13
Image Hardening and CI/CD Security

In this chapter, we will look at Google's approach to Compute Engine image hardening and DevOps pipeline security. One of the most critical issues facing industries today is software supply chain attacks. To address this matter in the cloud, we need to be able to build secure infrastructure, monitor operations, and fix vulnerabilities. This is a very broad topic. We will only cover the topics required for the exam in this chapter.

In this chapter, we will cover the following topics:

- Overview of image management
- Custom images for Compute Engine
- Image management pipeline
- Controlling access to images
- Image lifecycle
- Enforcing lifecycle policies
- Secure CI/CD pipeline
- Best practices for a CI/CD pipeline
- Shielded VMs
- Confidential computing

Overview of image management

A Google Compute Engine image is a pre-configured **virtual machine** (**VM**) image that can be used to quickly deploy applications and services to Google Compute Engine. An image includes the operating system, configuration settings, and any other software required for the application or service to run. Securing a Google Compute Engine image is important to be able to prevent unauthorized users and malicious software from accessing applications and services. This can be done by setting up strong access control measures with **identity and Access Management** (**IAM**) tools, creating secure user accounts, and using encryption and authentication measures. Regularly updating the image with the latest security patches and using intrusion detection and prevention systems can also help protect the image from potential security threats. Let's understand the concept of image management used for hardening a Compute Engine image.

An image in Compute Engine refers to an immutable disk image. This is not to be confused with a container or Docker image. We will cover container image security in *Chapter 15*, *Container Security*. The disk image is used to create a VM. Let's take a look at *Figure 13.1*.

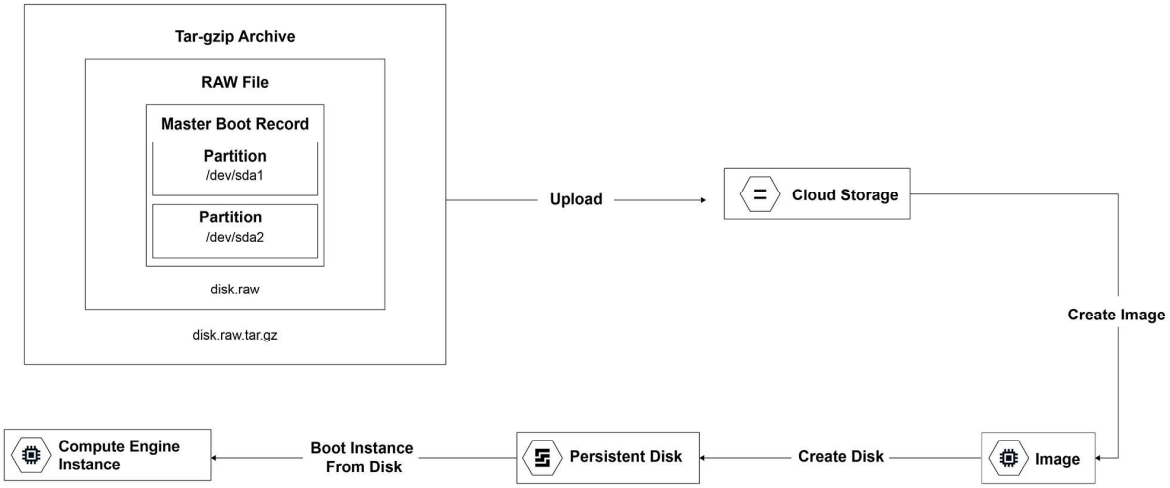

Figure 13.1 – A pipeline for creating a VM image

A machine image includes all of the configuration, metadata, permissions, and data from multiple disks of a VM instance. This information may be retrieved quickly and easily. Performing system maintenance, creating backups and recoveries, and cloning instances are just some of the many uses for machine images.

The following data from the source instance is captured by a machine image:

- Type of machine
- Metadata of instance

- Labels
- Network tags
- Preventative maintenance plan
- Volume mapping
- Hard drive information that persists across reboots
- Where relevant, **Unified Extensible Firmware Interface** (**UEFI**) settings
- A machine image does not capture data in memory and in the local SSD. However, local SSD device mapping and information about the original instance, such as its name or IP address, can be captured.

There are two types of image options available within the Google Cloud environment:

- **Public images**: Google, some open source groups, or third-party businesses produce and maintain these images. These images are accessible to all Google projects; no further IAM permissions are required. These can be downloaded for free. Some of the images are premium, for which you must pay. A list of public and premium images can be found in the Google Cloud documentation.
- **Custom images**: These are your images, and they are only accessible through the cloud project in which you created them. By granting appropriate IAM permissions, you can share these with other cloud projects. These incur a cloud storage fee, but there is no usage fee in Compute Engine.

Let's next explore more details about custom images and the options available, both manual and automated, to bake images.

Custom images for Google Compute Engine

It is possible to customize your Compute Engine instance by way of a startup script, which allows you to install the necessary software components and harden the image. An efficient method is to create a custom image with your specific needs of security configurations and image standards. There are three ways you can customize an image. The process of customization is generally referred to as image baking:

- Manual baking
- Automated baking
- Import an existing image

The images needs to be encrypted and that can be achieved either by using Google's default encryption keys or by using your own encryptions keys (customer-managed encryption keys).

Changing or baking images offers the following benefits:

- Shorter boot time for applications to be ready
- A more stable environment for applications to be installed
- Easier to go back to older versions of the image
- During application startup, there are fewer external services that need to be used
- Scaling up compute instances that have identical software versions

Manual baking

When you start the process of establishing a new VM instance from a public image, you will have the opportunity to generate a custom image for that instance. You are then able to customize the newly created instance with the applications and settings of your choosing once the new instance has been created. After you have finished configuring those features, you will be able to create a custom image based on that instance.

Automated baking

If you only have a few images that need to be managed, manual baking is an easy method to get started. If you have many custom images, however, employing the manual technique of management becomes more challenging and difficult to audit. HashiCorp Packer is a free and open source program that makes the process of creating images reproducible, auditable, and configurable.

To learn more about creating an automated image-creation pipeline, visit the link provided in the *Further reading* section. It explains how to use Jenkins, Packer, and Kubernetes to build images. You can also use Packer as part of a Spinnaker pipeline to generate images.

Importing existing images

You can migrate images by exporting them from your existing cloud infrastructure and moving them to Google Cloud Compute Engine. Another way to get your old images onto your new computer is to use Google Cloud Migrate for Compute Engine.

Migrate for Compute Engine is a suite of tools and a service that streamlines the process of migrating machines from one platform to another with minimal disruption. This is made possible via continuous replication at the block level. Creating images can be done by manually baking on Compute Engine after migrating your VMs there.

Encrypting images

By default, Google's encryption keys are used to encrypt all Compute Engine disks. Disk images are likewise encrypted during creation. You also have the option of using your own encryption keys throughout the disk creation process. If you want to produce an encrypted image after you've made the disk, you'll need to supply the relevant command with the encryption keys. **Encryption at rest** in the Google Cloud documentation has more details about using customer-supplied encryption keys. Images that are made from disks are also protected by encryption. Alternatively, you can provide your own encryption keys when you create the disks. Then, after you create the disk, you can create an encrypted image by giving the image command the encryption keys you want.

Please refer to the encryption at rest documentation link in the *Further reading* section. This document provides an overview of encryption at rest features available in Google Cloud. It explains why encryption is important, how it works, and what tools are available to make encryption easier to manage.

Image management pipeline

In this section, we will look at the process of creating an image management pipeline. The process involves a few procedures, and we will look at the high-level steps, including prerequisites that you need to meet before you build your image factory. Let's now look at the steps:

1. Create a strategy for managing image lifecycles, such as golden image promotion, versioning, deprecation, and deletion.
2. Bake a golden image, and use an automated compliance check, such as Chef InSpec.
3. Ensure that an organization policy is in place to limit the compute images available to individuals working on specific projects.

You will also need to meet a couple of prerequisites:

- Ensure you have a Google Cloud account.
- You either need to have Project Editor access for an existing project, or you need organization permissions to be able to create a new project under the organization.

The process of creating an image is quite similar to the process of creating source code. You will have access to a repository that stores the base image as well as the templates that you will need to construct an image build. The following is a tutorial that will walk you through the process of building an image.

Creating a VM image using Packer and Cloud Build

In this section, we will go over how to create a Google Cloud VM image using HashiCorp Packer. This process will allow you to set up automation. We will go over the process and how the pieces fit together. You do not need to use Cloud Build or third-party products such as Packer for the exam, but it is important to know how to create an automated VM image pipeline.

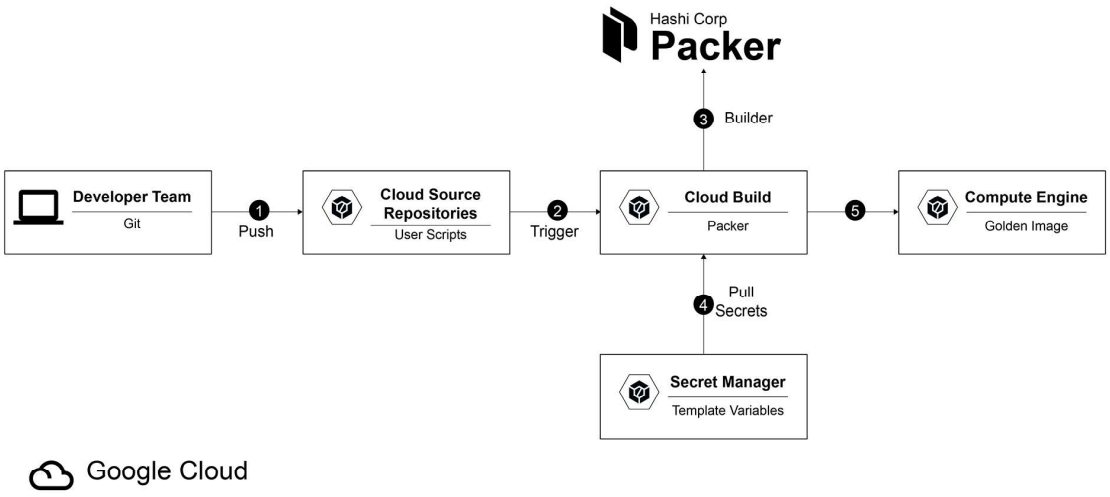

Figure 13.2 – VM image creation using Packer

Let us analyze the components of the architecture seen in *Figure 13.2*, and then outline the steps required to build an automated image factory with Packer. Prior to starting, it is important to have an understanding of Packer. The exam does not include Packer, yet these steps are included to explain the automated VM image creation process.

Before you begin, there are the following prerequisites:

- A Google Cloud account
- A Packer template
- The Packer binary
- The Google Cloud SDK

Step 1: Creating an infrastructure for the image creation

First, you need to create an infrastructure for the image creation, and for that, we need to follow these steps:

1. Create a Compute Engine instance to use as a builder.
2. Create a Cloud Storage bucket to store the image.
3. Create a service account with the necessary permissions.

Step 2: Creating the Packer template

Now that your infrastructure is in place, you can create the Packer template. The Packer template is a JSON file that defines the configuration of your image. It should include the following information:

- Builder type: `googlecompute`
- Source image
- Zone
- Machine type
- Boot commands
- Post-processing

Here is an example of a JSON Packer template:

```
{
   "variables": {
      "project": "{{ env `GCE_PROJECT` }}",
      "zone": "{{ env `GCE_ZONE` }}",
      "machine_type": "{{ env `GCE_MACHINE_TYPE` }}",
      "disk_image_name": "{{ env `GCE_DISK_IMAGE_NAME` }}",
      "disk_image_project": "{{ env `GCE_DISK_IMAGE_PROJECT` }}"
   },
   "builders": [
      {
         "type": "googlecompute",
         "project_id": "{{ user `project` }}",
         "source_image_family": "{{ user `disk_image_name` }}",
         "zone": "{{ user `zone` }}",
         "disk_size": "10",
         "disk_type": "pd-ssd",
         "image_family": "{{ user `disk_image_name` }}",
         "image_project": "{{ user `disk_image_project` }}",
```

```
            "machine_type": "{{ user `machine_type` }}"
        }
    ],
    "provisioners": [
        {
            "type": "shell",
            "script": "install.sh"
        }
    ]
}
```

The JSON Packer template is provided as an illustration only; the intent is to give a simplistic example. For an individual use case, you will need to modify a template or create a new template.

Step 3: Installing the Packer binary

Once you have the Packer template ready, you need to install the Packer binary. This is a command-line utility that you can use to create images with Packer. To install the Packer binary, follow these steps:

1. Download the Packer binary from the official download site.
2. Extract the binary and copy it to the preferred location.
3. Add the binary to your `PATH` environment variable.

Step 4: Creating the image

Once you have the Packer binary installed, you can use it to create the image. To do this, you need to run the following command:

```
packer build -var-file=<variable-file> <packer-template>
```

This command will use the Packer template and the variable file to create the image. Once the image is created, it will be stored in the Cloud Storage bucket you created in *Step 1*.

Step 5: Automating image creation with Cloud Build

The last step is to automate image creation with Cloud Build. Cloud Build is a managed build service on Google Cloud that can be used to automate your build process. To use Cloud Build to automate image creation, you need to create a build configuration file. The configuration file should specify the following information:

- **Source repository**: The location of the Packer template and the variable file
- **Steps**: The commands that Cloud Build should execute
- **Trigger**: The events that will trigger the build process

Once you have the configuration file ready, you can upload it to Cloud Build and configure the triggers. After that, Cloud Build will take care of the image creation process and you will have a fully automated VM image creation pipeline.

Using this image factory, you can now automatically create new images from a Cloud Source repository every time a new tag is pushed to that repository.

> **Note**
>
> The full tutorial can be obtained from the Google Cloud community: `https://packt.link/qTlcv`. A corresponding tutorial can also be found on GitHub: `https://packt.link/02Rgp`.

Controlling access to the images

To split workloads, environments, and user access, you will usually need different Google Cloud projects. Although most cloud workloads do not require sharing between projects, images are an excellent choice for doing so. You can follow a consistent approach to distributing images with best practices for security, permission, package management, and operations pre-configured for the rest of the company by using a shared collection of images.

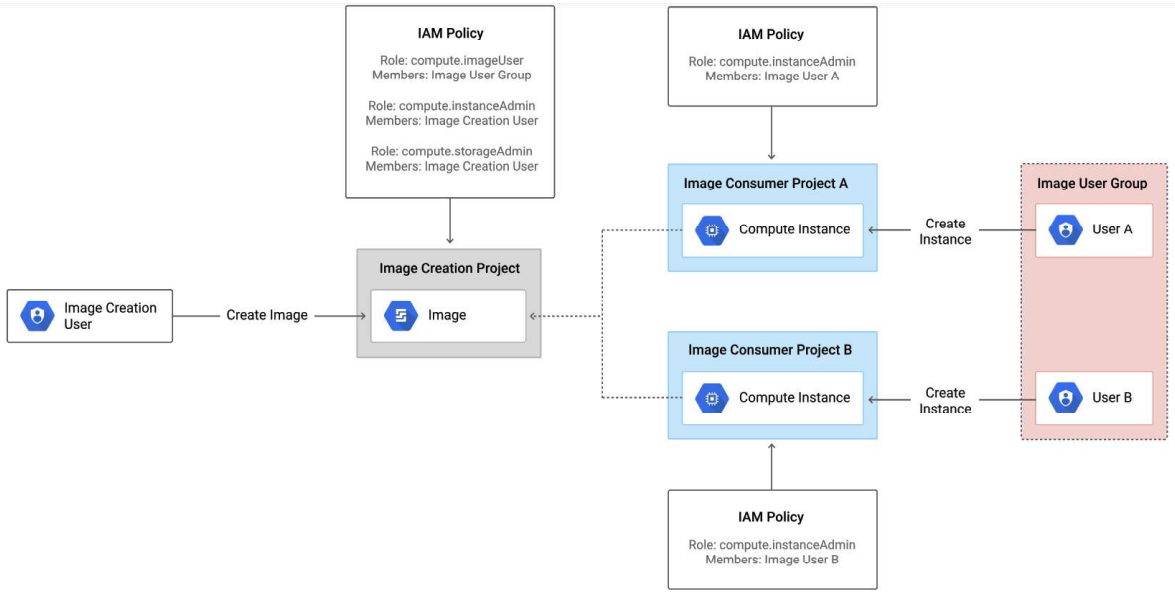

Figure 13.3 – Sharing images between projects

You can share images by giving different IAM roles to different projects inside an organization. **Image Creation Project**, represented in *Figure 13.3*, is the project that holds the images that you want to share with other projects, and it must have the following IAM roles and policies applied to it:

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

In order for your projects to be able to use the shared images, you will need to ensure that users have access to the `compute.image` API. The `compute.imageUser` role allows other users to create instances by giving them the `compute.instanceAdmin` role.

Image lifecycle

After you've built up an image build pipeline, you'll need to keep the images up to date. While the pipeline creates the images, you must make sure that your deployment techniques use the most recent versions. You'll also need a method for curating images so that no outdated or obsolete images are accidentally used. With the image lifecycle, you can achieve that. Next, we will look at image families, which we covered earlier, in the *Overview of image management* section, that is, public image families and custom image families.

Image families

With Compute Engine's image families, you can rest assured that your automation systems will always be able to make use of the most up-to-date images. An image collection can be organized into a *family* by the administrator. Then, instead of remembering specific image names, users will just need to remember the name of the image family. Since every image must have a distinct name, image-build pipelines frequently generate names for images that include information about the program, the date, and the version, such as `my-application-v3-20210101`. Instead of always having to change the image's name in automation tools, you may just refer to the family name. Having the most up-to-date version of an image, such as `my-application`, is a breeze when you use image families.

There are distinct categories for the various types of public images. Every member of the public image family always links to the most up-to-date version of an image in their respective region. The zonal fault tolerance of your workflows is improved by the fact that newly released images are made available in image families at different times across different zones upon their initial global release.

Using a custom image family, you can create your own images. The image used to create the image family is linked to the most up-to-date version of that image. Deprecating the most up-to-date image in a family is one way to revert to an older version of the family's images (if the previous image is not deprecated).

You may more easily manage the images in your project with the assistance of image families, which help group images that are related together and allow you to roll ahead and back between specific image versions.

Deprecating an image

You also have the ability, as an administrator, to roll back the image to which the image family is pointing by deprecating the image and entering the following command:

```
gcloud compute images deprecate nginx-v3-20191019 --state DEPRECATED
```

The following table shows the various stages of deprecation you can choose from:

State	Description
`DEPRECATED`	Although the image is marked as `DEPRECATED`, it can still be used for creating a VM. However, new links to this image are not allowed. Additionally, image families no longer refer to this image, even if it is the most up-to-date image within the family.
`OBSOLETE`	The image has been deemed obsolete and is no longer available for use. Attempts to use this image in a request will result in an error message. Links to this image that already exist are still valid.
`DELETED`	If you try to use a deleted image, an error message will be returned.

Table 13.1 – States of image deprecation

We will now move on to the next section and look at how to enforce the lifecycle policies for your VM images.

Enforcing lifecycle policies

Lifecycle policies for VM images are necessary to ensure that images are updated regularly with the latest security patches and features. By enforcing lifecycle policies, organizations can ensure that their VMs remain secure and up to date, thereby reducing the risk of security breaches and other problems due to outdated software. Additionally, regular updates can help improve the performance and reliability of VMs, thus increasing their overall efficiency and cost-effectiveness.

Google Cloud VM lifecycle policies allow administrators to define a set of rules for how their VMs should be handled over time. These policies are important for ensuring that images are regularly updated, properly configured, and can be easily identified for deletion or obsolescence.

The most basic policy is the image deletion policy, which sets the time frame for when an image should be deleted. This helps to keep images up to date and avoid any potential security vulnerabilities caused by outdated images.

The image retirement policy allows administrators to set a time frame for when an image should be retired and no longer used. This helps to avoid any potential problems caused by outdated images. Additionally, administrators can use the image deprecation policy to mark images that they no longer use, allowing them to easily identify these images for deletion or obsolescence.

These policies are important for helping administrators to keep their VMs up to date and ensure that they can easily identify images for deletion or obsolescence.

Let's look at an example where you can mark an image for deletion or obsolescence; you can do so by using the following command:

```
gcloud compute images deprecate
```

You can also 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 may also use this command as part of an image-build process to implement an image lifecycle policy that prevents your project from accumulating stale or expired images. You could, for example, implement an additional check for images that need to be deprecated or destroyed at the conclusion of your image-build process, and then conduct those actions directly.

While deprecated and deleted images are no longer visible through the API and UI by default, the `--show-deprecated` parameter allows you to see them. You must explicitly use the `delete` command for that image to entirely erase it and its data.

Securing a CI/CD pipeline

The Continuous Integration and Continuous Delivery (CI/CD) pipeline is a set of procedures that software developers follow in order to work on smaller chunks of code and increase overall productivity and efficiency. Its goal is to expose errors as early as possible in the process, allowing for faster failures. Developers typically activate CI processes by pushing code changes. Linting, testing, and building are all processes in the pipeline that validate the modifications. Typically, a CI pipeline produces an artifact that may be deployed later in the deployment process. The CI/CD pipeline is the DevOps processes' foundational infrastructure. It is critical to secure the CI/CD pipeline at every step to ensure that the applications, services, and data in the cloud are secure and protected. Securing your CI/CD

pipeline helps to reduce the risk of malicious attacks, data breaches, and other security vulnerabilities. Additionally, it can help organizations to improve the overall efficiency of their cloud deployments by ensuring that only necessary code changes, configurations, and updates are applied.

CI/CD security

Securing your CI/CD pipeline is essential for ensuring the integrity, availability, and confidentiality of the application and its data. Without proper security controls in place, a pipeline can be vulnerable to malicious attacks, which can lead to data breaches and other security incidents. Implementing secure configuration guidelines, automating security testing, implementing role-based access controls, and monitoring and auditing logs are all important steps that can help protect the pipeline and its components. Let us look at some guidelines for incorporating security into CI/CD:

- Secure configuration guidelines should be established to ensure that security settings are properly configured and maintained across all components in the CI/CD pipeline. These guidelines should include requirements for authentication, authorization, encryption, logging, and other security-related settings.
- Automated security testing should be integrated into the CI/CD pipeline to detect and address security vulnerabilities. This can include static and dynamic analysis, penetration testing, and vulnerability scanning.
- Role-based access controls should be implemented to restrict access to the CI/CD pipeline and its components. This should include access control policies that define who can access and modify the pipeline and its components.
- Logs should be monitored and audited regularly to detect any suspicious activity. This includes monitoring access to and modification of the pipeline and its components, as well as any errors or exceptions that occur.

The primary focus in safeguarding a CI/CD pipeline should be to ensure that all potential security flaws are addressed at every step the code takes from inception to deployment.

CI/CD security threats

CI/CD are powerful tools for increasing the speed and efficiency of software development. However, as with any technology, there are security threats that must be taken into consideration when using these tools. These threats can range from unauthorized access to data leakage and malware injection. In addition, poorly configured CI/CD pipelines can introduce security risks, as can insufficient logging and monitoring, and inadequate change management. By understanding these threats and taking the appropriate steps to protect against them, organizations can ensure that their CI/CD processes remain secure and reliable. Here are a few examples:

- Unauthorized access to the CI/CD pipeline can lead to malicious code being injected and confidential information, such as credentials, being stolen.

- Without proper testing and security scanning, malicious code can be deployed and security threats can be introduced into the system.
- Without patching, vulnerabilities can be exploited to gain access to the system, leading to data loss and system disruption.
- Poor user access controls can lead to users being able to access more than they should, resulting in malicious code being injected or confidential information being stolen.
- Poorly configured CI/CD pipelines can lead to insecure code being deployed or private information being exposed.
- Data leakage can occur when sensitive information such as credentials, tokens, or passwords is exposed to unauthorized individuals.
- Malware can be injected into the system and can be used to steal confidential information or disrupt the system.

This is not an exhaustive list of CI/CD security concerns—there are certainly more—but the goal here is to identify a few key ones. Let's take a look at how you can further secure your CI/CD pipeline.

How to secure a CI/CD pipeline

Securing a CI/CD pipeline is essential to ensure the safety and reliability of code. Access control policies and encryption algorithms should be applied to the source code repository. Authentication and authorization mechanisms should be used to protect the delivery pipeline. Firewalls and network segmentation should be used to secure the infrastructure. Secure scripting and testing tools should be used to check for any malicious code. Logging and analytics tools should be used to monitor the whole pipeline. All these measures help to ensure that the CI/CD pipeline is secure and reliable.

Source Composition Analysis (SCA)

Source Composition Analysis (**SCA**) is a process used in CI/CD pipelines to assess the security posture of the source code used in development. SCA runs scans on the source code to identify vulnerabilities and security issues. SCA is often used to detect any malicious or unauthorized code that may have been introduced into the development process. SCA should be placed near the beginning of the CI/CD pipeline, ideally before the code is even checked in, in order to get an initial assessment of the security posture of the code before it is released to the public.

Static Application Security Testing (SAST)

Static Application Security Testing (**SAST**) is a process of analyzing the source code of an application for potential security vulnerabilities. It is typically done before the application is deployed, and it is the first step of the security testing process. It can detect security issues in the source code, such as missing input validation, hardcoded credentials, SQL injection, and cross-site scripting vulnerabilities.

SAST can be used to identify vulnerabilities in any application, regardless of the technology used or the development methodology.

SAST should be included in the CI/CD pipeline in order to ensure that vulnerabilities are discovered and addressed prior to the application being deployed. It should be done at the beginning of the pipeline, before any integration or deployment steps, and should be repeated after any code changes are made. This will ensure that any newly introduced vulnerabilities are identified and addressed before the application is made available to users.

CI/CD IAM controls

IAM controls are an essential part of ensuring the security of a CI/CD pipeline. They are responsible for protecting access to the systems and resources within the CI/CD pipeline, as well as providing secure authentication and authorization for users. IAM controls can be used to ensure that only authorized users have access to the necessary systems and resources to ensure a secure CI/CD pipeline. Additionally, IAM controls can help to ensure that only users with the correct permissions are able to make changes to the pipeline. By providing secure access control and authentication, IAM controls can help to reduce the risk of unauthorized access and malicious activity within the CI/CD pipeline.

To mitigate the risks posed by a lack of IAM controls in a CI/CD pipeline, it is important to ensure that all users and systems within the pipeline are adequately authenticated and authorized. This can be done by implementing strong access control and authentication measures, such as multi-factor authentication and role-based access control. Additionally, it is important to ensure that all users are assigned to the appropriate roles based on their access needs and that all access is regularly monitored and reviewed. Finally, it is important to ensure that all users are routinely trained on best practices for security and that all security policies are regularly updated and enforced.

Secrets management and CI/CD

Passwords and access keys are frequently required at many phases of the CI/CD process, such as when integrating or building code, deploying apps in testing or production, and so on.

Hardcoded secrets are easily accessible to anyone with access to configuration files or **Infrastructure as Code** (**IaC**) templates, posing a serious security concern.

A preferable method is to keep sensitive data in Google Cloud Secret Manager and communicate it as needed throughout CI/CD operations using variables.

Container registry scanning

Container registry scanning is an important part of the CI/CD pipeline security process. It is the process of scanning container images in a container registry to detect and prevent the introduction of any malicious code that could lead to a security breach. By scanning the container images and ensuring

their integrity, organizations can ensure that only trusted images are deployed in their environment, helping to protect their application and infrastructure from malicious code injection.

Container registry scanning also enables organizations to detect and identify any vulnerabilities in the images, allowing them to take corrective action before any malicious code is deployed. In addition, it can help to detect any changes made to the images over time, allowing organizations to track the source of any changes and take appropriate action if necessary.

Container runtime security

The security of a container runtime is an essential element of a CI/CD pipeline. Containers are an efficient way to package and deploy applications and offer numerous advantages over traditional virtualization. Containers must be secured to protect applications from malicious actors, unauthorized access, and other security risks.

Container runtime security ensures that the container environment is secure and that any malicious code or malicious actors do not have access to the underlying host system. Container security also helps to ensure that applications are not exposed to a variety of potential threats, such as data breaches, denial-of-service attacks, and more. Furthermore, it also ensures that applications are running in an optimal environment, with all necessary security measures in place. By incorporating container runtime security into a CI/CD pipeline, organizations can maximize their security posture while also ensuring that applications are being deployed quickly and efficiently.

Establishing trust in CI/CD

Trust is when your business believes that someone or something is reliable, honest, able, or strong. You can keep track of both implicit and explicit trust. Both implicit and explicit trust have different weights in terms of confidence when they are talked about outside of the pipeline. Because implicit trust can't be checked, an artifact in a repository might or might not have been pushed through a security-aware CI/CD pipeline. If trust is explicitly conveyed, an audit trail can be made for each item that can be checked. This makes it possible to build trust outside of the pipeline environment.

An example of implicitly recorded trust is when a **quality assurance** (**QA**) team confirms the quality of a software release version and tells others that they validated the build, but doesn't document the validation. An example of explicitly recorded trust is when a QA team checks the quality of a software release version and then writes down what they did. They could, for example, send an email or make a digital note of the date, time, and situation of a successful operation. Trust must be written down so that you can confidently rebuild the series of stages. Getting more trust is the same as keeping track of trust. Over time, trust is built by going through a series of steps. As an item moves from a lower-level environment into production, its trust increases. Each stage of the pipeline includes tasks such as running an automated test suite or manually validating code through exploratory testing. After the successful completion of these tasks, digital evidence that has been cryptographically signed to prove its authenticity can be linked to the resulting artifacts.

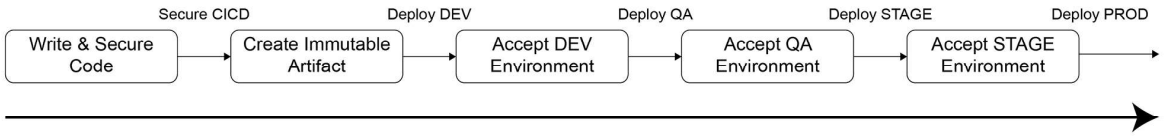

Figure 13.4 – Artifact trust

As illustrated in the preceding diagram, trust is seen as follows:

- When feature development is complete, trust is implicit.
- When code security tools are complete, trust is stated explicitly.
- When DEV members accept the DEV environment, trust is implicit.
- When QA approves one or more releases, trust is stated explicitly.
- When stakeholders accept the staging environment, trust is stated explicitly.

Next, we will look at binary authorization, which is a security control that enables organizations to verify the integrity of their container images prior to deployment. It is an important security measure that allows organizations to ensure that only trusted, approved images are being used in their production environments.

Binary authorization

Attestations of successful events can be combined to show the quality, security, and functionality of the artifact. The Binary Authorization service from Google is a security control that can be used when a program is being deployed in **Google Kubernetes Engine** (**GKE**). It helps ensure that only trusted container images are used.

The Binary Authorization toolset is made up of a GKE cluster, an admission controller, a structured policy document protocol, and programmatic access to the digital attestations that go with a given container.

Policies can be developed to allow or reject deployment based on the presence or lack of attestations. Diverse policies can be developed for different contexts because policies are applied at the cluster level. As environments come closer to production, each cluster can require a higher level of attestation.

CI/CD pipelines can automate the process, codify trust, and increase confidence, but they are not risk-free. Because artifacts are pulled from a repository outside of the pipeline, a hacked item could be sent out without any more security checks. The goal of binary permission policies is to reduce this risk. They should be used to restrict deployments to files from a known set of repositories.

Establishing confidence with attestations in the items traveling through pipelines is important to pipeline security. Docker image attestation is the process used for verifying the provenance and integrity of Docker images in the pipeline. It is a secure process used to confirm the contents of a Docker image. It works by verifying that the contents of the image are approved and safe to use. The process begins by

scanning the image for known vulnerabilities and malware. If any issues are found, they are reported to the user, and the image is marked as not approved. Next, the image is compared to a list of approved images and verified to be the same version. Finally, a signature is generated to ensure that all the contents of the image are unaltered. This signature is then used to verify the integrity of the image at any point during its lifetime. Obtaining these attestations creates a verifiable chain of trust, allowing you to make an informed decision about deploying an artifact to a certain cluster or environment.

Best practices for CI/CD security

The foundation of DevOps is CI/CD. Through automated monitoring and processes throughout the development cycle, CI/CD provides value to software production. In this rapidly evolving, technology-driven environment, security must be balanced with the need for flexibility. To prevent data breaches, the best approach is to build security into the development process.

Here are some of the best practices for securing CI/CD pipelines:

- Use IaC to manage and provision IT resources on Google Cloud. IaC uses configuration files and automation tools to manage resources to ensure that resources are set up correctly and efficiently.
- Automate builds and deployments with tools such as Jenkins, CircleCI, and Spinnaker to define and execute the necessary steps in the process. These automation tools make it easy to quickly deploy code changes and reduce the risk of potential bugs and are essential for a successful CI/CD pipeline.
- Using Git for version control is essential to the success of a project. Git is the most commonly used version control system and helps to keep track of changes and maintain consistency throughout the project's lifecycle. By using Git, teams are able to easily collaborate, share their work, and resolve conflicts quickly. Additionally, Git allows for easy rollbacks and provides a secure environment for all stakeholders involved.
- Use notifications and alerts to identify any problems in the CI/CD pipeline and notify team members of any significant changes or updates. This is particularly essential when rolling out to production.
- Track performance regularly by using monitoring tools such as Google Cloud's operations suite to make sure your pipeline runs efficiently. With these tools, you can detect any potential issues and take the necessary steps to solve them.
- Perform testing frequently throughout the CI/CD pipeline to ensure the code is functioning as expected. Automated and manual tests should be used to validate the application code and confirm whether the application is working properly.
- Use cloud-native tools such as Kubernetes and Cloud Build to simplify deploying applications and managing the CI/CD pipeline on Google Cloud.

- Adopting containerization can help make application deployment more efficient and reliable. It isolates applications into containers, enabling them to be quickly and easily deployed to multiple cloud environments. This reduces the time and resources needed to deploy applications, while also making it simpler to manage and scale applications. Additionally, containerization leads to increased resource utilization, making it possible to get more out of existing resources.
- Ensure that changes to the CI/CD pipeline are reversible to support automated rollbacks. If any issues arise, changes that have been made can be quickly reverted.
- Enable authentication and authorization in the CI/CD pipeline using a centralized authentication system such as Google Cloud IAM. This will ensure efficient management of access to the CI/CD pipeline.
- Implement a security monitoring system to audit all CI/CD activities and detect any suspicious behavior. This will enable the organization to be aware of any potential threats and react accordingly.
- Implement the principle of least privilege access by utilizing Google Cloud IAM to define roles and assign users with the minimum necessary access level to the CI/CD pipeline. This will help ensure that only authorized personnel can make changes to the environment, and any unauthorized access attempts can be readily identified.
- Ensure the security of your configuration by utilizing Google Cloud Security Command Center to detect any misconfigurations and enforce secure configurations.
- Monitoring and patching of vulnerable components should be done using Google Cloud Security Scanner or any other security scanners to identify and fix any weak points in the CI/CD pipeline. Doing so will ensure that the pipeline remains secure and free from any potential risks.
- It is important to regularly back up the CI/CD pipeline so that any modifications can be undone if needed. This will help to ensure that any mistakes or issues that arise can be easily rectified.
- Encryption is a must for safeguarding data, both when it's in motion and when it's at rest. Encrypting data will prevent unauthorized access and protect it from potential cyber-attacks. Doing so will also help to ensure that any information that is shared is kept confidential and secure.
- Utilize containers to separate the CI/CD pipeline and its elements. Containers are a great way to ensure that the pipeline and its components are adequately isolated from each other. This helps to guarantee that the pipeline runs smoothly and provides a stable environment for the components. Additionally, using containers makes it easier to troubleshoot any issues that may arise during the CI/CD process.
- Set up a threat detection system. Establish a system to detect potential threats and trigger notifications when malicious activity is identified.
- Take advantage of Google Cloud's security offerings to secure your CI/CD pipeline. These include Cloud Armor, Cloud Security Scanner, and Cloud IAM.

This concludes the best practices for CI/CD security. Next, we'll look at Shielded VMs, what they are, and how and where to use them.

Shielded VMs

Shielded VMs on Google Cloud are protected against rootkits and bootkits by a set of security safeguards. Shielded VMs safeguard company workloads from dangers such as remote attacks, privilege escalation, and hostile insiders.

Shielded VMs' verifiable integrity is achieved using the following features:

- Secure Boot
- **Virtual trusted platform module** (**vTPM**)-enabled Measured Boot
- Integrity monitoring

Let us look at each of them in more detail.

Secure Boot

Secure Boot checks all boot components' digital signatures and stops the booting process if the signature verification fails. Let's look at how Secure Boot for Shielded VMs works.

Shielded VM instances run software that has been certified and confirmed by Google's Certificate Authority Service. This makes sure that the firmware of the instance hasn't been changed and gives Secure Boot its foundation of trust. The UEFI 2.3.1 firmware protects the certificates that include the keys that software vendors use to sign the system firmware, the system boot loader, and any binaries they load. Shielded VM instances use UEFI firmware.

The UEFI firmware checks each boot component's digital signature against a secure repository of permitted keys at each boot. Any boot component that isn't signed properly, or at all, isn't allowed to run.

If this happens, an entry will appear in the VM instance's serial console log with the strings **UEFI: Failed to load image** and **Status: Security Violation**, as well as a description of the boot option that failed.

Now that we understand how Secure Boot works, let's take a look at Google's secure chip called a **virtual Trusted Platform Module** (**vTPM**).

Virtual Trusted Platform Module (vTPM)

A vTPM is a specialized computer chip that can be used to protect assets used to authenticate access to your system, such as keys and certificates. Let's look at some key aspects of vTPMs:

- The BoringSSL library is used by the Shielded VM vTPM, which is fully compatible with the **Trusted Computing Group** (**TCG**) library specification 2.0. The BoringCrypto module is used by the BoringSSL library.

> **Note**
> You can refer to NIST Cryptographic Module Validation Program Certificate #3678 for FIPS 140-2 (`https://packt.link/x1N4p`) data on the BoringCrypto module.

- Measured Boot is enabled by the Shielded VM vTPM, which performs the measurements required to generate a known good boot baseline, called the integrity policy baseline. To see if anything has changed, the integrity policy baseline is compared to measurements from successive VM boots.
- The vTPM can also be used to secure secrets by shielding or sealing them.

We will now look at the importance of integrity monitoring and how it helps in understanding the security state of the VM.

Integrity monitoring

Integrity monitoring assists in understanding and making decisions regarding the state of your VMs. Let's look at how the process of integrity monitoring works:

- Measured Boot uses **Platform Configuration Registers** (**PCRs**) to hold information about the components and component load order of both the integrity policy baseline (a known good boot sequence) and the most recent boot sequence for integrity monitoring.
- Integrity monitoring compares the most recent boot measurements to the integrity policy baseline and returns two pass/fail outcomes, one for the early boot sequence and the other for the late boot sequence, depending on whether they match or not:
  - The boot sequence from the start of the UEFI firmware until it delivers control to the bootloader is known as early boot.
  - The boot sequence from the bootloader to the operating system kernel is known as late boot.
- An integrity validation failure occurs if any section of the most recent boot sequence does not match the baseline.

In summary, integrity monitoring for VMs is a service that uses multiple layers of intelligence to help identify and respond to potential security threats. It is an important tool for protecting cloud-hosted VMs from malicious actors, malicious software, and potential configuration vulnerabilities.

IAM authorization

Shielded VM uses Cloud IAM for authorization. The following Compute Engine permissions are used by Shielded VMs:

- `compute.instances.updateShieldedInstanceConfig`: Allows the user to change the Shielded VM options on a VM instance
- `compute.instances.setShieldedInstanceIntegrityPolicy`: Allows the user to update the integrity policy baseline on a VM instance
- `compute.instances.getShieldedInstanceIdentity`: Allows the user to retrieve endorsement key information from the vTPM

Shielded VM permissions are granted to the following Compute Engine roles:

- `roles/compute.instanceAdmin.v1`
- `roles/compute.securityAdmin`

You can also grant Shielded VM permissions to custom roles.

Organization policy constraints for Shielded VMs

You can set the `constraints/compute.requireShieldedVm` organization policy constraint to `True` to require that Compute Engine VM instances created in your organization be Shielded VM instances.

Confidential computing

Confidential computing involves the use of hardware-based **Trusted Execution Environments** (**TEEs**) to protect data while it is being used. TEEs are secure and isolated environments that keep applications and data from being accessed or changed while they are in use. A group called the **Confidential Computing Consortium** came up with this security standard.

The three states of end-to-end encryption are as follows:

- **Encryption at rest**: This protects your data while it is being stored
- **Encryption in transit**: This protects your data when it is moving between two points
- **Encryption in use**: This protects your data while it is being processed

Confidential computing gives you the last piece of end-to-end encryption: encryption in use.

Key features of Google Cloud Confidential Computing

When you use a Confidential VM, your data and applications stay private and encrypted even when they are being used. This is a type of Compute Engine VM. It runs on hosts that have AMD EPYC processors that have AMD **Secure Encrypted Virtualization** (**SEV**). Confidential VMs run on these hosts. Adding SEV to a Confidential VM has the following benefits and features:

- **Isolation**: Encryption keys are made by the AMD **Secure Processor** (**SP**) when a VM is created, and they only live on the AMD **System-on-Chip** (**SOC**). These keys can't even be found by Google, which makes them more private.
- **Attesting**: Confidential VMs use the vTPM. Whenever an AMD SEV-based confidential VM starts up, an event is made that says *launch attestation report*.
- **High performance**: AMD SEV has a lot of power for a lot of complicated tasks. Allowing Confidential VMs has little or no effect on most applications, with a performance loss of only zero to 6% when it is turned on for most applications.

In addition to Compute Engine, the following Google Cloud Confidential Computing services are also available:

- Confidential Google Kubernetes Engine nodes require all of your GKE nodes to use Confidential VMs
- Dataproc Confidential Compute uses Confidential VMs in Dataproc clusters

Benefits of Confidential Computing

Let us look at some of the key benefits of using Google Cloud Confidential Computing:

- Google Cloud's customers shouldn't have to choose between usability, performance, and confidentiality. Google's goal is to make confidential computing easy. The transition to Confidential VMs is seamless—all workloads you run today can run as a Confidential VM without the need to change anything.
- With Confidential VMs, customers do not need to make any changes to their applications. Entire VMs can be run in a confidential manner.
- The VM memory is encrypted by a key per VM and generated in hardware (AMD). The keys are ephemeral and not extractable by anyone, including Google.
- Finally, customers can protect their data end to end, from data-in-transit to data-at-rest, and finally to data-in-use.

With Google Cloud Confidential Computing, organizations can securely process and store data in the cloud without having to worry about it being exposed to third parties. Additionally, it helps organizations comply with regulations such as GDPR, as well as providing an added layer of security when it comes to protecting sensitive data. As more organizations move to the cloud, Google Cloud Confidential Computing provides an important security layer that helps protect their data from unauthorized access.

Summary

This chapter highlights the need for hardening images for both VMs and containers. It outlines the steps for managing images, securing and hardening them, and building image management pipelines. In addition, we discussed the use of cloud-native tools on Google Cloud to build security scanning of the CI/CD pipeline. To further secure applications running in the cloud environment, we explored Google Compute Engine security capabilities such as Shielded VMs, vTPMs, and Confidential Computing. Finally, we discussed various measures for securing containers and applications running inside them, such as network policies to control traffic flow and key management systems for encryption key security and management.

In the next chapter, we will cover Security Command Center, which is used to monitor the security posture of your Google Cloud organization.

Further reading

For more information on Google Cloud image hardening and CI/CD security, refer to the following links:

- Creating a Cloud Build image factory using Packer: `https://packt.link/tr8ks`
- Shifting left on security: `https://packt.link/BWbBN`
- vTPMs: `https://packt.link/JiQqG`
- AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More: `https://packt.link/28XiO`
- Default encryption at rest: `https://packt.link/jcUlr`

14

Security Command Center

In this chapter, we will look at the capabilities of Google's Security Command Center. The exam typically has questions on how to configure and use **Security Command Center** (**SCC**) to monitor the security posture of your Google Cloud organization. You may also get questions on how to detect threats and vulnerabilities in your cloud environment and workloads. This is one of the critical aspects of security operations, so make sure you understand it very well.

In this chapter, we will cover the following topics:

- Overview of SCC
- Core services:
  - Cloud Asset Inventory
  - Detecting security misconfiguration using Security Health Analytics
  - VM Manager
- Rapid vulnerability detection
- Threat detection
- Continuous compliance monitoring using SCC
- Exporting SCC findings
- Automating findings response

Overview of SCC

Google introduced SCC in mid-2018. SCC provides many features to monitor your organization's security controls, detect threats, and use alerts to automate security responses.

Here are some of the key features of SCC:

- **Gain centralized visibility and control over your Google Cloud data and resources**:
  - SCC gives enterprises centralized visibility of their cloud assets across App Engine, BigQuery, Cloud SQL, Cloud Storage, Compute Engine, **Identity and Access Management** (**IAM**) policies, **Google Kubernetes Engine** (**GKE**), and more.
  - SCC enables enterprises to quickly find out the number of projects in their environment, what resources are deployed, where sensitive data is located, and which service accounts have been added or removed. You can leverage the SCC REST API to access assets and findings and make it easier to integrate with existing systems.
  - You can view your Google Cloud asset history to understand exactly what changed in your environment and respond to the most pressing issues first.
  - You can receive notifications about new findings or updates to existing findings. The notifications are typically delivered within six to twelve minutes, ensuring that you stay informed and can take appropriate actions promptly.
- **Find and fix risky misconfigurations**:
  - **Security Health Analytics** (**SHA**) is a built-in service in SCC that helps you identify security misconfigurations in your Google Cloud assets and resolve them by following actionable recommendations.
  - **Web Security Scanner** (**WSS**) is a built-in service in SCC that can automatically detect web applications running in Google Cloud and scan them for web app vulnerabilities. This can help you catch web application vulnerabilities before they hit production and reduce your exposure to risks.
- **Report on and maintain compliance**:
  - SCC Premium helps you identify compliance violations in your Google Cloud assets and resolve them by following actionable recommendations.
  - You can review and export compliance reports to help ensure all your resources are meeting their compliance requirements.
- **Detect threats targeting your Google Cloud assets**:
  - Event Threat Detection is a built-in service in SCC Premium that helps customers detect threats using logs running in Google Cloud at scale:
    - Uncover suspicious cloud-based activity using Google's unique threat intelligence

- Use kernel-level instrumentation to identify potential compromises of containers, including suspicious binaries
- Combine threat intelligence from first- and third-party providers, such as Palo Alto Networks, to better protect your enterprise from costly and high-risk threats
- Detect whether your **virtual machines** (**VMs**) are being used for abuse cases, such as coin mining
- Respond to threats by using a Cloud Pub/Sub event or Cloud Functions and reduce your time to resolution

- **Container Threat Detection** (**KTD**) is a built-in service in SCC that detects the most common container runtime attacks and alerts you to any suspicious activity

Now that you have a brief idea of what SCC can do, let us dive deep into the core services of SCC.

Core services

Core services contribute to various parts of the security architecture of your Google Cloud organization for detection and alerting. The following diagram shows the core services offered by SCC.

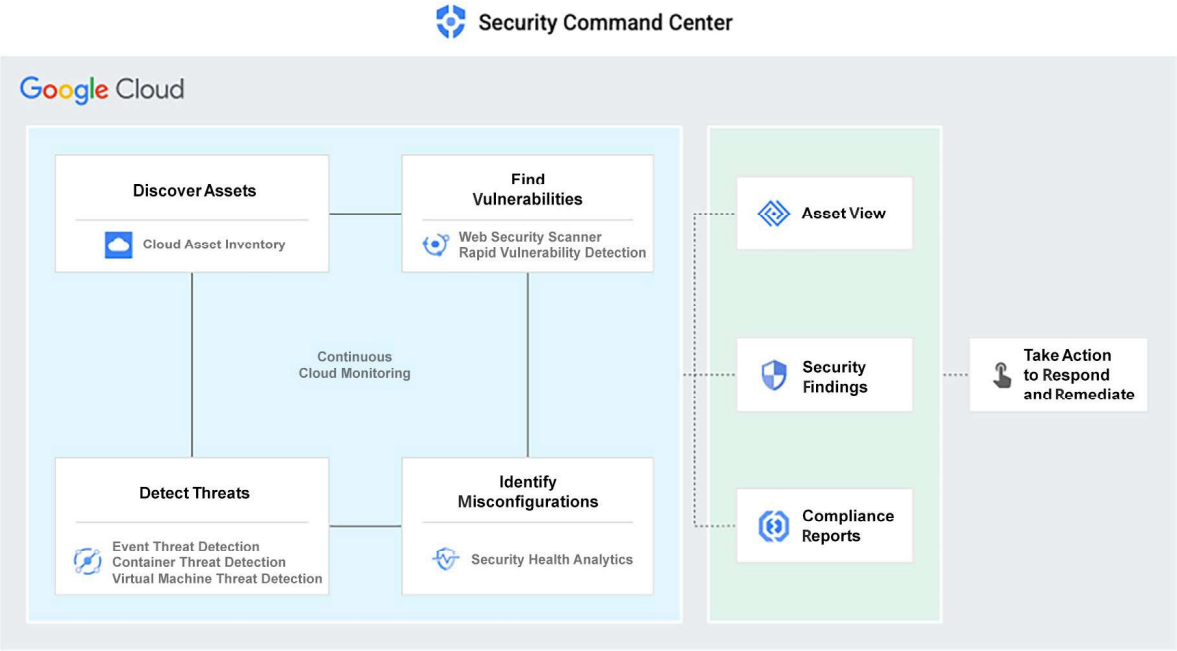

Figure 14.1 – SCC core services

As shown in *Figure 14.1*, SCC is a collection of various modules providing detection and alerting capability:

- **Cloud Asset Inventory** (**CAI**): CAI provides full visibility of all assets in your Google Cloud organization. You can search by type of resources, projects, and locations. CAI also provides all IAM policies in your Google Cloud organization.
- **Vulnerability Findings**: You can detect misconfigurations and vulnerabilities in your Google Cloud organization using SHA, VM Manager, WSS, and Rapid Vulnerability Detection.
- **Event Threat Detection** (**ETD**): ETD is a collection of threat detection capabilities that provides alerts on threats to your Google Cloud organization. It will also alert threats to containers and VMs.

Now let us go over each of these capabilities and understand the details of how SCC generates findings.

Cloud Asset Inventory

CAI provides a list of asset metadata for your cloud resources based on when they were created and/or updated. Note that the roles in SCC are assigned at various levels of your Google Cloud resource hierarchy. The amount of access you have determines your ability to see, edit, create, or change findings, assets, and security sources. Typically, security operations have access to see findings at the organization level, while project teams have access to see findings at the individual project level.

As a member of the security team, you need to know how to query assets to be able to quickly find what has changed in your cloud environment. For example, it would be highly suspicious behavior if you found accelerator-optimized machines (GPUs and HPC) being unexpectedly provisioned.

Let us go over various features of CAI so you can understand how it works and how to use it.

Listing assets

Assets are Google Cloud resources within your Google Cloud organization, such as Compute Engine instances, projects, BigQuery datasets, or Cloud Storage buckets.

The following command can be used to list assets for a given Google Cloud organization:

```
ORGANIZATION_ID=12344321
gcloud scc assets list $ORGANIZATION_ID
```

The output will look like this:

```
asset:
  createTime: '2021-10-05T17:55:14.823Z'
  iamPolicy:
    policyBlob: '{"bindings":[{"role":"roles/
owner","members":["serviceAccount:SERVICE_ACCOUNT@PROJECT_ID.iam.
```

```
gserviceaccount.com","user:USER_EMAIL@gmail.com"]}]}'
  name: organizations/ORGANIZATION_ID/assets/ASSET_ID
  resourceProperties:
    createTime: '2021-10-05T17:36:17.915Z'
    lifecycleState: ACTIVE
    name: PROJECT_ID
    parent: '{"id":"ORGANIZATION_ID","type":"organization"}'
    projectId: PROJECT_ID
    projectNumber: 'PROJECT_NUMBER'
  securityCenterProperties:
    resourceDisplayName: PROJECT_ID
    resourceName: //cloudresourcemanager.googleapis.com/projects/
PROJECT_NUMBER
    resourceOwners:
    - serviceAccount:SERVICE_ACCOUNT@PROJECT_ID.iam.gserviceaccount.
com
    - user:USER_EMAIL@gmail.com
    resourceParent: //cloudresourcemanager.googleapis.com/
organizations/ORGANIZATION_ID
    resourceParentDisplayName: ORGANIZATION_NAME
    resourceProject: //cloudresourcemanager.googleapis.com/projects/
PROJECT_NUMBER
    resourceProjectDisplayName: PROJECT_ID
    resourceType: google.cloud.resourcemanager.Project
  securityMarks:
    name: organizations/ORGANIZATION_ID/assets/ASSET_ID/securityMarks
  updateTime: '2021-10-05T17:55:14.823Z'
```

As you can see, you are going to get a very large list of assets, so it is imperative to understand how to filter them based on your requirements. Let us look at that now.

Filtering assets

An organization may have many assets. Because there are no filters in the preceding example, all assets are returned. You can utilize asset filters in SCC to get information about certain assets. Filters, like `where` clauses in SQL queries, only apply to objects returned by the API instead of columns.

The previous example's sample output displays certain fields and subfields, as well as their characteristics, that can be utilized in asset filters. The SCC API accepts complete JSON arrays and objects as potential property types. You can refine your results by using the following filters:

- Array elements
- Full JSON objects with partial string matches within the object
- JSON object subfields

Filter expressions must use the following comparison operators and sub-fields must be numbers, strings, or Booleans:

- Strings:
  - Full equality, =
  - Partial string matching, :
- Numbers:
  - Inequalities, <, >, <=, >=
  - Equality, =
- Booleans:
  - Equality, =

Now let us see how to create a filter and use it in a command to filter assets:

```
ORGANIZATION_ID=12344321
FILTER="security_center_properties.resource_type=\"google.cloud.
resourcemanager.Project\""
 gcloud scc assets list $ORGANIZATION_ID -filter="$FILTER"
```

Let us see some examples of how to construct a filter.

To find a project with a specific owner, you can form a filter like this:

```
"security_center_properties.resource_type = \"google.cloud.
resourcemanager.Project\" AND security_center_properties.resource_
owners : \"$USER\""
```

In this example, `$USER` is typically in the format `user:someone@domain.com`. This compares the `$USER` using the substring operator `:`, so an exact match is not necessary.

To find firewall rules with open HTTP ports, you can create a filter like this:

```
"security_center_properties.resource_type = \"google.compute.
Firewall\" AND resource_properties.name =\"default-allow-http\""
```

Now let us see a few examples of filters based on time conditions. To find project assets created at or before a specific time, create a filter as follows.

Let us say our requirement is for querying assets that were generated on or before `July 18, 2019` at `8:26:21` PM GMT.

You can express time with the `create_time` filter in the following forms and types:

```
Unix time (in milliseconds) as an integer literal
"create_time <= 1563481581000"
RFC 3339 as a string literal
"create_time <= \"2019-07-18T20:26:21+00:00\""
```

Listing assets at a point in time

The previous examples demonstrate how to create a list of an organization's current assets. You can also view a historical snapshot of an organization's assets using SCC.

The following sample retrieves the current state of all assets at a given point in time. Millisecond time resolutions are supported by SCC:

```
# ORGANIZATION_ID=12344321
# READ_TIME follows the format YYYY-MM-DDThh:mm:ss.ffffffZ
READ_TIME=2019-02-28T07:00:06.861Z
gcloud scc assets list $ORGANIZATION_ID -read time=$READ_TIME
```

Listing assets that changed state

A prime requirement is to be able to find assets that changed state after a certain time. You can compare an asset at two points in time to see whether it was added, removed, or present at the specified time using SCC.

The following example compares projects that exist at `READ_TIME` to a previous point in time specified by `COMPARE_DURATION`. Note that `COMPARE_DURATION` is provided in seconds.

When `COMPARE_DURATION` is set, the `stateChange` attribute in the result of the list asset query is updated with one of the following values:

- `ADDED`: The asset was not present at the start of `compareDuration`, but was present at `readTime`
- `REMOVED`: The asset was present at the start of `compareDuration`, but was not present at `readTime`
- `ACTIVE`: The asset was present at both the start and the end time defined by `compareDuration` and `readTime`

Now let us see how to form the filter for this condition:

```
# ORGANIZATION_ID=12344321
# READ_TIME follows the format YYYY-MM-DDThh:mm:ss.ffffffZ
READ_TIME=2019-02-28T07:00:06.861Z
 FILTER="security_center_properties.resource_type=\"google.cloud.
resourcemanager.Project\""
COMPARE_DURATION=86400s
```

Now execute the following command using the preceding filter:

```
gcloud scc assets list $ORGANIZATION_ID --read-time=$READ_TIME
--filter="$FILTER" \
--compare-duration=$COMPARE_DURATION
```

We will take a bit of a detour and look at an important aspect of asset analysis: BigQuery exports. You can export assets to BigQuery and run queries using SQL. Let us see how to do that.

Exporting assets to BigQuery

In this section, we will quickly see how to export assets to BigQuery and then run data analysis on your asset inventory. BigQuery provides a SQL-like experience for users to analyze data and produce meaningful insights without the use of custom scripts.

Before you execute the queries, follow this URL to set up the BigQuery export of assets: `https://packt.link/wvqQq`.

Let us look at some of the most run queries for assets. Replace `PROJECT_ID`, `DATASET_ID`, and `TABLE_NAME` with the appropriate values for your project:

1. Find the quantity of each asset type by using the following query:

```
SELECT asset_type, COUNT(*) AS asset_count
FROM `PROJECT_ID.DATASET_ID.TABLE_NAME`
GROUP BY asset_type
ORDER BY asset_count DESC
```

2. To find an organization, folder, or project that allows creating using a public IP, run the following query. This query is useful because allowing public IPs with Cloud SQL instances can introduce vulnerabilities unless SSL or a proxy is configured:

```
SELECT name
FROM `PROJECT_ID.DATASET_ID.TABLE_NAME`
JOIN UNNEST(org_policy) AS op
WHERE
  op.constraint = "constraints/sql.restrictPublicIp"
  AND (op.boolean_policy IS NULL OR op.boolean_policy.enforced =
FALSE);
```

The preceding two queries are for general cloud assets; however, one of the important aspects of asset inventory is to find out IAM policies that potentially violate your organization's security policies. Let us look at these now. Replace `PROJECT_ID`, `DATASET_ID`, and `TABLE_NAME` with the appropriate values for your project.

3. To find IAM policies that grant access to Gmail accounts, run the following query. This query is useful to see whether there has been any exfiltration from your Google Cloud organization to consumer accounts:

```
SELECT name, asset_type, bindings.role
FROM `PROJECT_ID.DATASET_ID.TABLE_NAME`
JOIN UNNEST(iam_policy.bindings) AS bindings
JOIN UNNEST(bindings.members) AS principals
WHERE principals like %@gmail.com
```

4. To find IAM policies that grant access to direct users rather than groups (as you know this is an anti-pattern for IAM access control), run the following query:

```
SELECT name, asset_type, bindings.role, members
FROM `asset-inventory-api.iam_asset_inventory.iam_policy_
inventory`
JOIN UNNEST(iam_policy.bindings) AS bindings
JOIN UNNEST(bindings.members) AS members
WHERE members like "%@acme.com"order by name
```

5. To find IAM policies containing all service accounts, run the following query. This query is important to find out whether any service accounts are being granted privileged roles:

```
SELECT name, asset_type, bindings.role, members
FROM `asset-inventory-api.iam_asset_inventory.iam_policy_
inventory`
JOIN UNNEST(iam_policy.bindings) AS bindings
JOIN UNNEST(bindings.members) AS members
WHERE members like "serviceAccount%"
order by name
```

6. To find IAM policies that are granted using groups, execute a query like this:

```
SELECT name, asset_type, bindings.role, members
FROM `asset-inventory-api.iam_asset_inventory.iam_policy_
inventory`
JOIN UNNEST(iam_policy.bindings) AS bindings
JOIN UNNEST(bindings.members) AS members
WHERE members like "group%"
order by name
```

Now that you have a solid understanding of how to query assets either by using SCC commands or BigQuery, we will move on to another module of SCC for detecting security misconfigurations.

Detecting security misconfigurations and vulnerabilities

Most of the cloud threats that you will find will be due to security misconfigurations or a lack of understanding of how the cloud works. So, it is critical to understand how to find misconfigurations and how to quickly analyze and fix them. SCC reports findings from four categories of detectors:

- Security Health Analytics
- Rapid Vulnerability Detection
- Web Security Scanner
- VM Manager vulnerabilities

Now let us look at each of them to understand the details.

Security Health Analytics

SHA is a service within SCC that has built-in detectors to identify misconfigurations. SHA automatically scans your Google Cloud organization for known vulnerable configurations against compliance benchmarks such as CIS, PCI DSS, NIS 800-53, ISO 27001, and the OWASP Top 10. SHA scans begin around an hour after SCC is turned on and can be done in one of two modes: batch mode, which conducts scans twice a day, 12 hours apart; and real-time mode, which executes scans in response to asset configuration changes.

SHA scans are available in three different modes:

- **Batch scan**: All detectors are set to run two or more times per day for all enrolled organizations. Detectors run on various schedules to satisfy **service-level objectives** (**SLOs**). Detectors execute batch scans every 6 or 12 hours to meet 12- and 24-hour SLOs, respectively. Interim batch scan resources and policy changes are not immediately collected and applied in the next batch scan.

> **Note**
> Batch scan timetables are performance goals rather than service assurances.

- **Real-time scan**: When CAI reports a change in an asset's configuration, supported detectors begin scanning. The findings are sent to SCC right away.
- **Mixed mode**: Some detectors that offer real-time scans may not be able to detect changes in all supported assets in real time. Configuration updates for some assets are caught immediately, while batch scans capture changes for others. In the tables on this page, exceptions are highlighted.

> **Note**
> There are some detectors that do not support real-time scanning mode. For detector latency information, check out the Google Cloud documentation at `https://packt.link/TeQlR`.

SCC Premium must use most SHA detectors. The Standard tier provides the following finding types:

- `DATAPROC_IMAGE_OUTDATED`
- `LEGACY_AUTHORIZATION_ENABLED`
- `MFA_NOT_ENFORCED`
- `NON_ORG_IAM_MEMBER`
- `OPEN_CISCOSECURE_WEBSM_PORT`
- `OPEN_DIRECTORY_SERVICES_PORT`
- `OPEN_FIREWALL`
- `OPEN_GROUP_IAM_MEMBER`
- `OPEN_RDP_PORT`
- `OPEN_SSH_PORT`
- `OPEN_TELNET_PORT`
- `PUBLIC_BUCKET_ACL`
- `PUBLIC_COMPUTE_IMAGE`
- `PUBLIC_DATASET`
- `PUBLIC_IP_ADDRESS`
- `PUBLIC_LOG_BUCKET`
- `PUBLIC_SQL_INSTANCE`
- `SSL_NOT_ENFORCED`
- `WEB_UI_ENABLED`

The following SHA detectors are not enabled by default:

- `BIGQUERY_TABLE_CMEK_DISABLED`
- `BUCKET_CMEK_DISABLED`
- `DATASET_CMEK_DISABLED`

- DISK_CMEK_DISABLED
- DISK_CSEK_DISABLED
- NODEPOOL_BOOT_CMEK_DISABLED
- PUBSUB_CMEK_DISABLED
- SQL_CMEK_DISABLED
- SQL_NO_ROOT_PASSWORD
- SQL_WEAK_ROOT_PASSWORD

To turn on a detector, also known as a module, run the following commands in the Google Cloud CLI:

```
gcloud alpha scc settings services modules enable \
  --organization=ORGANIZATION_ID \
  --service=SECURITY_HEALTH_ANALYTICS \
  --module=DETECTOR_NAME
```

In addition to SHA findings, SCC also reports on VM Manager and Rapid Vulnerability Detection. Let us go over those now.

VM Manager

VM Manager is a suite of tools that can be used to manage the OS for VMs running Windows and Linux on Compute Engine. This is a paid service and you will need to enable it. The findings are reported to SCC only for the SCC Premium version. Findings simplify the process of using VM Manager's patch compliance feature, which is in preview at the time of writing this book. The feature lets you conduct patch management at the organization level across all of your projects. Note that VM Manager supports patch management at the single project level.

VM Manager reports findings in SCC. Here are some features of the findings that you should be aware of:

- VM Manager's vulnerability reports detail vulnerabilities in installed OS packages for Compute Engine VMs, including **common vulnerabilities and exposures** (**CVEs**).
- Assets excluded from scans: **SUSE Linux Enterprise Server** (**SLES**), Windows OSs.
- Findings appear in SCC shortly after vulnerabilities are detected.
- Vulnerability reports in VM Manager are generated as follows:
  - For most vulnerabilities in the installed OS package, the OS Config API generates a vulnerability report within a few minutes of the change
  - For CVEs, the OS Config API generates the vulnerability report within three to four hours of the CVE being published to the OS

VM Manager also provides a nice graphical interface for visualization of the findings. Let us look at Rapid Vulnerability Detection.

Rapid Vulnerability Detection

Rapid Vulnerability Detection detects weak credentials, incomplete software installations, and other critical vulnerabilities that have a high likelihood of being exploited. The service automatically discovers network endpoints, protocols, open ports, network services, and installed software packages.

Rapid Vulnerability Detection findings are early warnings of vulnerabilities that Google recommends you fix immediately.

Here is a list of findings generated by Rapid Vulnerability Detection:

> **Note**
> Please refer to the Google Cloud documentation to find out the details of the findings and the remediation.

Weak credential findings:

- `WEAK_CREDENTIALS`

Exposed Interface findings:

- `ELASTICSEARCH_API_EXPOSED`
- `EXPOSED_GRAFANA_ENDPOINT`
- `EXPOSED_METABASE`
- `EXPOSED_SPRING_BOOT_ACTUATOR_ENDPOINT`
- `HADOOP_YARN_UNAUTHENTICATED_RESOURCE_MANAGER_API`
- `JAVA_JMX_RMI_EXPOSED`
- `JUPYTER_NOTEBOOK_EXPOSED_UI`
- `KUBERNETES_API_EXPOSED`
- `UNFINISHED_WORDPRESS_INSTALLATION`
- `UNAUTHENTICATED_JENKINS_NEW_ITEM_CONSOLE`

Software findings for remote code execution vulnerabilities:

- APACHE_HTTPD_RCE
- APACHE_HTTPD_SSRF
- CONSUL_RCE
- DRUID_RCE
- DRUPAL_RCE
- FLINK_FILE_DISCLOSURE
- GITLAB_RCE
- GoCD_RCE
- JENKINS_RCE
- JOOMLA_RCE
- LOG4J_RCE
- MANTISBT_PRIVILEGE_ESCALATION
- OGNL_RCE
- OPENAM_RCE
- ORACLE_WEBLOGIC_RCE
- PHPUNIT_RCE
- PHP_CGI_RCE
- PORTAL_RCE
- REDIS_RCE
- SOLR_FILE_EXPOSED
- SOLR_RCE
- STRUTS_RCE
- TOMCAT_FILE_DISCLOSURE
- VBULLETIN_RCE
- VCENTER_RCE
- WEBLOGIC_RCE

Now that we have looked at Rapid Vulnerability Detection, let us move on to web security vulnerabilities.

Web Security Scanner

WSS scans your App Engine, GKE, and Compute Engine web apps for security flaws. It crawls your apps, following all links that fall within the scope of your application's entry URLs, and tries to test as many user inputs and event handlers as possible. WSS only works with public URLs and IP addresses that are not behind a firewall.

App Engine standard and flexible environments, Compute Engine instances, and GKE resources are presently supported by WSS.

WSS works with your secure design and development procedures. It uses a combination of automated and manual techniques to scan web applications and services for common vulnerabilities such as SQL injection, cross-site scripting, and other security issues. It works by sending a series of requests to the target application or service and then analyzing the responses for possible vulnerabilities. WSS can also be used to conduct manual security reviews by providing detailed reports on potential issues. Additionally, WSS also provides remediation advice and steps to address any identified vulnerabilities.

The **OWASP Top Ten**, a publication that ranks and provides remedial assistance for the top 10 most critical online application security vulnerabilities, is supported by WSS.

There are two types of scans WSS supports:

- **Managed scans**: Managed scans detect and scan public web endpoints once a week automatically:
  - These scans do not employ authentication and perform GET-only queries, so no forms are submitted on live websites.
  - Managed scans are done independently of custom scans defined at the project level. Managed scan findings are automatically available on the SCC **Vulnerabilities** tab and related reports when you enable WSS as a service.

  A managed scan is especially useful without involving individual project teams, as you can utilize managed scans to centrally manage basic web application vulnerability detection for projects in your organization. You can work with those teams later to set up more comprehensive bespoke scans after the discoveries are made.
- **Custom scans**: WSS custom scans provide detailed information on application vulnerabilities such as obsolete libraries, cross-site scripting, and mixed content usage.

Here is a list of all the findings generated by WSS:

> **Note**
> For a description of these findings and remediation, look up the relevant Google Cloud documentation.

- `ACCESSIBLE_GIT_REPOSITORY`
- `ACCESSIBLE_SVN_REPOSITORY`
- `CACHEABLE_PASSWORD_INPUT`
- `CLEAR_TEXT_PASSWORD`
- `INSECURE_ALLOW_ORIGIN_ENDS_WITH_VALIDATION`
- `INSECURE_ALLOW_ORIGIN_STARTS_WITH_VALIDATION`
- `INVALID_CONTENT_TYPE`
- `INVALID_HEADER`
- `MISMATCHING_SECURITY_HEADER_VALUES`
- `MISSPELLED_SECURITY_HEADER_NAME`
- `MIXED_CONTENT`
- `OUTDATED_LIBRARY`
- `SERVER_SIDE_REQUEST_FORGERY`
- `SESSION_ID_LEAK`
- `SQL_INJECTION`
- `STRUTS_INSECURE_DESERIALIZATION`
- `XSS`
- `XSS_ANGULAR_CALLBACK`
- `XSS_ERROR`
- `XXE_REFLECTED_FILE_LEAKAGE`

While WSS is easy enough to use, there are some best practices that you should follow while using it. We will go over those now.