Setting Template Properties and Using Environment Variables

One advantage of using templates for your deployment is the ability to create and define custom properties, which let you reuse templates across zones, regions, and projects.

Template properties are arbitrary variables. Any configuration file or template file can provide a value for a template property without modifying the template. Therefore, you can change a property's value for various configurations without changing the template itself.

To reference an arbitrary value, use this syntax in a template:

```
xt.properties["property-name"]
```

In addition to template properties, you can also use environment variables specific to your deployment, which are pre-populated with information about the deployment. You can use environment variables in templates to get unique information about your deployment.

You call an environment variable by using this syntax:

```
xt.env['variable-name']
```

Valid environment variables include the deployment name, the project ID, the name property of your resource, and the type of your configuration. Learn more about environment variables (/deployment-manager/docs/configuration/templates/use-environment-variables).

Template properties and environment variables in a template

In this step, **`vm-template.py`** shows the benefits of template properties and environment variables. Open **`vm-template.py`**:

```
ploymentmanager-samples/examples/v2/step_by_step_guide/step7_use_environment_variabl
```

```
vm-template.py  # use your preferred text editor
```

Various parts of the template have been replaced with template properties and environment variables. For example, the project ID is replaced with `context.env[project]`, so you don't have to manually replace the project ID in your templates.

The comments in the file describe other changes to the template.

examples/v2/step_by_step_guide/step7_use_environment_variables/python/vm-template.py (https://github.com/GoogleCloudPlatform/deploymentmanager-samples/blob/master/examples/v2/step_by_step_guide/step7_use_environment_variables/python/vm-template.py)

/blob/master/examples/v2/step_by_step_guide/step7_use_environment_variables/python/vm-template.py)

```
# Copyright 2016 Google Inc. All rights reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

"""Creates the virtual machine."""

COMPUTE_URL_BASE = 'https://www.googleapis.com/compute/v1/'


def GenerateConfig(context):
  """Creates the virtual machine with environment variables."""

  resources = [{
      # `the-first-vm` is replaced with `context.env[`name`]`
      'name': context.env['name'],
      'type': 'compute.v1.instance',
      'properties': {
          # All occurrences of `us-central1-f` are replaced with
          # `context.properties[`zone`].
          # All occurrences of the project ID are replaced with
          # `context.env[`project`]`.
          # `f1-micro` is replaced with `context.properties["machineType"].
          'zone': context.properties['zone'],
          'machineType': ''.join([COMPUTE_URL_BASE, 'projects/',
                                  context.env['project'], '/zones/',
                                  context.properties['zone'], '/machineTypes/',
                                  context.properties['machineType']]),
          'disks': [{
              'deviceName': 'boot',
              'type': 'PERSISTENT',
              'boot': True,
              'autoDelete': True,
              'initializeParams': {
                  'sourceImage': ''.join([COMPUTE_URL_BASE, 'projects/',
                                          'debian-cloud/global/',
                                          'images/family/debian-9'])
              }
          }],
          # `$(ref.a-new-network.selfLink)` is replaced with
          # `$(ref.` + context.properties[`network`] + `selfLink)`.
          'networkInterfaces': [{
              'network': '$(ref.' + context.properties['network']
                         + '.selfLink)',
              'accessConfigs': [{
                  'name': 'External NAT',
                  'type': 'ONE_TO_ONE_NAT'
              }]
          }]
      }
  }]
  return {'resources': resources}
```

Similarly, `network-template.py` and `firewall-template.py` use the deployment's name in their definition, by calling `context.env['name']`.

Deploy the configuration

To view the configuration file for this deployment, run the following command:

```
config-with-many-templates.yaml
```

Save your changes and redeploy your configuration to confirm the variables work.

```
d deployment-manager deployments create deployment-with-template-properties --config
```

Delete Your Deployment

We recommend that you delete the deployment to avoid charges. You don't need this deployment for the next step. Run the following command to delete the deployment:

```
d deployment-manager deployments delete deployment-with-template-properties
```

Looking ahead: helper scripts

Next, learn about helper scripts to efficiently perform repeated tasks.

Previous
← Using multiple templates (/deployment-manager/docs/step-by-step-guide/using-multiple-templates)

Next
Exploring helper scripts (/deployment-manager/docs/step-by-step-guide/learn-helper-scripts) →

Last updated 2020-06-22 UTC.