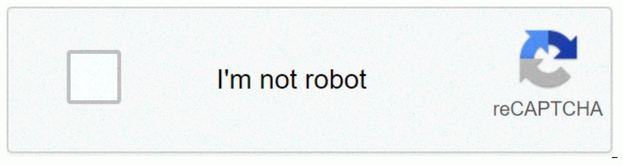

Continue

Google deployment manager templates

Google Cloud Deployment Manager is an infrastructure management service that makes it easy to create, deploy, and manage resources on the Google Cloud Platform. With Deployment Manager, you can create a static or dynamic template that describes the configuration of your Google Cloud environment, and then use Deployment Manager to create these resources as a single deployment. This repository contains example templates that you can use with Deployment Manager. For an overview of Deployment Manager, the topic Run the scenario tutorial To try the samples without cloning the data store to cloudshell, editing the samples, and installing them within Cloud Shell, without installing them. If you want to try a simple deployment in Cloud Shell, open the quickstart. Cloud Foundation Toolkit We're happy to introduce the Cloud Foundation toolkit, which provides a common baseline for gcp best practices implemented as code with infrastructure. Most of the templates are available under social/cloud foundation, revised but not yet committed templates are included in the cloud base branch These templates define a high level of infrastructure automation by adding a detailed schema, documentation, and integration tests. The example configuration allows you to install these templates within minutes. Contribution contributions to this directory are always welcome and strongly encouraged. For more information about getting started, see CONTRIBUTE. License Apache 2.0 - For more information, see LICENSE. Page 2 You cannot perform this action at this time. You are logged on with a different tab or window. To update the session, reload it. You have logged out of another tab or window. To update the session, reload it. We use optional third-party analytics cookies to understand how you use GitHub.com to make better products. find out more. We use optional third-party analytics cookies to understand how you use GitHub.com to make better products. You can update your selection at any time by clicking the Cookie Settings button at the bottom of the page. For more information, see our privacy statement. We use basic cookies to perform the basic functions of the website, e.g. they are used to log in. For more information, see Always active: We use analytics cookies to understand how you use our websites to improve them, such as using them to collect information about the pages you visit and how many clicks you need to complete a task. Learn more In the last two lessons, we covered the creation of a deployment manager template file and covered the role of the configuration file. In this I'll show you how to combine the two to deploy a VM instance. So, let's get started. Before I do anything, I need to decide which project I'm going to use, as Deployment Manager requires me to specify a GCP console project. For this exercise, I'm using the existing Cloud Academy project. I'm going to run this installation on Google Cloud Cloud which comes with gcloud installed. Gcloud is what we're going to use to install our template. To run cloud shell deployment, I need to start the shell and then start the code editing window. So let's go ahead and activate Cloud Shell. Then we start the code editing window. At this point, I need to drag the templatedemo folder from the workstation to the explorer here code editor. This allows me to run my commands and access the files for that reference. So let me go down here and open the table here. You can see here that my templatedemo folder and everything in it, now put cloud shell So, now that I have everything in my environment set up, I can start the installation. What I'm going to do here is change my templatedemo directory and then run the installation from there, as this is where the files are. To run here, using the demovm template I showed earlier, I need to run the command I show here on the screen. And I'm going to copy this here because it's a long order. So what we're going to do here is we're going to use gcloud deployment-manager to create a centralized myvm-with-templates, and we're going to configure that deployment to the myvm.yaml file. When this deployment is running, the configuration file imports the template and remember that the configuration file is the YAML file. And it will import the ginger file, which is the template through the import section, which we added to the configuration file, and we covered this earlier. Then let's go ahead and press Enter here. And we see the mission begin. Now, if you switch to the console and go into the compute engine, you can see the first vm deployed. Switch back to Cloud Shell and continue with the installation. It takes a few minutes and here you can see that it was completed without error. Now I'm going to run the gcloud deployment-manager command, which we're going to set up, and that command allows me to watch my deployment. Let me take that order. We call the Gcloud deployment manager and use the deployment command and use a description instead of creating it, and essentially tell Deployment Manager to show you a deployment called myvm-emplate. Then let's go ahead and press Enter here. And as you can see, this command will return my installation information. You can see the name of the end time, the type of operation, the deployment startup, and its status. We can also see who was driving it. Here below you can see that the status is complete. So now that you know how to create a simple VM by deploying a template! Create and manage cloud resources with simple templates. Try the free Contact sales View documentation for your product. Google Cloud Deployment Manager allows you to use all the resources you need for your app using yaml. You can also parameterize the configuration using Python or Jinja2 templates and allow you to reuse common deployment paradigms, such as the load-balancing, autoscaled instance group. Treat the configuration as code and perform repeatable deployments. By creating configuration files that define resources, the process of creating resources can be repeated over and over again with consistent results. Many tools take an essential approach that requires the user to determine the steps to create and configure resources. The declarative approach allows the user to specify what the configuration should be and let the system figure out the steps they need to take. The user can focus on a set of resources that are made up of the application or service rather than deploying each resource separately. Templates allow you to use building blocks to create abstractions or resources that are typically deployed together, such as an instance template, instance group, and autosize. These templates can be parameterized to be used over and over again by changing input values to determine which image to install, in which zone to install, or how many virtual machines to install. You can deploy multiple resources at the same time, in parallel. Python and Jinja2 templates are programmed to control what will be installed. Add, delete, or modify resources in the deployment. Pass variables (e.g. zone, machine size, number of machines, status: test, prod, staging) into the templates and return output values (e.g. IP address assigned, link to the instance). JSON schema for defining and limiting parameters. One resource definition can refer to another resource that creates dependencies and controls the order in which the resource is created. Before you commit your changes, see what changes Deployment Manager makes to a create or update operation. View deployments in the Google Cloud Console, where you can see a view of the entire deployment in a hierarchical view. Wix Media Group creates and manages deployments installed on the Google Cloud Platform in Deployment Manager. This allows us to easily install systems on different sites, provides us with the control we need to roll out new services easily, and gives us the flexibility we need to install multiple versions of the code at once. — Golan Parashi, Infrastructure Technical Manager, Wix Media Group Deployment Manager is available free of charge to Google Cloud Platform customers. [{ type: thumb-down, id: hardToUnderstand, label:Hard to understand },{ type: thumb-down, id: incorrectInformationOrSampleCode, label:Incorrect or sample code },{ type: thumb-down, id: missingTheInformationSamplesINeed, label:Missing the information/samples Need },{ type: thumb-down, id: otherDown, label:Other }] [{ type: thumb-up, id: easyToUnderstand, label:Easy to understand },{ type: thumb-up, id: solvedMyProblem, label:Solved my problem },{ type: type: id: otherUp, label:Other }] The following components are the basics of Deployment Manager. Configuration A describes all the resources required for a single deployment. The configuration is a file written using yaml syntax, which lists the resources you want to create and their corresponding resource properties. The configuration must include a resource: section, followed by a list of resources to create. Each resource must contain three components: name - A user-defined string that identifies this resource, such as my-vm, project-data-disk, a-test-network. Type - The type of resource installed, such as compute.v1.instance, compute.v1.disk. For a description of the base resource types and supported resource types, see the documentation. properties - Parameters for the resource type. It must match the properties of the type as the zone: asia-east1-a, boot: true. This is an example configuration: resources: - name: a-first-vm type: compute.v1.instance properties: zone: us-central1-a machineType: disks: - deviceName: boot type: CONSTANT boot: TRUE autoDelete: true initializeParams: sourceImage: networkInterfaces: - network: accessConfigs: - name: External NAT type: ONE_TO_ONE_NAT Templates The configuration can be templates that are essentially part of the configuration file, towed into each building block. After you create a template, you can reuse them between deployments as needed. Similarly, if you find yourself rewriting configurations with very similar properties, you can make shared parts abstract into templates. Templates are more flexible than individual configuration files and are designed to support easy portability between deployments. The template file is written in Python or Jinja2. Deployment Manager interprets all templates recursively and to the results within the configuration file. Thus, interpreting each template eventually results in the same YAML syntax for resources as for the configuration file. To create a simple template, see Create a simple template. Configurations are described as fully expanded or unopened. The fully expanded configuration describes all the resources and properties of the deployment, including content from imported template files. For example, it provides an unopened configuration that uses a template such as import: - path: vm_template.jinja resources: - name: vm instance type: vm_template.jinja properties: zone: us-central1-a project: myproject Once expanded, your configuration file would contain the of all your templates, like so: resources: - name: the-first-vm type: type: properties: zone: us-central1-a machineType: disks: - deviceName: boot type: PERSISTENT boot: true autoDelete: true initializeParams: sourceImage: networkInterfaces: - network: accessConfigs: - name: External NAT type: ONE_TO_ONE_NAT Resource A resource represents a single API resource. This can be an API resource provided by a Google-managed base type, or an API resource provided by a type provider. For example, a Compute Engine instance is a single resource, a Cloud SQL instance is a single resource, and so on. To specify a resource, enter a type for that resource. For more information about these types, see the Types section below. Types: To create a resource in Deployment Manager, you must specify a type. A type can represent an API resource, called a base type, or a pool of resources known as a composite type that is created as part of the deployment. For example, to create an instance of a compute engine VM, specify the appropriate base type as in configuration: resources: - name: a-first-vm type: compute.v1.instance # Resource type to deploy ... Deployment Manager offers a list of basic types maintained by Google that you can use right away. For a list of these types, see the Supported Resource Types and Properties documentation. Base types and type providers: The base type creates a single primitive resource. Google-owned core types include compute.v1.instance, storage.v1.bucket, and sqladmin.v1beta4.database, all of which are served by the corresponding Compute Engine V1 API, Cloud Storage V1 API, and Cloud SQL v1beta4 Admin API. The basic types are supported by a RESTful API that supports create, read, update, and delete (CRUD) operations. You can also create additional basic types by adding a type provider if google-owned types don't meet your needs on their own. Creating a type provider makes all resources in the API available as base types. To create a type provider, you must specify an API descriptive document that can be an OpenAPI specification or Google Discovery, modify the input mappings required for the API, and register the type with Deployment Manager. Once created, you and other users with access to the project can use the types provided by your service provider. When you add a type provider, all resources provided by the API and supported by the RESTful interface that contains THE CREATE, Read, Update, and Delete (CRUD) operations are displayed as types that can be used in the deployment. Creating your own type provider is a special scenario, and Google recommends that you do so only if you are very familiar with the API you want to integrate. For information about creating a type provider, see Integration with deployment. you can read. When you call a base type in templates or configurations, you use one of the following syntaxes, depending on the type. For google-managed base types, type [API]. [VERSION]. [RESOURCE] For example, compute.v1.instance. For Google-managed type providers (beta), use gcp-types/[PROVIDER]:[RESOURCE] For a list of supported type providers, see Supported GCP type providers. For base types provided by a type provider, use [PROJECT_ID]/[TYPE_PROVIDER]:[COLLECTION] Where [COLLECTION] is the path to the API resource that can be deployed. Composite types The composite type contains one or more templates that are preconfigured for collaboration. These templates can be expanded to a set of base types when deployed. Complex types are essentially hosted templates that you can add to Setup Manager. You can create complex types for common solutions so that the solution can be easily reused, or you can create complex settings that you can reuse in the future. For example, you can create a composite type that installs a managed NLB instance group. Network Load Balancer requires multiple Google Cloud Platform resources and cross-resource configuration, so you can set these resources once in a configuration and register the type in Deployment Manager. You and other users who have access to the project can then call this type and install it in future configurations. To call a composite type, use [PROJECT_ID]/composite:[TYPE_NAME] For example, resources: - name: my-composite-type type: myproject/composite:example-composite-type To learn how to create a composite type, see Add a composite type to deployment manager. The manifest file is a read-only object that contains the original configuration you specified, including imported templates, and includes a fully expanded resource list created by Deployment Manager. For each deployment update, Deployment Manager creates a new manifest that reflects the new state of the deployment. When troubleshooting deployment problems, it is useful to view the manifest file. For more information, see View manifest. Deployment: A deployment is a collection of resources that are installed and managed together, using a configuration. For more information, see Create a deployment. What's next?