Integrations Storage UpCloud services Updated on 20.10.2022

How to create custom images using UpCloud Packer builder

Packer is an easy-to-use automation solution for creating any type of machine image. It embraces modern configuration management by allowing automated software installation and setup within Packer-built images. UpCloud Packer builder integrates Packer with our Cloud Servers and makes creating private templates even faster!

UpCloud Packer builder is a plugin for Packer to simplify template configuration and make deploying custom Cloud Servers quick and easy. In this guide, we'll show the steps required to install Packer on Linux and how to create your first custom template on UpCloud. Packer is also available for macOS and Windows with their own installation instructions on their download page.

Test hosting on UpCloud!

Installing Packer

Packer can be downloaded for most operating systems as well as installed using common package managers. To install the precompiled binary, you will need to download the appropriate package for your OS. On most popular Linux distributions, you can run the following commands to install Packer using your native package manager.

```
# Debian and Ubuntu
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) mai
n"
```

```
sudo apt-get update && sudo apt-get install packer
```

```
# CentOS
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo
sudo yum -y install packer
```

Afterwards, verify that Packer is working, for example, with the command below.

```
packer --version
```

```
1.7.3
```

The Packer builder leverages the UpCloud Go API to interface with the UpCloud API. You will need to provide a username and password with access rights to the API functions to authenticate. We recommend setting up a new workspace member account with only the API privileges for security purposes. You can do this at your UpCloud Control Panel. Learn more about creating API credentials in our guide for getting started with UpCloud API.

Enter the API user credentials in your terminal with the following two commands. Replace the `username` and `password` with your user details.

```
export PKR_VAR_UPCLOUD_USERNAME=username
export PKR_VAR_UPCLOUD_PASSWORD=password
```

You should also save these in your profile file to avoid having to repeat the export command every time you open a new terminal. Simply add the same lines to the `~/.profile` or `~/.bashrc` file.

Installing Packer is all that you really need to build custom templates on UpCloud. Continue ahead in the next section to get started with configuring your templates.

Configuring templates

Packer uses the Hashicorp configuration language (HCL) format for the configuration files to define the template you wish to build. You can find an example configuration on our GitHub repository at `/packer-plugin-upcloud/examples`.

To start, you might want to make a new directory for arranging your Packer configurations, for example, `~/packer`.

```
mkdir ~/packer
```

Next create a new template with your favourite editor, for example using the following command.

```
nano ~/packer/upcloud-template.pkr.hcl
```

Then copy and paste the example template underneath.

```
variable "UPCLOUD_USERNAME" {
  type = string
  default = ""
}
variable "UPCLOUD_PASSWORD" {
  type = string
  sensitive = true
  default = ""
}

packer {
    required_plugins {
        upcloud = {
            version = "≥v1.0.0"
            source = "github.com/UpCloudLtd/upcloud"
        }
    }
}

source "upcloud" "example" {
  username = "${var.UPCLOUD_USERNAME}"
  password = "${var.UPCLOUD_PASSWORD}"
  zone = "nl-ams1"
  storage_name = "ubuntu server 20.04"
  template_prefix = "ubuntu-server"
}

build {
  sources = ["source.upcloud.example"]

  provisioner "shell" {
    inline = [
      "apt-get update",
      "apt-get upgrade -y",
      "echo 'ssh-rsa-key' | tee /root/.ssh/authorized_keys"
    ]
  }
}
```

The basic template is almost ready to deploy. However, you should take a look at the parameters in the source and `build` segments.

The `username` and `password` are rather self-explanatory and should be the same for every template. Since we already set our actual UpCloud API credentials in the environmental variables, there’s no need to include them in the configuration file. Packer will then find the credentials from your environmental variables at run time and set them in the username and password variables as defined in your configuration.

The important parts are the target `zone` and the source `storage-name`. These tell Packer which public template you wish to use as the basis for generating your own and where it should be made available.

Choose the zone where you wish to deploy cloud servers with the custom template. Note that the custom template is only available in the zone it was created in. If you want to use the custom template to deploy Cloud Servers in multiple zones, you will need to recreate the custom template in each zone.

The currently available zones are the following:

Europe

- Amsterdam `nl-ams1`
- Frankfurt `de-fra1`
- Helsinki `fi-hel1`
- Helsinki `fi-hel2`
- London `uk-lon1`
- Madrid `es-mad1`
- Warsaw `pl-waw1`

Americas

- Chicago `us-chi1`
- New York `us-nyc1`
- San Jose `us-sjo1`

Asia-Pacific

- Singapore `sg-sin1`
- Sydney `au-syd1`

The second bit you should select is the public template that will be used to generate your custom template. The example configuration below uses the Ubuntu 20.04 image, but you can use any Linux template you wish available as a public template on UpCloud.

With the basic configuration done, the customization to the template can then be added to the `provisioners` segment. The example provisioner runs the basic update and upgrades commands in the shell.

Additionally, if you want to log into a server deployed with the template, you need to include an SSH key to your root user by replacing the ssh-rsa-key with your public key or provisioning another username. The Packer generates a temporary SSH key while building the template which cannot be used afterwards.

You can find instructions on how to use different types of provisioners to customize your template in the Packer documentation for provisioners.

Once you have made the configurations, save the file and exit the editor.

Building templates using Packer

With the template configuration ready, continue by initialising Packer using the following command.

```
packer init ~/packer/basic_example.pkr.hcl
```

This will tell Packer to prepare to build the template and check for the required plugin. Packer will download the UpCloud Packer builder and save it in `~/.packer.d/plugins` directory. You should see a confirmation such as in the example output below.

```
Installed plugin github.com/upcloudltd/upcloud v1.2.0 in
"/home/user/.packer.d/plugins/github.com/upcloudltd/upcloud/packer-plugin-upcloud_v1.2.0_x5.0_linux
_amd64"
```

With Packer initialised and the UpCloud plugin installed, validate your Packer configuration using the next command.

```
packer validate ~/packer/basic_example.pkr.hcl
```

Packer will then check through your template configuration file syntax. If you get no warnings, everything is in order.

You are then ready to build your template. Use the command below to generate your custom template based on the configuration file.

```
packer build ~/packer/basic_example.pkr.hcl
```

```
==> upcloud: Creating temporary SSH key ...
==> upcloud: Getting storage ...
==> upcloud.com: Creating server based on storage "Ubuntu Server 20.04 LTS (Focal Fossa)" ...
...
==> Wait completed after 3 minutes 49 seconds

==> Builds finished. The artifacts of successful builds are:
--> upcloud.com: Storage template created, UUID: 01f13bb4-ca38-4d7f-baa0-2bb14a18631d
```

When the deployment process finishes, you should see an output similar to the example above.

Your new custom template is then available to deploy via your UpCloud Control Panel or using any of our cloud management tools. For example, you can deploy the template to a new Cloud Server using Terraform and the following configuration.

```
resource "upcloud_server" "example" {
  hostname = "example"
  zone     = "nl-ams1"
  plan     = "2xCPU-4GB"

  template {
      storage = "01f13bb4-ca38-4d7f-baa0-2bb14a18631d"
      size    = 100
  }

  network_interface {
    type = "public"
  }

  connection {
    host        = self.network_interface[0].ip_address
    type        = "ssh"
    user        = "root"
    private_key = file("~/.ssh/id_rsa")
  }
}
```

Find out more about Terraform and how to get started at our other tutorials.

Conclusions

Congratulations, you should now have your own custom template visible in your UpCloud Control Panel under Storage and Custom images tab. With the simple configuration process and the fast deployment, you can have a purpose-build template ready in minutes.

Test it out by pressing the Deploy button and start a new server from the custom template to verify it was built to your specifications.

Janne Ruostemaa

Latest tutorials

How to get started with Managed Kubernetes

How to set up a private VPN Server using UpCloud and UTunnel

How to enable Anti-affinity using Server Groups with the UpCloud API

How to scale Cloud Servers without shutdown using Hot Resize

How to add SSL Certificates to Load Balancers

How to get started with Managed Load Balancer

How to export cloud resources and import to Terraform

See all tutorials →

})(window);