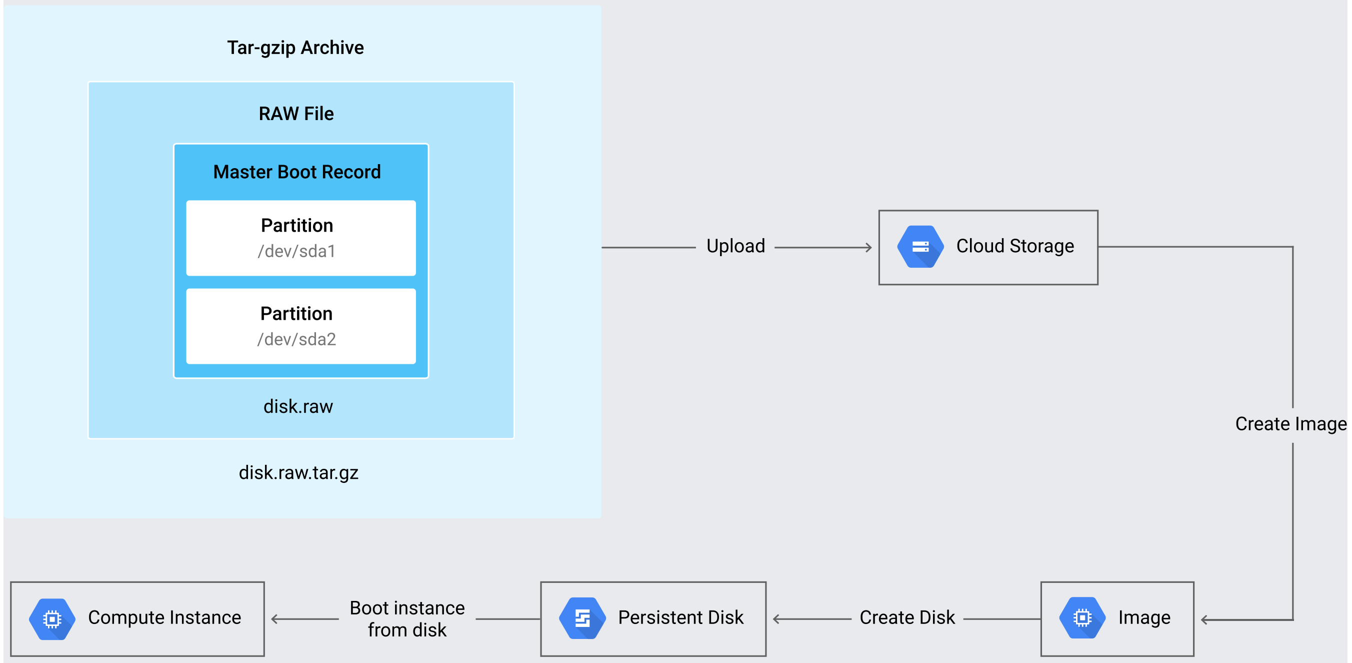
This solution provides in-depth guidance on how to manage Compute Engine images. Images provide the base operating environment for applications that run in Compute Engine, and they are critical to ensuring your application deploys and scales quickly and reliably. You can also use images to archive application versions for disaster recovery or rollback scenarios.

An image (/compute/docs/images) in Compute Engine is a cloud resource that provides a reference to an immutable disk. That disk representation is then encapsulated using a few data formats.



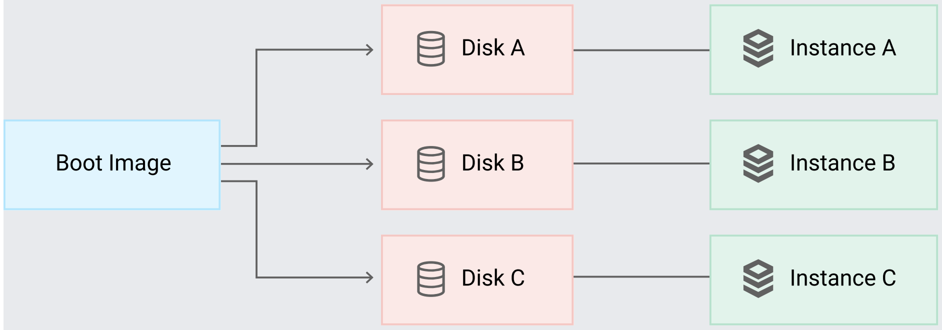
An image is a bundle of the raw bytes used to create a prepopulated hard disk. Written on any formatted disk is a partition table that points to one or more partitions that contain data. For an image to be bootable, it must contain a master boot record (https://wikipedia.org/wiki/Master_boot_record) and a bootable partition. For a disk to be imported as a Compute Engine image, the disk's bytes must be written to a file named `disk.raw`.

After the complete sequence of bytes from the disk are written to the file, the file is archived using the tar format (http://www.gnu.org/software/tar/manual/html_node/Standard.html) and then compressed using the GZIP format. You can then upload the resulting `*.tar.gz` file to Cloud Storage and register it as an image in Compute Engine, as shown in the preceding diagram. After you register an image, you can use it to create exact replicas of the original disk in any region of Google Cloud. The newly registered images are often used as boot volumes for Compute Engine instances.

For a more basic introduction to these Compute Engine terms, see Virtual machine instances (/compute/docs/instances/) and Images (/compute/docs/images) in the documentation.

The first step in using Compute Engine is to choose the image you want as the operating system for your virtual machine (VM) instance. You can use public images supplied by Google Cloud (/compute/docs/images#os-compute-support), which are updated on a regular basis. Google Cloud provides a variety of operating systems, including Debian, Ubuntu, and CentOS, for your use at no extra cost. Some operating systems, such as Red Hat Enterprise Linux and Microsoft Windows, are premium images, which incur additional fees for every hour that the instances run.

For more information on a particular image, such as automatic update policies, security patching, and support channels, see the Operating system details (/compute/docs/images#os-details) section of the product documentation.



You can use the Google Cloud public images to boot a Compute Engine instance, after which you can customize the instance to run your application.

One approach to configuring your instance is to use the startup script (/compute/docs/startupscript) to run the commands that deploy your application as it boots. Keep in mind that this script runs every time the instance boots, so you must make the script idempotent to avoid ending up in an inconsistent or partially configured state. If your instances are part of a managed instance group, you can use the Instance Group Updater to restart or rebuild your instances, which reruns your startup script. A common practice is to use the startup script to run a configuration management tool such as Chef or Ansible.

While configuring an instance's startup script is a viable way to provision your infrastructure, a more efficient method is to create a new custom image with your configuration incorporated into the public image. You can customize images in several ways:

- Manual
- Automated
- Import

The process of creating a custom image is called *baking*.

Baking your images has the following benefits:

- Shorter time from boot to application readiness.
- Enhanced reliability for application deployments.
- Easier rollback to previous versions.
- Fewer dependencies on external services during application bootstrap.
- Scaling up creates instances that contain identical software versions.

You can create a simple custom image by creating a new VM instance from a public image (/compute/docs/images#os-compute-support), configuring the instance with the applications and settings that you want, and then creating a custom image from that instance. Use this method if you can configure your images from scratch manually rather than using automated baking (#automated_baking) or importing existing images (#importing_existing_images).

You can create a simple custom image using the following steps:

1. Create an instance from a public image (/compute/docs/instances/create-start-instance#publicimage).
2. Connect to the instance (/compute/docs/instances/connecting-to-instance).
3. Customize the instance for your needs.
4. Stop the instance (/compute/docs/instances/stopping-or-deleting-an-instance#stop_an_instance).
5. Create a custom image (/compute/docs/images/create-delete-deprecate-private-images#creating_a_custom_image) from the

boot disk of that instance. This process requires you to delete the instance but keep the boot disk.

Manual baking is an easy way to start if you have a small number of images, but large numbers of images become difficult to audit and manage. Packer (https://packer.io) is an open source tool for making image creation more reproducible, auditable, configurable, and reliable. For more information on how to create an automated image-creation pipeline, see Automated image builds with Jenkins, Packer, and Kubernetes (/solutions/automated-build-images-with-jenkins-kubernetes). You can also use Packer as part of a Spinnaker pipeline to produce images that are deployed to clusters of instances.

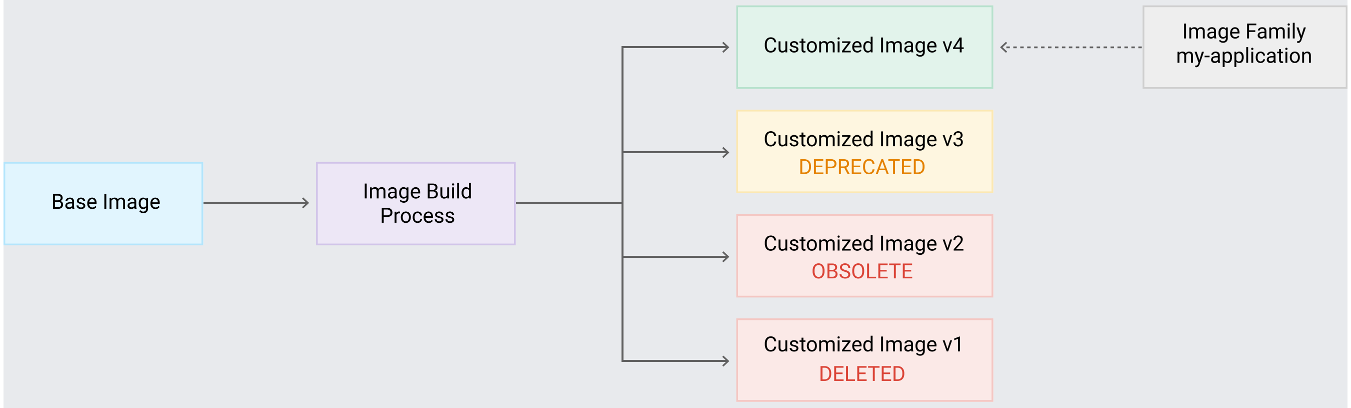
You can migrate images by exporting them from their existing infrastructure to Compute Engine. For Linux machines, here's an in-depth guide on migrating RAW disk images, Amazon Machine Images (AMI) and VirtualBox images (/compute/docs/images/import-existing-image).

Another option for importing your existing images is to use a paid migration service like CloudEndure (http://cloudendure.com). CloudEndure is a tool chain and online service that facilitates the migration of machines from one platform to another with minimal downtime using continuous block-level replication. With CloudEndure you can migrate your machines to Compute Engine and then use manual baking to create images.

All disks in Compute Engine are encrypted by default using Google’s encryption keys. Images built from disks are also encrypted. Alternatively, you can provide your own encryption keys when your disks are created (/compute/docs/disks/customer-supplied-encryption#encrypt_a_new_persistent_disk_with_your_own_keys). After you create the disk, you can create an encrypted image by providing your encryption keys to the image create command. For more information about encryption at rest and customer-supplied encryption keys, see Encryption at rest (/security/encryption-at-rest/) in the Google Cloud documentation.

After you set up an image-build pipeline, you can use images to reliably launch instances of an application. While the pipeline can handle creating images, you must also ensure that your deployment mechanisms use the latest versions of the images. Finally, you need a process to curate images, so that old and obsolete images aren't used inadvertently.

Compute Engine provides image families (/compute/docs/images#image_families) to help you ensure that your automation systems and users run the latest images. As an administrator, you can group a set of images that belong to the same application or use case as an image family. Then users of the images only have to keep track of the image-family name, rather than an exact image name. Because image names must be unique, image-build pipelines often create image names with information encoded in them such as the application name, date, and version, for example, `my-application-v3-20161011`. Rather than changing automated tools to direct consumers to the latest image by propagating the specific name to other systems, you can simply reference the image family name, which will always return the latest image in the family, for example, `my-application`.


Base Image
Image Build Process
Customized Image v4
Customized Image v3 DEPRECATED
Customized Image v2 OBSOLETE
Customized Image v1 DELETED
Image Family my-application

To add an image to an image family, or to create an image family if one doesn't exist, you must add an additional `--family` flag to the image create step, for example:

After you run this command, any calls to run an instance based on the image `my-application` will point to the newly created image, `my-application-v3-20161011`.

As an administrator, you can also roll back the image (/compute/docs/images/create-delete-deprecate-private-images#setting_families) that the image family points to by deprecating the image using the following command:

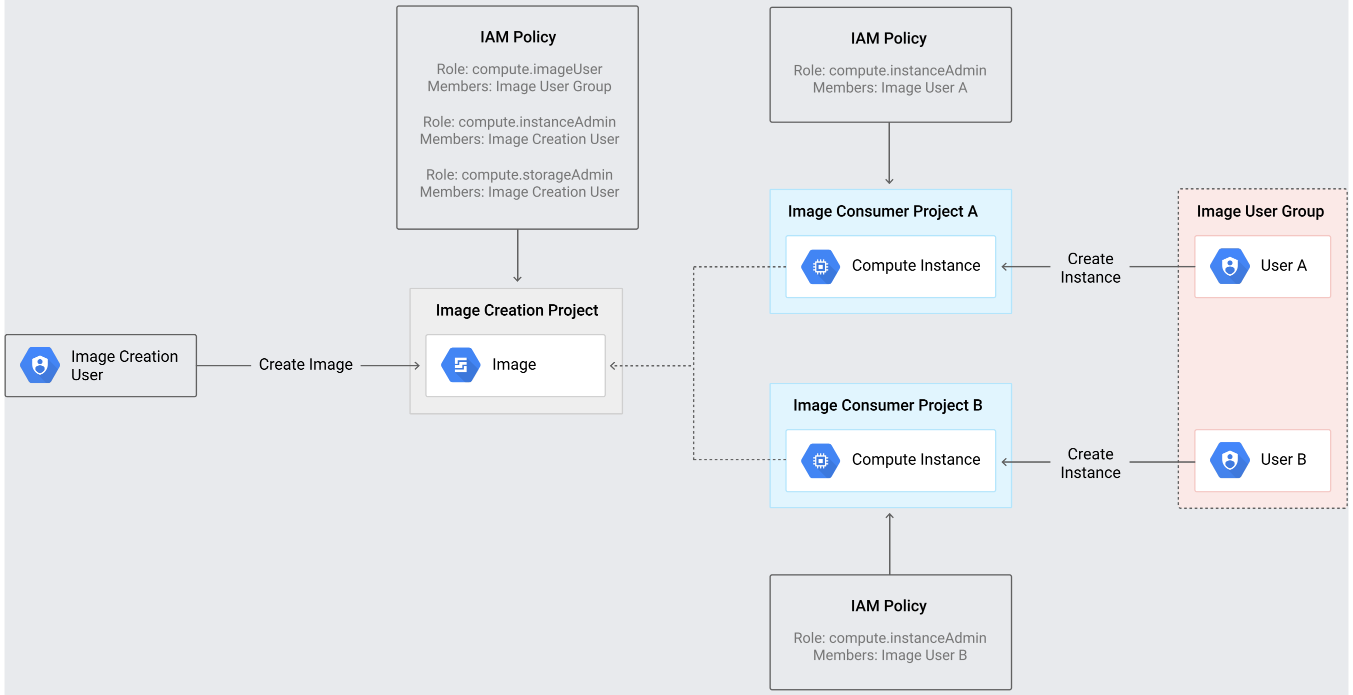
You can choose from various states of deprecation:

State	Description
DEPRECATED	Images that are no longer the latest, but can still be launched by users. Users will see a warning at launch that they are no longer using the most recent image.
OBSOLETE	Images that should not be launched by users or automation. An attempt to create an instance from these images will fail. You can use this image state to archive images so their data is still available when mounted as a non-boot disk.
DELETED	Images that have already been deleted or are marked for deletion in the future. These cannot be launched, and you should delete them as soon as possible.

You can mark images for deletion or obsolescence by using the `gcloud compute images deprecate` command. You can attach metadata to images to mark them for future deletion by providing one of the `--delete-in` or `--delete-on` flags. To attach metadata to mark images for future obsolescence, provide the `--obsolete-in` or `--obsolete-on` flags. You can incorporate this command into an image-build process to enforce an image-lifecycle policy that restricts the proliferation of stale and expired images in your project. For example, at the end of your image-build pipeline, you could include an additional check for images that need to be deprecated or deleted and then perform those actions explicitly.

While deprecated and deleted images are no longer shown through the API and UI by default, you can still see them by providing the `--show-deprecated` flag. To completely delete the image and its data, you must send an explicit delete command (/sdk/gcloud/reference/compute/images/delete) for that image.

Organizations often create multiple Google Cloud projects to partition their resources, environments, and user access. Isolating resources into projects allows for granular billing, security enforcement, and segregated networking. Although most cloud resources do not need to span multiple projects, images are good candidates for sharing across projects. By using a shared set of images, you can follow a common process to deliver images with best practices for security, authorization, package management, and operations pre-configured for the rest of the organization.



You share images by assigning IAM roles (/compute/docs/access/iam) to an organization's projects. The project that contains the images you want to share with other projects, referred to in the preceding diagram as the "Image Creation Project," must have the following IAM roles and policies applied:

1. Allow users of the "Image User Group" to create instances from these images by granting them the `compute.imageUser` role.
2. Allow the "Image Creation User" to create instances in this project by granting them the `compute.instanceAdmin` role.
3. Allow the "Image Creation User" to create images and disks in this project by granting them the `compute.storageAdmin` role.

Projects that you want to be able to use the shared images must allow users with the `compute.imageUser` role to create instances by assigning them the `compute.instanceAdmin` role.

For more detailed instructions on sharing images between projects, see Sharing images across projects in the Compute Engine documentation. (/compute/docs/images/sharing-images-across-projects)

- Customizing root disks and images (/compute/docs/disks/create-root-persistent-disks)
- Manually importing virtual disks (/compute/docs/images/import-existing-image)
- Creating, deleting, and deprecating images (/compute/docs/images/create-delete-deprecate-private-images)
- Sharing images across projects (/compute/docs/images/sharing-images-across-projects)
- Try out other Google Cloud features for yourself. Have a look at our tutorials (/docs/tutorials).