
terraform-example-foundation

This example repository shows how the CFT Terraform modules can build a secure Google Cloud foundation, following the Google Cloud Enterprise Foundations Blueprint (previously called the *Security Foundations Guide*). The supplied structure and code is intended to form a starting point for building your own foundation with pragmatic defaults that you can customize to meet your own requirements. Currently, the step 0 is manually executed. From step 1 onwards, the Terraform code is deployed by using either Google Cloud Build (default) or Jenkins. Cloud Build has been chosen by default to allow you to quickly get started without having to deploy a CI/CD tool, although it is worth noting the code can easily be executed by your preferred tool.

Overview

This repo contains several distinct Terraform projects, each within their own directory that must be applied separately, but in sequence. Each of these Terraform projects are to be layered on top of each other, and run in the following order.

0. bootstrap

This stage executes the CFT Bootstrap module which bootstraps an existing Google Cloud organization, creating all the required Google Cloud resources and permissions to start using the Cloud Foundation Toolkit (CFT). For CI/CD Pipelines, you can use either Cloud Build (by default) or Jenkins. If you want to use Jenkins instead of Cloud Build, see README-Jenkins on how to use the Jenkins submodule.

The bootstrap step includes:

- The `prj-b-seed` project that contains the following:
 - Terraform state bucket
 - Custom service accounts used by Terraform to create new resources in Google Cloud
- The `prj-b-cicd` project that contains the following:
 - A CI/CD Pipeline implemented with either Cloud Build or Jenkins
 - If using Cloud Build, the following items:
 - ★ Cloud Source Repository
 - ★ Artifact Registry
 - If using Jenkins, the following items:
 - ★ A Compute Engine instance configured as a Jenkins Agent

-
- ★ Custom service account to run Compute Engine instances for Jenkins Agents
 - ★ VPN connection with on-prem (or wherever your Jenkins Controller is located)

It is a best practice to separate concerns by having two projects here: one for the Terraform state and one for the CI/CD tool. - The `prj-b-seed` project stores Terraform state and has the service accounts that can create or modify infrastructure. - The `prj-b-cicd` project holds the CI/CD tool (either Cloud Build or Jenkins) that coordinates the infrastructure deployment.

To further separate the concerns at the IAM level as well, a distinct service account is created for each stage. The Terraform custom service accounts are granted the IAM permissions required to build the foundation. If using Cloud Build as the CI/CD tool, these service accounts are used directly in the pipeline to execute the pipeline steps (`plan` or `apply`). In this configuration, the baseline permissions of the CI/CD tool are unchanged.

If using Jenkins as the CI/CD tool, the service account of the Jenkins Agent (`sa-jenkins-agent-gce@prj-b-cicd-xxxx.iam.gserviceaccount.com`) is granted impersonation access so it can generate tokens over the Terraform custom Service Accounts. In this configuration, the baseline permissions of the CI/CD tool are limited.

After executing this step, you will have the following structure:

| | | |
|---|------------------------------------|----|
| 1 | <code>example-organization/</code> | └─ |
| 2 | <code>fldr-bootstrap</code> | └─ |
| 3 | <code>prj-b-cicd</code> | └─ |
| 4 | <code>prj-b-seed</code> | |

When this step uses the Cloud Build submodule, it sets up the `cicd` project (`prj-b-cicd`) with Cloud Build and Cloud Source Repositories for each of the stages below. Triggers are configured to run a `terraform plan` for any non-environment branch and `terraform apply` when changes are merged to an environment branch (`development`, `nonproduction` or `production`). Usage instructions are available in the 0-bootstrap README.

1. org

The purpose of this stage is to set up the common folder used to house projects that contain shared resources such as Security Command Center notification, Cloud Key Management Service (KMS), org level secrets, and org level logging. This stage also sets up the network folder used to house network related projects such as DNS Hub, Interconnect, network hub, and base and restricted projects for each environment (`development`, `nonproduction` or `production`). This will create the following folder and project structure:

| | | |
|---|-----------------------------------|----|
| 1 | <code>example-organization</code> | └─ |
|---|-----------------------------------|----|

| | | |
|----|--------------------------|---|
| 2 | fldr-common | — |
| 3 | prj-c-logging | — |
| 4 | prj-c-billing-logs | — |
| 5 | prj-c-scc | — |
| 6 | prj-c-kms | — |
| 7 | prj-c-secrets | — |
| 8 | fldr-network | — |
| 9 | prj-c-base-net-hub | — |
| 10 | prj-c-dns-hub | — |
| 11 | prj-c-interconnect | — |
| 12 | prj-c-restricted-net-hub | — |
| 13 | prj-d-shared-base | — |
| 14 | prj-d-shared-restricted | — |
| 15 | prj-n-shared-base | — |
| 16 | prj-n-shared-restricted | — |
| 17 | prj-p-shared-base | — |
| 18 | prj-p-shared-restricted | — |

Logs Among the four projects created under the common folder, two projects (`prj-c-logging`, `prj-c-billing-logs`) are used for logging. The first one is for organization-wide audit logs, and the second one is for billing logs. In both cases, the logs are collected into BigQuery datasets which you can then use for general querying, dashboarding, and reporting. Logs are also exported to Pub/Sub, a Cloud Storage bucket, and a log bucket.

Notes:

- Log export to Cloud Storage bucket has optional object versioning support via `log_export_storage_versions`.
- The various audit log types being captured in BigQuery are retained for 30 days.
- For billing data, a BigQuery dataset is created with permissions attached, however you will need to configure a billing export manually, as there is no easy way to automate this at the moment.

Security Command Center notification Another project created under the common folder. This project will host the Security Command Center notification resources at the organization level. This project will contain a Pub/Sub topic, a Pub/Sub subscription, and a Security Command Center notification configured to send all new findings to the created topic. You can adjust the filter when deploying this step.

KMS Another project created under the common folder. This project is allocated for Cloud Key Management for KMS resources shared by the organization.

Usage instructions are available for the org step in the README.

Secrets Another project created under the common folder. This project is allocated for Secret Manager for secrets shared by the organization.

Usage instructions are available for the org step in the README.

DNS hub This project is created under the network folder. This project will host the DNS hub for the organization.

Interconnect Another project created under the network folder. This project will host the Dedicated Interconnect connection for the organization. In case of Partner Interconnect, this project is unused and the VLAN attachments will be placed directly into the corresponding hub projects.

Networking Under the network folder, two projects, one for base and another for restricted network, are created per environment ([development](#), [nonproduction](#), and [production](#)) which is intended to be used as a Shared VPC host project for all projects in that environment. This stage only creates the projects and enables the correct APIs, the following networks stages, 3-networks-dual-svpc and 3-networks-hub-and-spoke, create the actual Shared VPC networks.

2. environments

The purpose of this stage is to set up the environments folders used for projects that contain monitoring and secrets projects. This will create the following folder and project structure:

| | | |
|----|----------------------|----|
| 1 | example-organization | └─ |
| 2 | fldr-development | └─ |
| 3 | prj-d-monitoring | └─ |
| 4 | prj-p-kms | └─ |
| 5 | prj-d-secrets | └─ |
| 6 | fldr-nonproduction | └─ |
| 7 | prj-n-monitoring | └─ |
| 8 | prj-n-kms | └─ |
| 9 | prj-n-secrets | └─ |
| 10 | fldr-production | └─ |
| 11 | prj-p-monitoring | └─ |
| 12 | prj-p-kms | └─ |
| 13 | prj-p-secrets | |

Monitoring Under the environment folder, a project is created per environment ([development](#), [nonproduction](#), and [production](#)), which is intended to be used as a Cloud Monitoring workspace for all projects in that environment. Please note that creating the workspace and linking projects can currently only be completed through the Cloud Console. If you have strong IAM requirements for these monitoring workspaces, it is worth considering creating these at a more granular level, such as per business unit or per application.

KMS Under the environment folder, a project is created per environment ([development](#), [nonproduction](#), and [production](#)), which is intended to be used by Cloud Key Management for KMS resources shared by the environment.

Usage instructions are available for the environments step in the README.

Secrets Under the environment folder, a project is created per environment ([development](#), [nonproduction](#), and [production](#)), which is intended to be used by Secret Manager for secrets shared by the environment.

Usage instructions are available for the environments step in the README.

3. networks-dual-svpc

This step focuses on creating a Shared VPC per environment ([development](#), [nonproduction](#), and [production](#)) in a standard configuration with a reasonable security baseline. Currently, this includes:

- (Optional) Example subnets for [development](#), [nonproduction](#), and [production](#) inclusive of secondary ranges for those that want to use Google Kubernetes Engine.
- Hierarchical firewall policy created to allow remote access to VMs through IAP, without needing public IPs.
- Hierarchical firewall policy created to allow for load balancing health checks.
- Hierarchical firewall policy created to allow Windows KMS activation.
- Private service networking configured to enable workload dependant resources like Cloud SQL.
- Base Shared VPC with private.googleapis.com configured for base access to googleapis.com and gcr.io. Route added for VIP so no internet access is required to access APIs.
- Restricted Shared VPC with restricted.googleapis.com configured for restricted access to googleapis.com and gcr.io. Route added for VIP so no internet access is required to access APIs.
- Default routes to internet removed, with tag based route [egress-internet](#) required on VMs in order to reach the internet.
- (Optional) Cloud NAT configured for all subnets with logging and static outbound IPs.

-
- Default Cloud DNS policy applied, with DNS logging and inbound query forwarding turned on.

Usage instructions are available for the networks step in the README.

3. networks-hub-and-spoke

This step configures the same network resources that the step 3-networks-dual-svpc does, but this time it makes use of the architecture based on the hub-and-spoke reference network model.

Usage instructions are available for the networks step in the README.

4. projects

This step is focused on creating service projects with a standard configuration that are attached to the Shared VPC created in the previous step and application infrastructure pipelines. Running this code as-is should generate a structure as shown below:

| | | |
|----|---------------------------|----|
| 1 | example-organization/ | └─ |
| 2 | fldr-development | └─ |
| 3 | fldr-development-bu1 | └─ |
| 4 | prj-d-bu1-kms | └─ |
| 5 | prj-d-bu1-sample-floating | └─ |
| 6 | prj-d-bu1-sample-base | └─ |
| 7 | prj-d-bu1-sample-restrict | └─ |
| 8 | prj-d-bu1-sample-peering | └─ |
| 9 | fldr-development-bu2 | └─ |
| 10 | prj-d-bu2-kms | └─ |
| 11 | prj-d-bu2-sample-floating | └─ |
| 12 | prj-d-bu2-sample-base | └─ |
| 13 | prj-d-bu2-sample-restrict | └─ |
| 14 | prj-d-bu2-sample-peering | └─ |
| 15 | fldr-nonproduction | └─ |
| 16 | fldr-nonproduction-bu1 | └─ |
| 17 | prj-n-bu1-kms | └─ |
| 18 | prj-n-bu1-sample-floating | └─ |
| 19 | prj-n-bu1-sample-base | └─ |
| 20 | prj-n-bu1-sample-restrict | └─ |
| 21 | prj-n-bu1-sample-peering | └─ |
| 22 | fldr-nonproduction-bu2 | └─ |
| 23 | prj-n-bu2-kms | └─ |
| 24 | prj-n-bu2-sample-floating | └─ |
| 25 | prj-n-bu2-sample-base | └─ |
| 26 | prj-n-bu2-sample-restrict | └─ |

| | | |
|----|---------------------------|----|
| 27 | prj-n-bu2-sample-peering | └─ |
| 28 | fldr-production | └─ |
| 29 | fldr-production-bu1 | └─ |
| 30 | prj-p-bu1-kms | └─ |
| 31 | prj-p-bu1-sample-floating | └─ |
| 32 | prj-p-bu1-sample-base | └─ |
| 33 | prj-p-bu1-sample-restrict | └─ |
| 34 | prj-p-bu1-sample-peering | └─ |
| 35 | fldr-production-bu2 | └─ |
| 36 | prj-p-bu2-kms | └─ |
| 37 | prj-p-bu2-sample-floating | └─ |
| 38 | prj-p-bu2-sample-base | └─ |
| 39 | prj-p-bu2-sample-restrict | └─ |
| 40 | prj-p-bu2-sample-peering | └─ |
| 41 | fldr-common | └─ |
| 42 | prj-c-bu1-infra-pipeline | └─ |
| 43 | prj-c-bu2-infra-pipeline | |

The code in this step includes two options for creating projects. The first is the standard projects module which creates a project per environment, and the second creates a standalone project for one environment. If relevant for your use case, there are also two optional submodules which can be used to create a subnet per project, and a dedicated private DNS zone per project.

Usage instructions are available for the projects step in the README.

5. app-infra

The purpose of this step is to deploy a simple Compute Engine instance in one of the business unit projects using the infra pipeline set up in 4-projects.

Usage instructions are available for the app-infra step in the README.

Final view

After all steps above have been executed, your Google Cloud organization should represent the structure shown below, with projects being the lowest nodes in the tree.

| | | |
|---|----------------------|----|
| 1 | example-organization | └─ |
| 2 | fldr-common | └─ |
| 3 | prj-c-logging | └─ |
| 4 | prj-c-billing-logs | └─ |
| 5 | prj-c-scc | └─ |
| 6 | prj-c-kms | └─ |

| | | |
|----|---------------------------|---|
| 7 | prj-c-secrets | — |
| 8 | prj-c-bu1-infra-pipeline | — |
| 9 | prj-c-bu2-infra-pipeline | — |
| 10 | fldr-network | — |
| 11 | prj-c-base-net-hub | — |
| 12 | prj-c-dns-hub | — |
| 13 | prj-c-interconnect | — |
| 14 | prj-c-restricted-net-hub | — |
| 15 | prj-d-shared-base | — |
| 16 | prj-d-shared-restricted | — |
| 17 | prj-n-shared-base | — |
| 18 | prj-n-shared-restricted | — |
| 19 | prj-p-shared-base | — |
| 20 | prj-p-shared-restricted | — |
| 21 | fldr-development | — |
| 22 | prj-d-monitoring | — |
| 23 | prj-d-kms | — |
| 24 | prj-d-secrets | — |
| 25 | fldr-development-bu1 | — |
| 26 | prj-d-bu1-kms | — |
| 27 | prj-d-bu1-sample-floating | — |
| 28 | prj-d-bu1-sample-base | — |
| 29 | prj-d-bu1-sample-restrict | — |
| 30 | prj-d-bu1-sample-peering | — |
| 31 | fldr-development-bu2 | — |
| 32 | prj-d-bu2-kms | — |
| 33 | prj-d-bu2-sample-floating | — |
| 34 | prj-d-bu2-sample-base | — |
| 35 | prj-d-bu2-sample-restrict | — |
| 36 | prj-d-bu2-sample-peering | — |
| 37 | fldr-development-bu2 | — |
| 38 | prj-d-bu2-kms | — |
| 39 | prj-d-bu2-sample-floating | — |
| 40 | prj-d-bu2-sample-base | — |
| 41 | prj-d-bu2-sample-restrict | — |
| 42 | prj-d-bu2-sample-peering | — |
| 43 | fldr-nonproduction | — |
| 44 | prj-n-monitoring | — |
| 45 | prj-n-kms | — |
| 46 | prj-n-secrets | — |
| 47 | fldr-nonproduction-bu1 | — |
| 48 | prj-n-bu1-kms | — |
| 49 | prj-n-bu1-sample-floating | — |
| 50 | prj-n-bu1-sample-base | — |

| | | |
|----|---------------------------|---|
| 51 | prj-n-bu1-sample-restrict | — |
| 52 | prj-n-bu1-sample-peering | — |
| 53 | fldr-nonproduction-bu2 | — |
| 54 | prj-n-bu2-kms | — |
| 55 | prj-n-bu2-sample-floating | — |
| 56 | prj-n-bu2-sample-base | — |
| 57 | prj-n-bu2-sample-restrict | — |
| 58 | prj-n-bu2-sample-peering | — |
| 59 | fldr-production | — |
| 60 | prj-p-monitoring | — |
| 61 | prj-p-kms | — |
| 62 | prj-p-secrets | — |
| 63 | fldr-production-bu1 | — |
| 64 | prj-p-bu1-kms | — |
| 65 | prj-p-bu1-sample-floating | — |
| 66 | prj-p-bu1-sample-base | — |
| 67 | prj-p-bu1-sample-restrict | — |
| 68 | prj-p-bu1-sample-peering | — |
| 69 | fldr-production-bu2 | — |
| 70 | prj-p-bu2-kms | — |
| 71 | prj-p-bu2-sample-floating | — |
| 72 | prj-p-bu2-sample-base | — |
| 73 | prj-p-bu2-sample-restrict | — |
| 74 | prj-p-bu2-sample-peering | — |
| 75 | fldr-bootstrap | — |
| 76 | prj-b-cicd | — |
| 77 | prj-b-seed | |

Branching strategy

There are three main named branches: `development`, `nonproduction`, and `production` that reflect the corresponding environments. These branches should be protected. When the CI/CD Pipeline (Jenkins or Cloud Build) runs on a particular named branch (say for instance `development`), only the corresponding environment (`development`) is applied. An exception is the `shared` environment, which is only applied when triggered on the `production` branch. This is because any changes in the `shared` environment may affect resources in other environments and can have adverse effects if not validated correctly.

Development happens on feature and bug fix branches (which can be named `feature/new-foo`, `bugfix/fix-bar`, etc.) and when complete, a pull request (PR) or merge request (MR) can be opened targeting the `development` branch. This will trigger the CI/CD Pipeline to perform a plan and validate against all environments (`development`, `nonproduction`, `shared`, and

`production`). After the code review is complete and changes are validated, this branch can be merged into `development`. This will trigger a CI/CD Pipeline that applies the latest changes in the `development` branch on the `development` environment.

After validated in `development`, changes can be promoted to `nonproduction` by opening a PR or MR targeting the `nonproduction` branch and merging them. Similarly, changes can be promoted from `nonproduction` to `production`.

Policy validation

This repo uses the terraform-tools component of the `gcloud` CLI to validate the Terraform plans against a library of Google Cloud policies.

The Scorecard bundle was used to create the policy-library folder with one extra constraint added.

See the policy-library documentation if you need to add more constraints from the samples folder in your configuration based in your type of workload.

Step 1-org has instructions on the creation of the shared repository to host these policies.

Optional Variables

Some variables used to deploy the steps have default values, check those **before deployment** to ensure they match your requirements. For more information, there are tables of inputs and outputs for the Terraform modules, each with a detailed description of their variables. Look for variables marked as **not required** in the section **Inputs** of these READMEs:

- Step 0-bootstrap: If you are using Cloud Build in the CI/CD Pipeline, check the main README of the step. If you are using Jenkins, check the README of the module `jenkins-agent`.
- Step 1-org: The README of the environment `shared`.
- Step 2-environments: The READMEs of the environments `development`, `nonproduction`, and `production`
- Step 3-networks-dual-svpc: The READMEs of the environments `shared`, `development`, `nonproduction`, and `production`
- Step 3-networks-hub-and-spoke: The READMEs of the environments `shared`, `development`, `nonproduction`, and `production`
- Step 4-projects: The READMEs of the environments `shared`, `development`, `nonproduction`, and `production`

Errata summary

Refer to the errata summary for an overview of the delta between the example foundation repository and the Google Cloud security foundations guide.

Contributing

Refer to the contribution guidelines for information on contributing to this module.