando alguien cambia la infraestructura fuera de Terraform. terraform plan -refresh-only la muestra sin tocar nada y terraform apply -refresh-only la registra en el estado sin cambiar la infraestructura; para conservar el cambio hay que actualizar también la configuración, o el siguiente apply lo revierte. Los objetos nuevos se importan, y -refresh=false o -target pueden ocultar la deriva.

2c Identify dependency chains before upgrades

- Antes de actualizar, terraform providers muestra qué módulo declara cada requisito de proveedor y terraform graph el orden de dependencias. Terraform elige la versión más reciente que cumple todas las restricciones de raíz e hijos; el archivo de bloqueo solo fija proveedores y los módulos reutilizables deben declarar solo un mínimo. Conservar los bloques moved históricos mantiene la ruta de actualización.

2d Upgrade Terraform versions across environments

- required_version en el bloque terraform restringe solo la CLI, admite únicamente valores constantes y debe cumplirse junto con las restricciones de los módulos hijos; si no, init falla. Se recomienda ~> para fijar la versión mayor y menor y aceptar parches. Aplicar con una versión nueva actualiza terraform_version en el estado y, si hace falta, su formato, que no se puede revertir.

2e Inspect Terraform state

- terraform state list enumera direcciones y filtra por dirección, por módulo (incluidos submódulos) o por ID con -id; terraform state show muestra una sola instancia y su salida es para personas. Para automatizar se usa terraform show -json, que incluye solo los outputs del módulo raíz y muestra los valores sensibles en texto plano; en un plan, buscar "delete" en actions detecta eliminaciones y reemplazos.

3. Infrastructure Lifecycle and Operations

3a Manage and scale infrastructure projects

- Para escalar, divide los sistemas grandes en configuraciones separadas según los límites de la arquitectura, cada una con su backend, y reutiliza módulos con raíces mínimas por despliegue en lugar de workspaces de la CLI. init instala los módulos nuevos pero no cambia los instalados sin -upgrade; version solo vale para orígenes del registro y en Git se usa ref tras el subdirectorio //.

3b Plan changes to infrastructure

- terraform plan sin -out crea un plan especulativo; con -out guarda un archivo opaco, con valores sensibles en texto claro, que terraform apply ejecuta tal cual. Los modos destroy y refresh-only son excluyentes; -replace fuerza un reemplazo, -target incluye las dependencias del objetivo y es excepcional, -refresh=false ignora cambios externos y -detailed-exitcode devuelve 0, 1 o 2.

3c Apply changes to infrastructure

- Si un apply falla, Terraform registra en el estado lo que ya cambió, libera el bloqueo y termina sin revertir nada: se corrige la causa y se vuelve a aplicar. Un plan guardado con -out se aplica tal cual, sin opciones nuevas; -lock-timeout espera al bloqueo y force-unlock LOCK_ID solo libera un bloqueo propio huérfano. Los provisioners son el último recurso y on_failure y when controlan su fallo y momento.

3d Import existing infrastructure to an infrastructure project

- La importación basada en configuración usa bloques import (to, id o identity, for_each y provider) junto a un bloque resource de destino, y se previsualiza con plan antes de cambiar el estado. terraform plan -generate-config-out escribe un borrador HCL en un archivo nuevo que hay que revisar. Conviene incluir los argumentos no predeterminados, añadir a mano las relaciones y comprobar que el plan no destruya el objeto importado.

3e Manage infrastructure lifecycle

- Para dejar de gestionar un recurso sin destruirlo se usa un bloque removed con destroy = false; borrar el bloque destruye el objeto aunque tenga prevent_destroy, porque esa regla no queda en el estado. Los bloques moved renombran o reubican recursos sin reemplazarlos y conviene encadenarlos. create_before_destroy se propaga a las dependencias, ignore_changes comparte la gestión y replace_triggered_by solo admite recursos.

3f Manage multiple infrastructure environments

- Los workspaces de la CLI son estados separados dentro de un mismo directorio y backend: sirven para copias temporales, pero no aíslan entornos con credenciales distintas, para lo que se recomiendan módulos y una configuración raíz con su backend por entorno. terraform.workspace adapta nombres y tamaños; select -or-create y TF_WORKSPACE ayudan en CI, y los archivos .tfvars siguen un orden de precedencia fijo.

3g Troubleshoot deployment issues

- terraform validate detecta errores de sintaxis y coherencia, como ciclos, atributos de each inexistentes o referencias sin clave de instancia en recursos con for_each, sin consultar servicios remotos. TF_LOG, TF_LOG_CORE, TF_LOG_PROVIDER y TF_LOG_PATH generan registros para diagnosticar fallos. Los bloques timeouts amplían esperas, y state pull y push permiten reparar el estado con cuidado, respetando serial y lineage.

3h Input and output data from infrastructure projects

- terraform_remote_state solo expone los outputs del módulo raíz y exige acceso a toda la instantánea del estado; en HCP Terraform se recomienda tfe_outputs y, para otros consumidores, publicar los datos en un almacén aparte con un par recurso/data source. Las variables siguen un orden de prioridad fijo, los outputs sensitive siguen en el estado y los outputs efímeros solo existen en módulos hijos.

3i Manage provider and module versions

- El archivo .terraform.lock.hcl solo fija proveedores: init reutiliza la versión registrada y -upgrade elige la más reciente permitida; los módulos no se bloquean, así que se fijan con una versión exacta en el registro o con ?ref= en Git. Los módulos reutilizables declaran solo un mínimo (>=) y los raíz usan ~>; terraform providers lock -platform añade sumas para varias plataformas.

4. CI/CD and Automation Workflows

4a Manage CI/CD pipeline functionality

- En automatización se usa init -input=false, plan -out=tfplan y apply tfplan: el archivo de plan cuenta como aprobación, no admite más opciones y guarda datos sensibles en claro. -detailed-exitcode devuelve 0, 1 o 2, TF_IN_AUTOMATION solo cambia la salida y TF_CLI_ARGS_name añade flags a un comando. Con HCP Terraform, un plan guardado exige el modo de ejecución local; en remoto se usan planes especulativos.

4b Understand deployment strategies

- Para varios entornos se recomienda un solo backend con terraform workspace select o TF_WORKSPACE en automatización, y -backend-config si los backends difieren. Conviene un único plan pendiente y aprobar automáticamente solo infraestructura no crítica. El bloqueo del estado es automático y silencioso, force-unlock exige el ID y solo libera tu propio bloqueo, y -plugin-dir o provider_installation controlan el origen de los proveedores.

4c Manage multiple sets of dynamic provider credentials across teams, projects, and workspaces

- Las credenciales dinámicas se activan con variables de entorno TFC_* que pueden compartirse mediante conjuntos de variables con ámbito de proyecto, globales o prioritarios. Varias configuraciones del mismo proveedor en un workspace usan variables con etiqueta (_TAG), cada una con su propio token, y TFC_DEFAULT_* para valores comunes. La confianza debe comprobar al menos aud y sub.

4d Overwrite Terraform variables using auto.tfvars / .tfvars

- Terraform aplica los valores de variables por precedencia: -var y -var-file en el orden de la línea de comandos, después *.auto.tfvars en orden léxico, terraform.tfvars.json, terraform.tfvars, variables TF_VAR_ y el default. Solo se cargan solos terraform.tfvars(.json) y *.auto.tfvars(.json); el resto se pasa con -var-file. En HCP Terraform las variables del workspace sobrescriben los *.auto.tfvars.

5. Terraform Security and Governance

5a Identify best practices for secrets

- sensitive oculta valores en la CLI y en la interfaz, pero Terraform los sigue guardando en el estado y en el plan; terraform output -json o -raw los muestra en claro. Para no guardarlos se usan variables efímeras, recursos ephemeral y argumentos de solo escritura. El estado debe ir en remoto, cifrado y con control de acceso y auditoría; HYOK cifra estado y plan con una clave propia.

5b Use secret rotation and integration with secrets management tools

- Para rotar un argumento de solo escritura hay que incrementar su argumento de versión, porque Terraform no guarda ni compara el valor. Los recursos efímeros abren, renuevan y cierran el lease de un secreto de Vault en cada ejecución. ignore_changes permite que otra herramienta rote un atributo, terraform_data con replace_triggered_by fuerza una rotación y apply -refresh-only sincroniza el estado.

5c Secure CI/CD pipeline identities and access

- Las credenciales dinámicas del proveedor sustituyen las claves estáticas: HCP Terraform emite por cada plan y apply un token de identidad firmado (JWT) que la plataforma cloud o Vault verifica, comprobando al menos aud y sub con la organización, y devuelve credenciales temporales; se pueden usar roles distintos para plan y apply. Si queda un secreto estático, mejor como variable de entorno sensible o variable efímera.

5d Establish core compliance/policy rules along with Team/Project specific policies

- Las reglas básicas se publican en un conjunto de políticas global (con exclusiones por proyecto, workspace o etiqueta) y las de cada equipo en conjuntos con alcance a proyectos, workspaces o etiquetas; varios conjuntos, incluso de marcos distintos, pueden aplicarse al mismo workspace. Guardar las políticas en VCS con sentinel.hcl aporta revisión y trazabilidad, y el nivel hard-mandatory sin anulación bloquea siempre.

5e Identify and mitigate risky infrastructure changes

- Revisa los símbolos del plan (-/+ reemplazo, - destrucción) y usa -detailed-exitcode y planes guardados con -out, que contienen valores sensibles en claro. create_before_destroy cambia el orden del reemplazo y se propaga a las dependencias; prevent_destroy no protege si se borra el bloque (usa removed con destroy = false). -target, -refresh=false y -lock=false son excepcionales y arriesgados.

6. Advanced Module Usage and Registry Practices

6a Understand when to refactor a Terraform configuration into a module

- Conviene crear un módulo cuando se aprovisiona repetidamente la misma colección de recursos o cuando varios recursos forman un concepto de arquitectura; no para envolver un solo tipo de recurso. Mantén el árbol de módulos plano, sin bloques provider dentro, y al mover recursos a un módulo usa bloques moved y terraform init. Para reutilizar con versiones, publícalo en un registro.

6b Understand the contract of a Terraform module

- El contrato de un módulo son sus variables de entrada (obligatorias si no tienen default, con tipos y atributos opcionales de objeto) y sus outputs, que el padre lee como module.<NOMBRE>.<OUTPUT>. Un módulo reutilizable no contiene bloques provider: hereda las configuraciones por defecto, recibe las que tienen alias con el argumento providers y declara configuration_aliases y solo la versión mínima (>=).

6c Use the Terraform registry to build or update a configuration

- Los módulos del registro público usan <NAMESPACE>/<NAME>/<PROVIDER> y los privados anteponen el hostname (app.terraform.io o localterraform.com); // indica un subdirectorio. El argumento version solo vale para registros, y tras cambiarlo hay que ejecutar terraform init (-upgrade para avanzar dentro de la restricción): el archivo de bloqueo no fija módulos. En CI el token va en TF_TOKEN_<host>.

6d Validate modules with Terraform unit tests

- terraform test ejecuta los archivos .tftest.hcl del directorio raíz y de tests/ con estado en memoria y destruye lo creado al terminar. Los bloques run usan command = apply por defecto; command = plan da pruebas de tipo unitario. mock_provider, mock_resource y los override_* generan valores sin credenciales; expect_failures prueba condiciones que deben fallar, y las variables del archivo de pruebas tienen la máxima precedencia.

7. HCP Terraform / Enterprise Operations

7a Understand how to operate HCP Terraform

- Un workspace es un módulo raíz, un conjunto de entradas y un estado, y admite políticas, run tasks, run triggers y evaluaciones de salud; los Stacks repiten componentes en deployments y aplazan cambios dependientes. En el flujo por API se crea una versión de configuración y se sube un tar.gz con la configuración en la raíz; en workspaces con VCS solo sirve para planes especulativos.

7b Track resource usage and cost

- HCP Terraform cuenta como recurso gestionado cada instancia con mode = managed en el estado de todos los workspaces, incluidas las de count, for_each y módulos, y excluye las fuentes de datos. La estimación de costes se activa para toda la organización y aparece como fase entre el plan y el apply; se desactiva con -target y solo las policy checks de Sentinel pueden leer sus datos.

7c Implement multi-environment configuration

- Los entornos se separan en workspaces y proyectos: un conjunto de variables con ámbito de proyecto llega a todos sus workspaces actuales y futuros, una variable del workspace sobrescribe a la del conjunto y los conflictos entre conjuntos se resuelven por orden léxico del nombre. Un conjunto prioritario gana incluso a -var, pero solo una política garantiza el cumplimiento.

7d Upgrade Terraform versions across workspaces

- Cada workspace usa la versión de Terraform de sus ajustes, exacta o como restricción (con ~> toma la más reciente que la cumpla). Para actualizar: probar con un plan-only eligiendo la versión nueva, ajustar required_version y el ajuste del workspace, y lanzar Allow empty apply si el estado debe actualizarse sin cambios. Si el estado es incompatible, aplicar antes con una versión compatible.

7e Define Policy as Code in a CI/CD pipeline

- Las políticas se publican desde CI/CD conectando el policy set a un repositorio de VCS, subiendo versiones tar.gz con la API o con el proveedor tfe. sentinel.hcl debe declarar cada política y en policies.hcl de OPA el nivel por defecto es advisory. sentinel test valida las políticas en la canalización y los planes especulativos de las pull requests muestran los resultados antes de fusionar.