C:\Users\Admin\Documents\Personal\cloud infra architect\terraform>terraform init

Initializing the backend...

Initializing modules...

Initializing provider plugins...

- Reusing previous version of hashicorp/google from the dependency lock file

- Reusing previous version of hashicorp/kubernetes from the dependency lock file

- Reusing previous version of hashicorp/helm from the dependency lock file

- Using previously-installed hashicorp/google v5.34.0

- Using previously-installed hashicorp/kubernetes v2.31.0

- Using previously-installed hashicorp/helm v2.14.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see

any changes that are required for your infrastructure. All Terraform commands

should now work.

If you ever set or change modules or backend configuration for Terraform,

rerun this command to reinitialize your working directory. If you forget, other

commands will detect it and remind you to do so if necessary.

C:\Users\Admin\Documents\Personal\cloud infra architect\terraform>terraform validate

Success! The configuration is valid.

C:\Users\Admin\Documents\Personal\cloud infra architect\terraform>terraform plan

module.gke.data.google\_client\_config.default: Reading...

module.gke.data.google\_client\_config.default: Read complete after 0s

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

# module.bigtable.google\_bigtable\_instance.default will be created

+ resource "google\_bigtable\_instance" "default" {

+ deletion\_protection = true

+ display\_name = (known after apply)

+ effective\_labels = (known after apply)

+ force\_destroy = false

+ id = (known after apply)

+ instance\_type = "PRODUCTION"

+ name = "bigtable-instance"

+ project = "stylishop-ecommerce"

+ terraform\_labels = (known after apply)

+ cluster {

+ cluster\_id = "bigtable-cluster"

+ kms\_key\_name = (known after apply)

+ num\_nodes = 3

+ state = (known after apply)

+ storage\_type = "SSD"

+ zone = "asia-south1-a"

}

}

# module.cdn.google\_compute\_backend\_bucket.default will be created

+ resource "google\_compute\_backend\_bucket" "default" {

+ bucket\_name = "stylishop"

+ creation\_timestamp = (known after apply)

+ id = (known after apply)

+ name = "backend-bucket"

+ project = "stylishop-ecommerce"

+ self\_link = (known after apply)

}

# module.cdn.google\_compute\_global\_forwarding\_rule.default will be created

+ resource "google\_compute\_global\_forwarding\_rule" "default" {

+ base\_forwarding\_rule = (known after apply)

+ effective\_labels = (known after apply)

+ id = (known after apply)

+ ip\_address = (known after apply)

+ ip\_protocol = "TCP"

+ label\_fingerprint = (known after apply)

+ load\_balancing\_scheme = "EXTERNAL"

+ name = "forwarding-rule"

+ network = (known after apply)

+ port\_range = "80"

+ project = "stylishop-ecommerce"

+ psc\_connection\_id = (known after apply)

+ psc\_connection\_status = (known after apply)

+ self\_link = (known after apply)

+ subnetwork = (known after apply)

+ target = (known after apply)

+ terraform\_labels = (known after apply)

}

# module.cdn.google\_compute\_target\_http\_proxy.default will be created

+ resource "google\_compute\_target\_http\_proxy" "default" {

+ creation\_timestamp = (known after apply)

+ id = (known after apply)

+ name = "http-proxy"

+ project = "stylishop-ecommerce"

+ proxy\_bind = (known after apply)

+ proxy\_id = (known after apply)

+ self\_link = (known after apply)

+ url\_map = (known after apply)

}

# module.cdn.google\_compute\_url\_map.default will be created

+ resource "google\_compute\_url\_map" "default" {

+ creation\_timestamp = (known after apply)

+ default\_service = (known after apply)

+ fingerprint = (known after apply)

+ id = (known after apply)

+ map\_id = (known after apply)

+ name = "url-map"

+ project = "stylishop-ecommerce"

+ self\_link = (known after apply)

}

# module.cloud\_functions.google\_cloudfunctions\_function.function will be created

+ resource "google\_cloudfunctions\_function" "function" {

+ available\_memory\_mb = 256

+ docker\_registry = (known after apply)

+ effective\_labels = (known after apply)

+ entry\_point = "helloWorld"

+ https\_trigger\_security\_level = (known after apply)

+ https\_trigger\_url = (known after apply)

+ id = (known after apply)

+ ingress\_settings = "ALLOW\_ALL"

+ max\_instances = (known after apply)

+ name = "my-function"

+ project = "stylishop-ecommerce"

+ region = "asia-south1"

+ runtime = "nodejs14"

+ service\_account\_email = (known after apply)

+ source\_archive\_bucket = "stylishop"

+ source\_archive\_object = "function-source.zip"

+ status = (known after apply)

+ terraform\_labels = (known after apply)

+ timeout = 60

+ trigger\_http = true

+ version\_id = (known after apply)

+ vpc\_connector = "connector"

+ vpc\_connector\_egress\_settings = (known after apply)

}

# module.cloud\_functions.google\_vpc\_access\_connector.access will be created

+ resource "google\_vpc\_access\_connector" "access" {

+ connected\_projects = (known after apply)

+ id = (known after apply)

+ machine\_type = "e2-micro"

+ max\_instances = (known after apply)

+ max\_throughput = 300

+ min\_instances = (known after apply)

+ min\_throughput = 200

+ name = "connector"

+ network = (known after apply)

+ project = "stylishop-ecommerce"

+ region = "asia-south1"

+ self\_link = (known after apply)

+ state = (known after apply)

}

# module.cloud\_sql.google\_sql\_database\_instance.default will be created

+ resource "google\_sql\_database\_instance" "default" {

+ available\_maintenance\_versions = (known after apply)

+ connection\_name = (known after apply)

+ database\_version = "MYSQL\_5\_7"

+ deletion\_protection = true

+ dns\_name = (known after apply)

+ encryption\_key\_name = (known after apply)

+ first\_ip\_address = (known after apply)

+ id = (known after apply)

+ instance\_type = (known after apply)

+ ip\_address = (known after apply)

+ maintenance\_version = (known after apply)

+ master\_instance\_name = (known after apply)

+ name = "sql-instance"

+ private\_ip\_address = (known after apply)

+ project = "stylishop-ecommerce"

+ psc\_service\_attachment\_link = (known after apply)

+ public\_ip\_address = (known after apply)

+ region = "asia-south1"

+ self\_link = (known after apply)

+ server\_ca\_cert = (sensitive value)

+ service\_account\_email\_address = (known after apply)

+ settings {

+ activation\_policy = "ALWAYS"

+ availability\_type = "ZONAL"

+ connector\_enforcement = (known after apply)

+ disk\_autoresize = true

+ disk\_autoresize\_limit = 0

+ disk\_size = (known after apply)

+ disk\_type = "PD\_SSD"

+ edition = "ENTERPRISE"

+ pricing\_plan = "PER\_USE"

+ tier = "db-f1-micro"

+ user\_labels = (known after apply)

+ version = (known after apply)

+ ip\_configuration {

+ ipv4\_enabled = true

+ private\_network = (known after apply)

+ ssl\_mode = (known after apply)

}

}

}

# module.cloud\_storage.google\_storage\_bucket.default will be created

+ resource "google\_storage\_bucket" "default" {

+ effective\_labels = (known after apply)

+ force\_destroy = false

+ id = (known after apply)

+ location = "ASIA-SOUTH1"

+ name = "stylishop"

+ project = "stylishop-ecommerce"

+ project\_number = (known after apply)

+ public\_access\_prevention = (known after apply)

+ rpo = (known after apply)

+ self\_link = (known after apply)

+ storage\_class = "STANDARD"

+ terraform\_labels = (known after apply)

+ uniform\_bucket\_level\_access = (known after apply)

+ url = (known after apply)

}

# module.compute\_instance.google\_compute\_autoscaler.default will be created

+ resource "google\_compute\_autoscaler" "default" {

+ creation\_timestamp = (known after apply)

+ id = (known after apply)

+ name = "nginx-autoscaler"

+ project = "stylishop-ecommerce"

+ self\_link = (known after apply)

+ target = (known after apply)

+ zone = "asia-south1-a"

+ autoscaling\_policy {

+ cooldown\_period = 60

+ max\_replicas = 10

+ min\_replicas = 3

+ mode = "ON"

+ cpu\_utilization {

+ predictive\_method = "NONE"

+ target = 0.6

}

}

}

# module.compute\_instance.google\_compute\_health\_check.default will be created

+ resource "google\_compute\_health\_check" "default" {

+ check\_interval\_sec = 5

+ creation\_timestamp = (known after apply)

+ healthy\_threshold = 2

+ id = (known after apply)

+ name = "nginx-health-check"

+ project = "stylishop-ecommerce"

+ self\_link = (known after apply)

+ timeout\_sec = 5

+ type = (known after apply)

+ unhealthy\_threshold = 2

+ tcp\_health\_check {

+ port = 80

+ proxy\_header = "NONE"

}

}

# module.compute\_instance.google\_compute\_instance\_group\_manager.default will be created

+ resource "google\_compute\_instance\_group\_manager" "default" {

+ base\_instance\_name = "nginx-instance"

+ creation\_timestamp = (known after apply)

+ fingerprint = (known after apply)

+ id = (known after apply)

+ instance\_group = (known after apply)

+ list\_managed\_instances\_results = "PAGELESS"

+ name = "nginx-instance-group"

+ operation = (known after apply)

+ project = "stylishop-ecommerce"

+ self\_link = (known after apply)

+ status = (known after apply)

+ target\_size = 3

+ wait\_for\_instances = false

+ wait\_for\_instances\_status = "STABLE"

+ zone = "asia-south1-a"

+ auto\_healing\_policies {

+ health\_check = (known after apply)

+ initial\_delay\_sec = 300

}

+ named\_port {

+ name = "http"

+ port = 80

}

+ version {

+ instance\_template = (known after apply)

}

}

# module.compute\_instance.google\_compute\_instance\_template.default will be created

+ resource "google\_compute\_instance\_template" "default" {

+ can\_ip\_forward = false

+ effective\_labels = (known after apply)

+ id = (known after apply)

+ machine\_type = "e2-medium"

+ metadata\_fingerprint = (known after apply)

+ metadata\_startup\_script = <<-EOT

#!/bin/bash

sudo apt-get update

sudo apt-get install -y nginx

sudo systemctl start nginx

sudo systemctl enable nginx

EOT

+ name = "nginx-template"

+ name\_prefix = (known after apply)

+ project = "stylishop-ecommerce"

+ region = (known after apply)

+ self\_link = (known after apply)

+ self\_link\_unique = (known after apply)

+ tags\_fingerprint = (known after apply)

+ terraform\_labels = (known after apply)

+ disk {

+ auto\_delete = true

+ boot = true

+ device\_name = (known after apply)

+ disk\_size\_gb = (known after apply)

+ disk\_type = (known after apply)

+ interface = (known after apply)

+ mode = (known after apply)

+ provisioned\_iops = (known after apply)

+ source\_image = "debian-cloud/debian-10"

+ type = (known after apply)

}

+ network\_interface {

+ internal\_ipv6\_prefix\_length = (known after apply)

+ ipv6\_access\_type = (known after apply)

+ ipv6\_address = (known after apply)

+ name = (known after apply)

+ network = (known after apply)

+ stack\_type = (known after apply)

+ subnetwork = (known after apply)

+ subnetwork\_project = (known after apply)

}

}

# module.gke.google\_container\_cluster.primary will be created

+ resource "google\_container\_cluster" "primary" {

+ cluster\_ipv4\_cidr = (known after apply)

+ datapath\_provider = (known after apply)

+ default\_max\_pods\_per\_node = (known after apply)

+ deletion\_protection = true

+ enable\_cilium\_clusterwide\_network\_policy = false

+ enable\_intranode\_visibility = (known after apply)

+ enable\_kubernetes\_alpha = false

+ enable\_l4\_ilb\_subsetting = false

+ enable\_legacy\_abac = false

+ enable\_shielded\_nodes = true

+ endpoint = (known after apply)

+ id = (known after apply)

+ label\_fingerprint = (known after apply)

+ location = "asia-south1"

+ logging\_service = (known after apply)

+ master\_version = (known after apply)

+ monitoring\_service = (known after apply)

+ name = "gke-cluster"

+ network = "default"

+ networking\_mode = (known after apply)

+ node\_locations = (known after apply)

+ node\_version = (known after apply)

+ operation = (known after apply)

+ private\_ipv6\_google\_access = (known after apply)

+ project = (known after apply)

+ remove\_default\_node\_pool = true

+ self\_link = (known after apply)

+ services\_ipv4\_cidr = (known after apply)

+ subnetwork = (known after apply)

+ tpu\_ipv4\_cidr\_block = (known after apply)

+ master\_auth {

+ client\_certificate = (known after apply)

+ client\_key = (sensitive value)

+ cluster\_ca\_certificate = (known after apply)

+ client\_certificate\_config {

+ issue\_client\_certificate = false

}

}

+ workload\_identity\_config {

+ workload\_pool = "stylishop-ecommerce.svc.id.goog"

}

}

# module.gke.google\_container\_node\_pool.primary\_nodes will be created

+ resource "google\_container\_node\_pool" "primary\_nodes" {

+ cluster = "gke-cluster"

+ id = (known after apply)

+ initial\_node\_count = (known after apply)

+ instance\_group\_urls = (known after apply)

+ location = "asia-south1"

+ managed\_instance\_group\_urls = (known after apply)

+ max\_pods\_per\_node = (known after apply)

+ name = (known after apply)

+ name\_prefix = (known after apply)

+ node\_count = 1

+ node\_locations = (known after apply)

+ operation = (known after apply)

+ project = "stylishop-ecommerce"

+ version = (known after apply)

+ autoscaling {

+ location\_policy = (known after apply)

+ max\_node\_count = 5

+ min\_node\_count = 1

}

+ node\_config {

+ disk\_size\_gb = (known after apply)

+ disk\_type = (known after apply)

+ effective\_taints = (known after apply)

+ guest\_accelerator = (known after apply)

+ image\_type = (known after apply)

+ labels = (known after apply)

+ local\_ssd\_count = (known after apply)

+ logging\_variant = (known after apply)

+ machine\_type = "e2-medium"

+ metadata = (known after apply)

+ min\_cpu\_platform = (known after apply)

+ oauth\_scopes = [

+ "https://www.googleapis.com/auth/cloud-platform",

]

+ preemptible = false

+ service\_account = (known after apply)

+ spot = false

+ workload\_metadata\_config {

+ mode = "GKE\_METADATA"

}

}

}

# module.gke.google\_project\_service.container will be created

+ resource "google\_project\_service" "container" {

+ disable\_on\_destroy = true

+ id = (known after apply)

+ project = "stylishop-ecommerce"

+ service = "container.googleapis.com"

}

# module.gke.helm\_release.argocd will be created

+ resource "helm\_release" "argocd" {

+ atomic = false

+ chart = "argo-cd"

+ cleanup\_on\_fail = false

+ create\_namespace = false

+ dependency\_update = false

+ disable\_crd\_hooks = false

+ disable\_openapi\_validation = false

+ disable\_webhooks = false

+ force\_update = false

+ id = (known after apply)

+ lint = false

+ manifest = (known after apply)

+ max\_history = 0

+ metadata = (known after apply)

+ name = "argocd"

+ namespace = "argocd"

+ pass\_credentials = false

+ recreate\_pods = false

+ render\_subchart\_notes = true

+ replace = false

+ repository = "https://argoproj.github.io/argo-helm"

+ reset\_values = false

+ reuse\_values = false

+ skip\_crds = false

+ status = "deployed"

+ timeout = 300

+ verify = false

+ version = "3.8.2"

+ wait = true

+ wait\_for\_jobs = false

+ set {

+ name = "configs.cm.apiVersion"

+ value = "argoproj.io/v1alpha1"

}

+ set {

+ name = "server.service.type"

+ value = "LoadBalancer"

}

}

# module.gke.helm\_release.prometheus will be created

+ resource "helm\_release" "prometheus" {

+ atomic = false

+ chart = "kube-prometheus-stack"

+ cleanup\_on\_fail = false

+ create\_namespace = false

+ dependency\_update = false

+ disable\_crd\_hooks = false

+ disable\_openapi\_validation = false

+ disable\_webhooks = false

+ force\_update = false

+ id = (known after apply)

+ lint = false

+ manifest = (known after apply)

+ max\_history = 0

+ metadata = (known after apply)

+ name = "prometheus"

+ namespace = "monitoring"

+ pass\_credentials = false

+ recreate\_pods = false

+ render\_subchart\_notes = true

+ replace = false

+ repository = "https://prometheus-community.github.io/helm-charts"

+ reset\_values = false

+ reuse\_values = false

+ skip\_crds = false

+ status = "deployed"

+ timeout = 300

+ verify = false

+ version = "15.4.6"

+ wait = true

+ wait\_for\_jobs = false

+ set {

+ name = "grafana.enabled"

+ value = "true"

}

+ set {

+ name = "grafana.service.type"

+ value = "LoadBalancer"

}

}

# module.gke.kubernetes\_namespace.argocd will be created

+ resource "kubernetes\_namespace" "argocd" {

+ id = (known after apply)

+ wait\_for\_default\_service\_account = false

+ metadata {

+ generation = (known after apply)

+ name = "argocd"

+ resource\_version = (known after apply)

+ uid = (known after apply)

}

}

# module.gke.kubernetes\_namespace.monitoring will be created

+ resource "kubernetes\_namespace" "monitoring" {

+ id = (known after apply)

+ wait\_for\_default\_service\_account = false

+ metadata {

+ generation = (known after apply)

+ name = "monitoring"

+ resource\_version = (known after apply)

+ uid = (known after apply)

}

}

# module.load\_balancer.google\_compute\_backend\_service.default will be created

+ resource "google\_compute\_backend\_service" "default" {

+ connection\_draining\_timeout\_sec = 300

+ creation\_timestamp = (known after apply)

+ fingerprint = (known after apply)

+ generated\_id = (known after apply)

+ health\_checks = (known after apply)

+ id = (known after apply)

+ load\_balancing\_scheme = "EXTERNAL"

+ name = "backend-service"

+ port\_name = "http"

+ project = "stylishop-ecommerce"

+ protocol = "HTTP"

+ self\_link = (known after apply)

+ session\_affinity = (known after apply)

+ timeout\_sec = (known after apply)

+ backend {

+ balancing\_mode = "UTILIZATION"

+ capacity\_scaler = 1

+ group = (known after apply)

+ max\_connections = (known after apply)

+ max\_connections\_per\_endpoint = (known after apply)

+ max\_connections\_per\_instance = (known after apply)

+ max\_rate = (known after apply)

+ max\_rate\_per\_endpoint = (known after apply)

+ max\_rate\_per\_instance = (known after apply)

+ max\_utilization = (known after apply)

}

}

# module.load\_balancer.google\_compute\_forwarding\_rule.default will be created

+ resource "google\_compute\_forwarding\_rule" "default" {

+ base\_forwarding\_rule = (known after apply)

+ creation\_timestamp = (known after apply)

+ effective\_labels = (known after apply)

+ id = (known after apply)

+ ip\_address = (known after apply)

+ ip\_protocol = "TCP"

+ ip\_version = (known after apply)

+ label\_fingerprint = (known after apply)

+ load\_balancing\_scheme = "EXTERNAL"

+ name = "forwarding-rule"

+ network = (known after apply)

+ network\_tier = (known after apply)

+ port\_range = "80"

+ project = "stylishop-ecommerce"

+ psc\_connection\_id = (known after apply)

+ psc\_connection\_status = (known after apply)

+ recreate\_closed\_psc = false

+ region = (known after apply)

+ self\_link = (known after apply)

+ service\_name = (known after apply)

+ subnetwork = (known after apply)

+ target = (known after apply)

+ terraform\_labels = (known after apply)

}

# module.load\_balancer.google\_compute\_global\_address.default will be created

+ resource "google\_compute\_global\_address" "default" {

+ address = (known after apply)

+ creation\_timestamp = (known after apply)

+ id = (known after apply)

+ name = "global-ip"

+ prefix\_length = (known after apply)

+ project = "stylishop-ecommerce"

+ self\_link = (known after apply)

}

# module.load\_balancer.google\_compute\_http\_health\_check.default will be created

+ resource "google\_compute\_http\_health\_check" "default" {

+ check\_interval\_sec = 5

+ creation\_timestamp = (known after apply)

+ healthy\_threshold = 2

+ id = (known after apply)

+ name = "http-health-check"

+ port = 80

+ project = "stylishop-ecommerce"

+ request\_path = "/"

+ self\_link = (known after apply)

+ timeout\_sec = 5

+ unhealthy\_threshold = 2

}

# module.load\_balancer.google\_compute\_target\_http\_proxy.default will be created

+ resource "google\_compute\_target\_http\_proxy" "default" {

+ creation\_timestamp = (known after apply)

+ id = (known after apply)

+ name = "http-proxy"

+ project = "stylishop-ecommerce"

+ proxy\_bind = (known after apply)

+ proxy\_id = (known after apply)

+ self\_link = (known after apply)

+ url\_map = (known after apply)

}

# module.load\_balancer.google\_compute\_url\_map.default will be created

+ resource "google\_compute\_url\_map" "default" {

+ creation\_timestamp = (known after apply)

+ default\_service = (known after apply)

+ fingerprint = (known after apply)

+ id = (known after apply)

+ map\_id = (known after apply)

+ name = "url-map"

+ project = "stylishop-ecommerce"

+ self\_link = (known after apply)

}

# module.memorystore.google\_redis\_instance.default will be created

+ resource "google\_redis\_instance" "default" {

+ alternative\_location\_id = (known after apply)

+ auth\_enabled = false

+ auth\_string = (sensitive value)

+ authorized\_network = (known after apply)

+ connect\_mode = "DIRECT\_PEERING"

+ create\_time = (known after apply)

+ current\_location\_id = (known after apply)

+ effective\_labels = (known after apply)

+ host = (known after apply)

+ id = (known after apply)

+ location\_id = (known after apply)

+ maintenance\_schedule = (known after apply)

+ memory\_size\_gb = 1

+ name = "redis-instance"

+ nodes = (known after apply)

+ persistence\_iam\_identity = (known after apply)

+ port = (known after apply)

+ project = "stylishop-ecommerce"

+ read\_endpoint = (known after apply)

+ read\_endpoint\_port = (known after apply)

+ read\_replicas\_mode = (known after apply)

+ redis\_version = (known after apply)

+ region = "asia-south1"

+ replica\_count = (known after apply)

+ reserved\_ip\_range = (known after apply)

+ secondary\_ip\_range = (known after apply)

+ server\_ca\_certs = (known after apply)

+ terraform\_labels = (known after apply)

+ tier = "STANDARD\_HA"

+ transit\_encryption\_mode = "DISABLED"

}

# module.vpc.google\_compute\_firewall.allow\_external will be created

+ resource "google\_compute\_firewall" "allow\_external" {

+ creation\_timestamp = (known after apply)

+ destination\_ranges = (known after apply)

+ direction = (known after apply)

+ enable\_logging = (known after apply)

+ id = (known after apply)

+ name = "allow-external"

+ network = "stylishop"

+ priority = 1000

+ project = "stylishop-ecommerce"

+ self\_link = (known after apply)

+ source\_ranges = [

+ "0.0.0.0/0",

]

+ allow {

+ ports = [

+ "80",

+ "443",

]

+ protocol = "tcp"

}

}

# module.vpc.google\_compute\_firewall.allow\_internal will be created

+ resource "google\_compute\_firewall" "allow\_internal" {

+ creation\_timestamp = (known after apply)

+ destination\_ranges = (known after apply)

+ direction = (known after apply)

+ enable\_logging = (known after apply)

+ id = (known after apply)

+ name = "allow-internal"

+ network = "stylishop"

+ priority = 1000

+ project = "stylishop-ecommerce"

+ self\_link = (known after apply)

+ source\_ranges = [

+ "10.128.0.0/9",

]

+ allow {

+ ports = [

+ "0-65535",

]

+ protocol = "tcp"

}

+ allow {

+ ports = [

+ "0-65535",

]

+ protocol = "udp"

}

+ allow {

+ ports = []

+ protocol = "icmp"

}

}

# module.vpc.google\_compute\_firewall.allow\_ssh will be created

+ resource "google\_compute\_firewall" "allow\_ssh" {

+ creation\_timestamp = (known after apply)

+ destination\_ranges = (known after apply)

+ direction = (known after apply)

+ enable\_logging = (known after apply)

+ id = (known after apply)

+ name = "allow-ssh"

+ network = "stylishop"

+ priority = 1000

+ project = "stylishop-ecommerce"

+ self\_link = (known after apply)

+ source\_ranges = [

+ "10.128.0.0/9",

]

+ allow {

+ ports = [

+ "22",

]

+ protocol = "tcp"

}

}

# module.vpc.google\_compute\_network.vpc will be created

+ resource "google\_compute\_network" "vpc" {

+ auto\_create\_subnetworks = false

+ delete\_default\_routes\_on\_create = false

+ gateway\_ipv4 = (known after apply)

+ id = (known after apply)

+ internal\_ipv6\_range = (known after apply)

+ mtu = (known after apply)

+ name = "stylishop"

+ network\_firewall\_policy\_enforcement\_order = "AFTER\_CLASSIC\_FIREWALL"

+ numeric\_id = (known after apply)

+ project = "stylishop-ecommerce"

+ routing\_mode = (known after apply)

+ self\_link = (known after apply)

}

# module.vpc.google\_compute\_subnetwork.private will be created

+ resource "google\_compute\_subnetwork" "private" {

+ creation\_timestamp = (known after apply)

+ external\_ipv6\_prefix = (known after apply)

+ fingerprint = (known after apply)

+ gateway\_address = (known after apply)

+ id = (known after apply)

+ internal\_ipv6\_prefix = (known after apply)

+ ip\_cidr\_range = "10.0.2.0/24"

+ ipv6\_cidr\_range = (known after apply)

+ name = "stylishop-private"

+ network = "stylishop"

+ private\_ip\_google\_access = (known after apply)

+ private\_ipv6\_google\_access = (known after apply)

+ project = "stylishop-ecommerce"

+ purpose = (known after apply)

+ region = "asia-south1"

+ secondary\_ip\_range = (known after apply)

+ self\_link = (known after apply)

+ stack\_type = (known after apply)

}

# module.vpc.google\_compute\_subnetwork.public will be created

+ resource "google\_compute\_subnetwork" "public" {

+ creation\_timestamp = (known after apply)

+ external\_ipv6\_prefix = (known after apply)

+ fingerprint = (known after apply)

+ gateway\_address = (known after apply)

+ id = (known after apply)

+ internal\_ipv6\_prefix = (known after apply)

+ ip\_cidr\_range = "10.0.1.0/24"

+ ipv6\_cidr\_range = (known after apply)

+ name = "stylishop-public"

+ network = "stylishop"

+ private\_ip\_google\_access = (known after apply)

+ private\_ipv6\_google\_access = (known after apply)

+ project = "stylishop-ecommerce"

+ purpose = (known after apply)

+ region = "asia-south1"

+ secondary\_ip\_range = (known after apply)

+ self\_link = (known after apply)

+ stack\_type = (known after apply)

}

Plan: 33 to add, 0 to change, 0 to destroy.

Changes to Outputs:

+ vpc\_id = (known after apply)

──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

C:\Users\Admin\Documents\Personal\cloud infra architect\terraform>