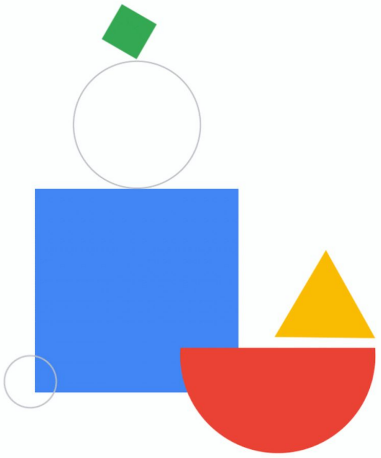
Developing Applications with Google Cloud: Foundations

Module 6: Deploying Applications



Welcome to Developing Applications with Google Cloud: Foundations, module 6: Deploying Applications.

To run reliable services, you must have reliable release processes. When you have many engineers building separate application components, it's crucial that they are able to run unit tests, integration tests, and other tests quickly.

With most user-facing software, teams want to release software often with new features and bug fixes. To enable high release velocity, build, test, and release processes must be automated as much as possible.

Agenda

01	Continuous integration and delivery
02	Containers and building application images

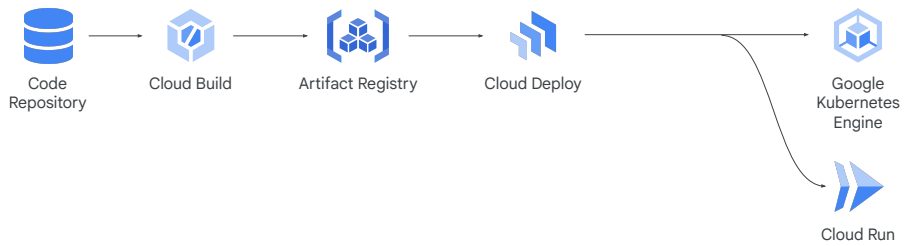
In this module, you learn the components of a continuous integration and delivery pipeline. You also learn how to build container images for your application by using Cloud Build, and how to push your images to Artifact Registry.

Agenda

01	Continuous integration and delivery
02	Containers and building application images

A continuous integration and continuous delivery pipeline provides a stable, repeatable process for building and deploying your applications.

Continuous integration and delivery (CI/CD) pipeline



Continuous integration occurs when developers commit their changes into a feature branch in a code repository, and a build service like Cloud Build is automatically triggered. Rules that you have established guide the generation of your application containers and executables. The application's artifacts are then stored in a repository like Artifact Registry.

Continuous delivery is triggered when changes are pushed to the main branch in the repository. The build system builds the code and creates application images. The deployment system, like Cloud Deploy, then deploys application images to Cloud Run or GKE in the staging environment to automatically run integration and performance tests. If all tests pass, the build is tagged as a release candidate build. You can manually approve a release candidate build. This approval can trigger deployment to production environments as a canary or blue-green release.

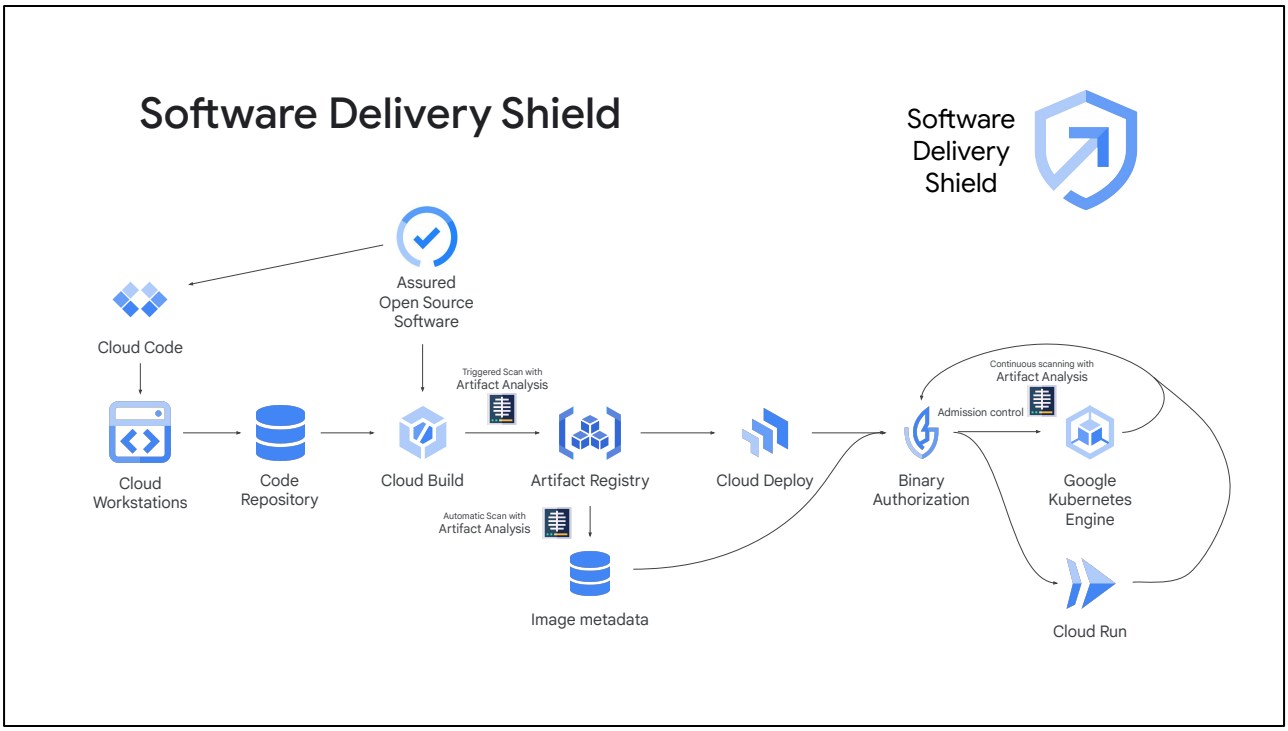
You can monitor the performance of your application in the production environment by using Cloud Monitoring. If the new deployment functions optimally, you can switch over all of your traffic to this new release. But if you discover problems, you can also quickly roll back to the last stable release.

The continuous deployment workflow varies slightly in that there is no manual approval process. The deployment system automatically deploys release candidates to the production environment.

When you first get started, this diagram shows a simple CI/CD pipeline that you might

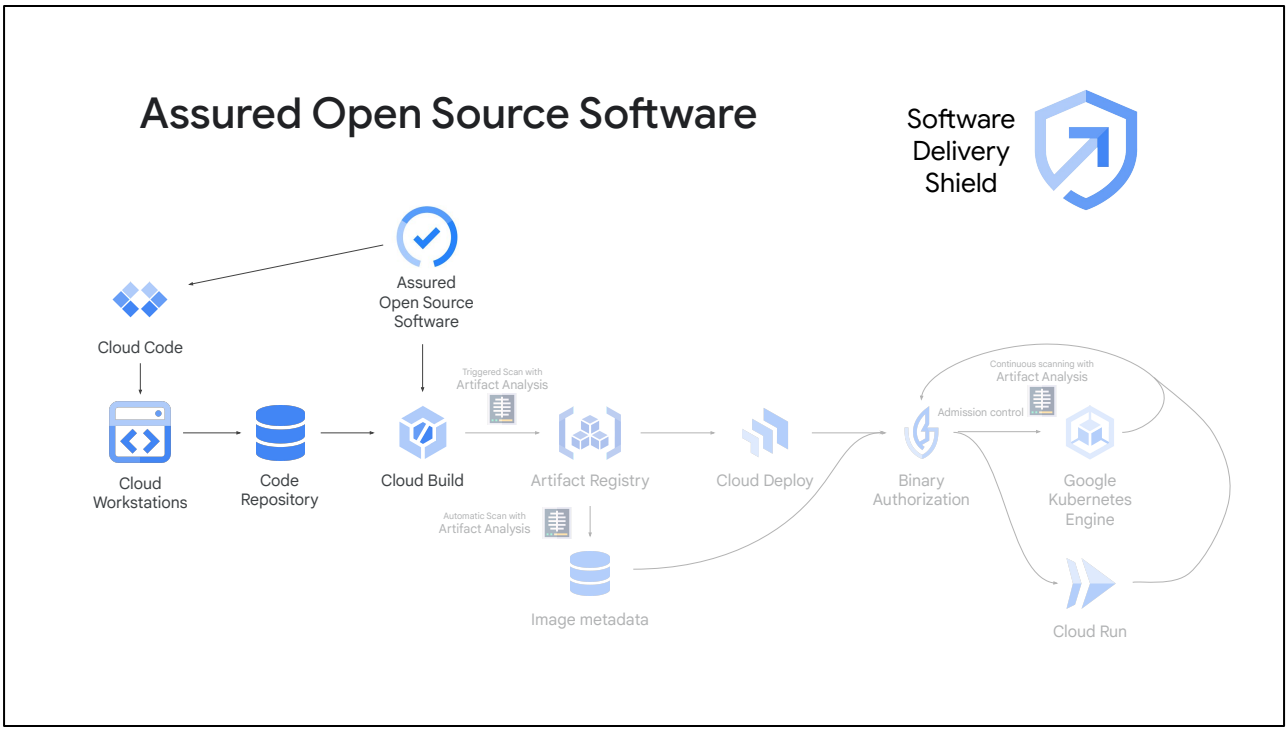
build. This type of pipeline provides an efficient and consistent build and deploy process. However, it's very important to consider security throughout the continuous integration and delivery process.

Google Cloud provides several services that will help secure your builds and deployments.



Google Cloud's Software Delivery Shield is a fully managed, end-to-end software supply chain security solution that protects every step of your CI/CD process.

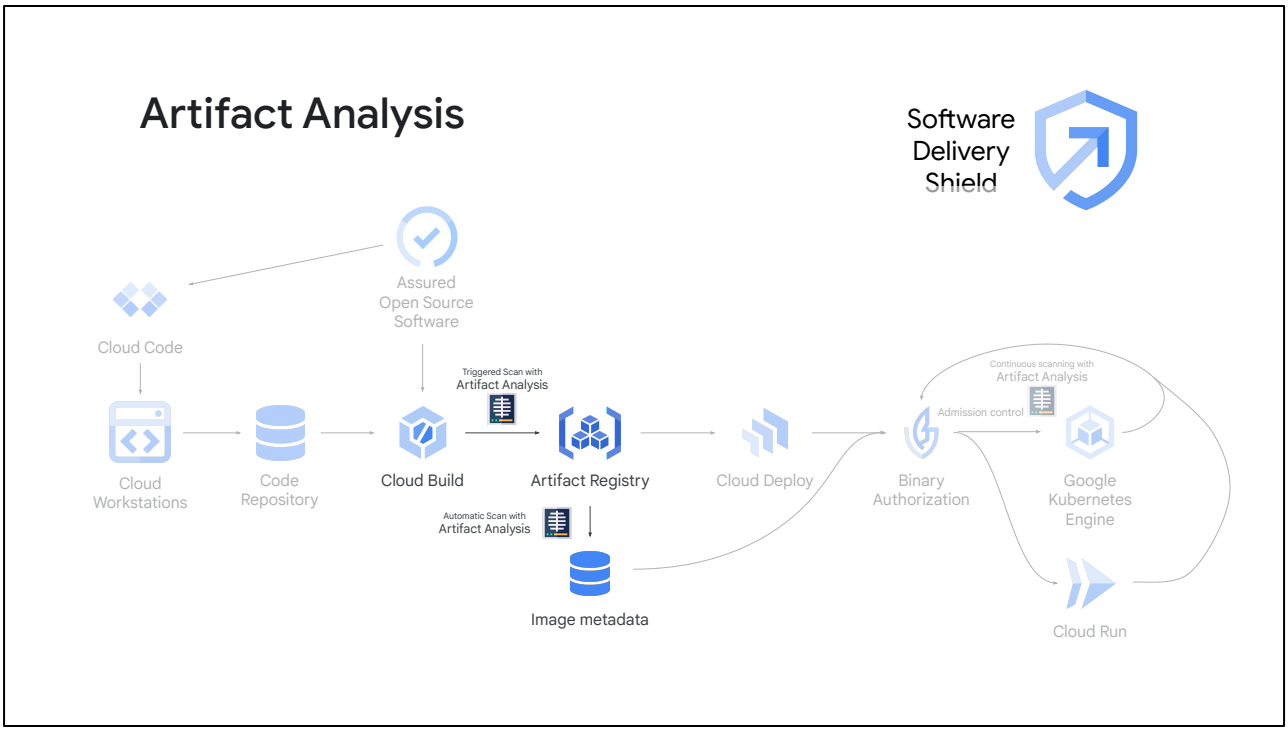
[Software Delivery Shield overview: https://cloud.google.com/software-supply-chain-security/docs/sds/overview]



The Assured Open Source Software service lets you incorporate open Java and Python source packages that have been verified and tested by Google. These packages are built using Google's secure pipelines, and the packages are regularly scanned, analyzed, and tested for vulnerabilities. Google actively finds and fixes vulnerabilities in these packages.

Cloud Build can import your source code and verified open source packages and execute your build on Google Cloud infrastructure. Cloud Build maintains verifiable metadata about the build, helping you to verify that a built artifact was created from trusted sources by a trusted build system.

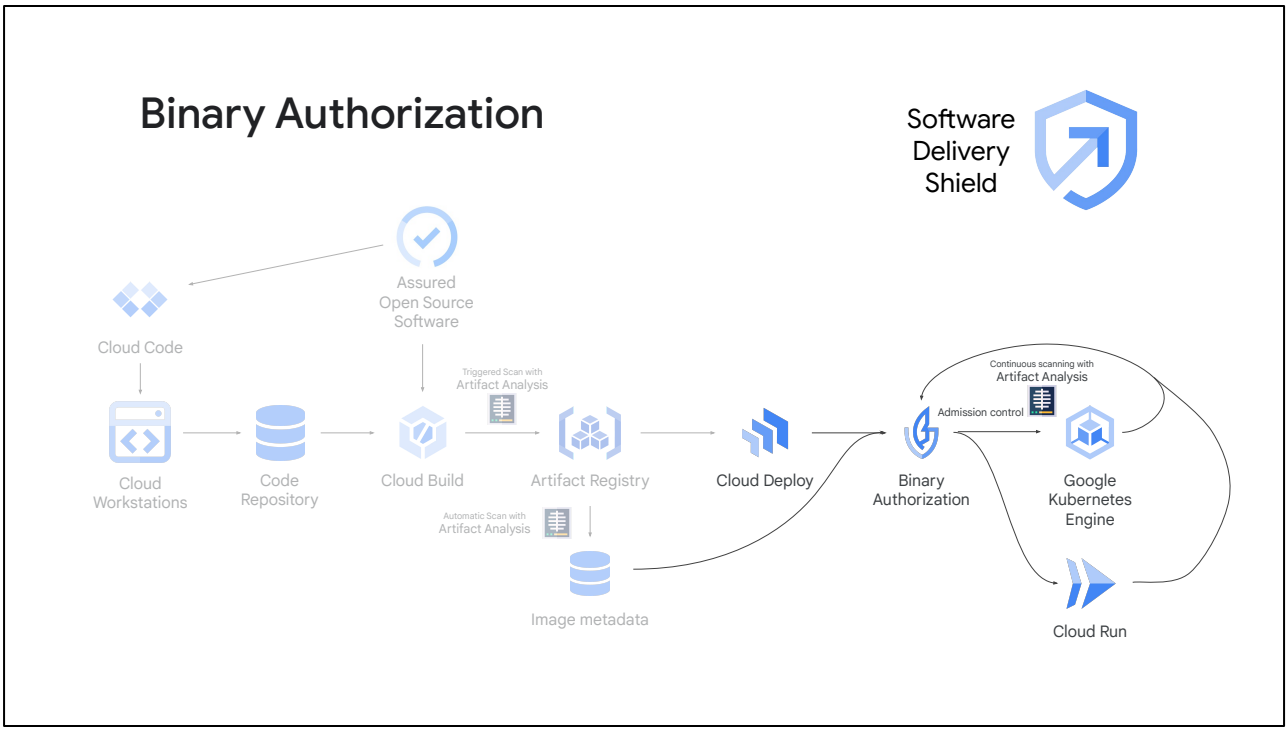
[Overview of Assured Open Source Software: https://cloud.google.com/assured-open-source-software/docs/overview
Overview of Cloud Build: https://cloud.google.com/build/docs/overview]



Artifact Registry lets you store, secure, and manage your build artifacts, and Artifact Analysis proactively detects vulnerabilities for artifacts in Artifact Registry.

Artifact Analysis provides integrated on-demand and automated scanning for base container images and Maven and Go packages in containers. When you push a Java project to Artifact Registry, Artifact Analysis scans for vulnerabilities within the open source dependencies used by your Maven artifacts. After the initial scan, Artifact Analysis continuously monitors the metadata for scanned images in Artifact Registry for new vulnerabilities.

[Artifact Registry overview: https://cloud.google.com/artifact-registry/docs/overview
Artifact Analysis: https://cloud.google.com/artifact-analysis/docs/artifact-analysis]



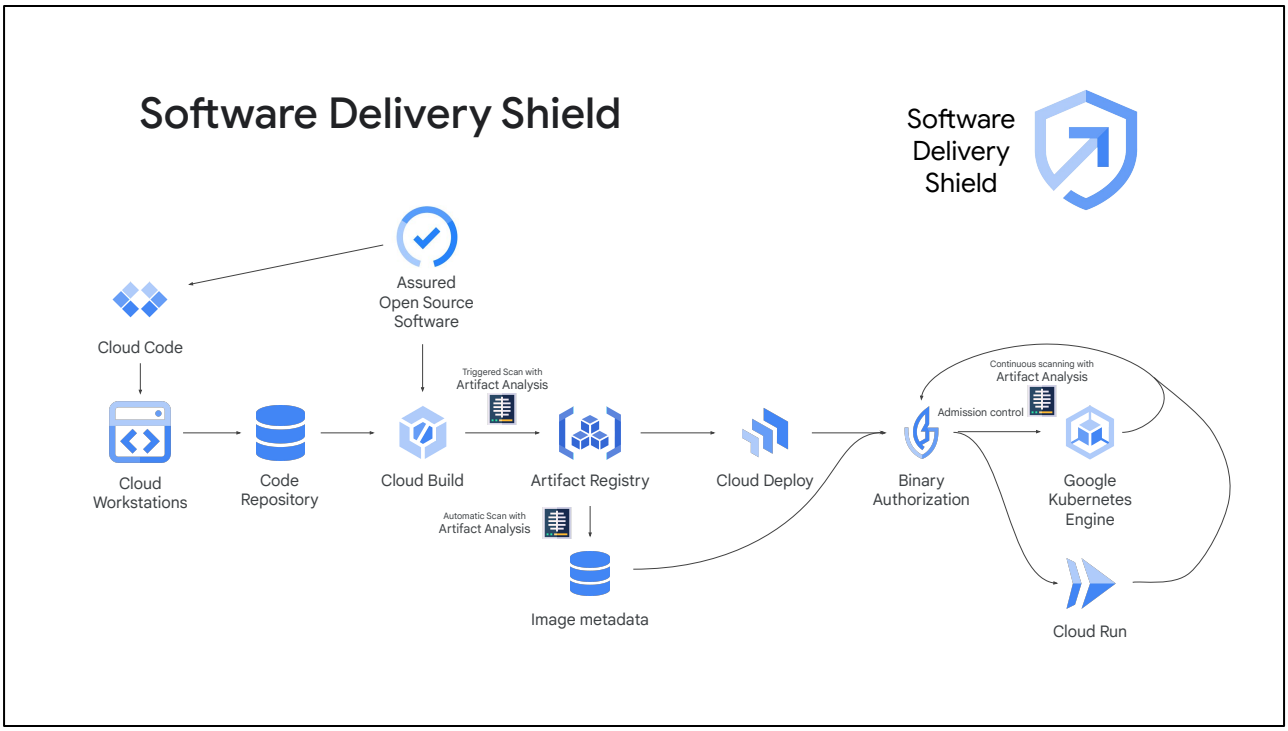
Cloud Deploy automates delivery of your applications to a series of target environments in a defined sequence. It supports continuous delivery directly to Cloud Run and GKE, with one-click approvals and rollbacks. It also displays security insights for deployed applications.

Binary Authorization helps establish, maintain, and verify a chain of trust along your software supply chain. Binary Authorization collects attestations, which are digital documents that certify that an image was built by successfully executing a specific, required process. Binary Authorization ensures that an image is deployed only when the attestations meet your organization's policy, and it alerts you when policy violations are found.

GKE and Cloud Run also contribute to your pipeline security. GKE can assess your container security and give active guidance around cluster settings, workload configuration, and vulnerabilities. It can evaluate your GKE clusters and workloads and provide you with actionable recommendations to improve security. Cloud Run also contains a security panel that displays security insights about your builds and vulnerabilities in running services.

[Overview of Cloud Deploy: https://cloud.google.com/deploy/docs/overview
Binary Authorization overview:
https://cloud.google.com/binary-authorization/docs/overview
About the security posture dashboard (GKE):
https://cloud.google.com/kubernetes-engine/docs/concepts/about-security-posture-da

shboard
Deploy on Cloud Run and view security insights:
https://cloud.google.com/software-supply-chain-security/docs/sds/deploy-run-view-security-insights]



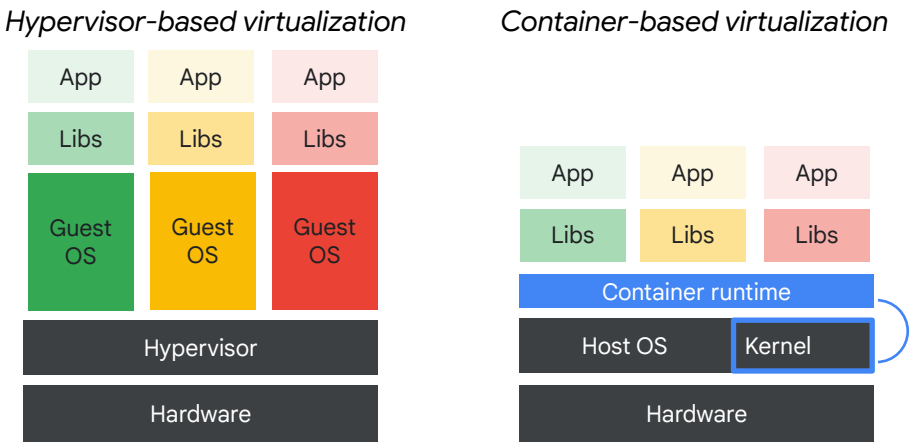
Software Delivery Shield helps you fully secure your CI/CD pipeline.

Agenda

01	Continuous integration and delivery
02	Containers and building application images

Now we discuss containerized applications and how they can be built on Google Cloud.

Containers: an efficient way to isolate code and manage workloads



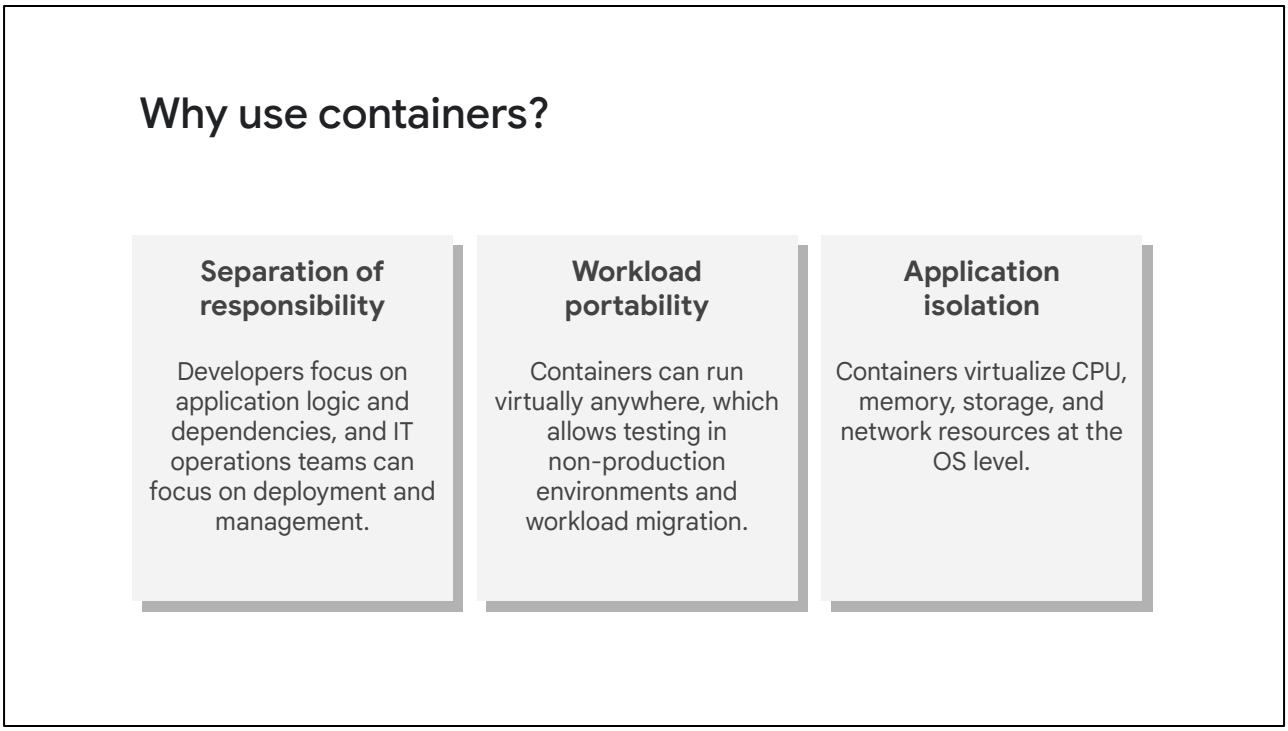
Containers are a preferred method for packaging and deploying your applications. Here's a quick refresher on Containers.

Container-based virtualization is an alternative to hardware virtualization, as used in traditional virtual machines. Virtual machines are isolated from one another in part by having each virtual machine have its own instance of the operating system. But operating systems can be slow to boot and can be resource-heavy. Containers respond to these problems by using built-in capabilities of modern operating systems to isolate container environments from one another.

A process is a running program. In Linux and Windows, the memory address spaces of running processes have long been isolated from one another. Popular implementations of software containers build on this isolation. They take advantage of additional operating-system features that give processes the capability to have their own namespaces and limit other processes' access to resources.

Containers start much faster than virtual machines and use fewer resources, because each container does not have its own instance of the operating system. Instead, developers configure each container with a minimal set of software libraries to do the job. A lightweight container runtime does the plumbing jobs needed to allow that container to launch and run, calling into the kernel as necessary. The container runtime also determines the image format.

Containers can be deployed to both Cloud Run and GKE.

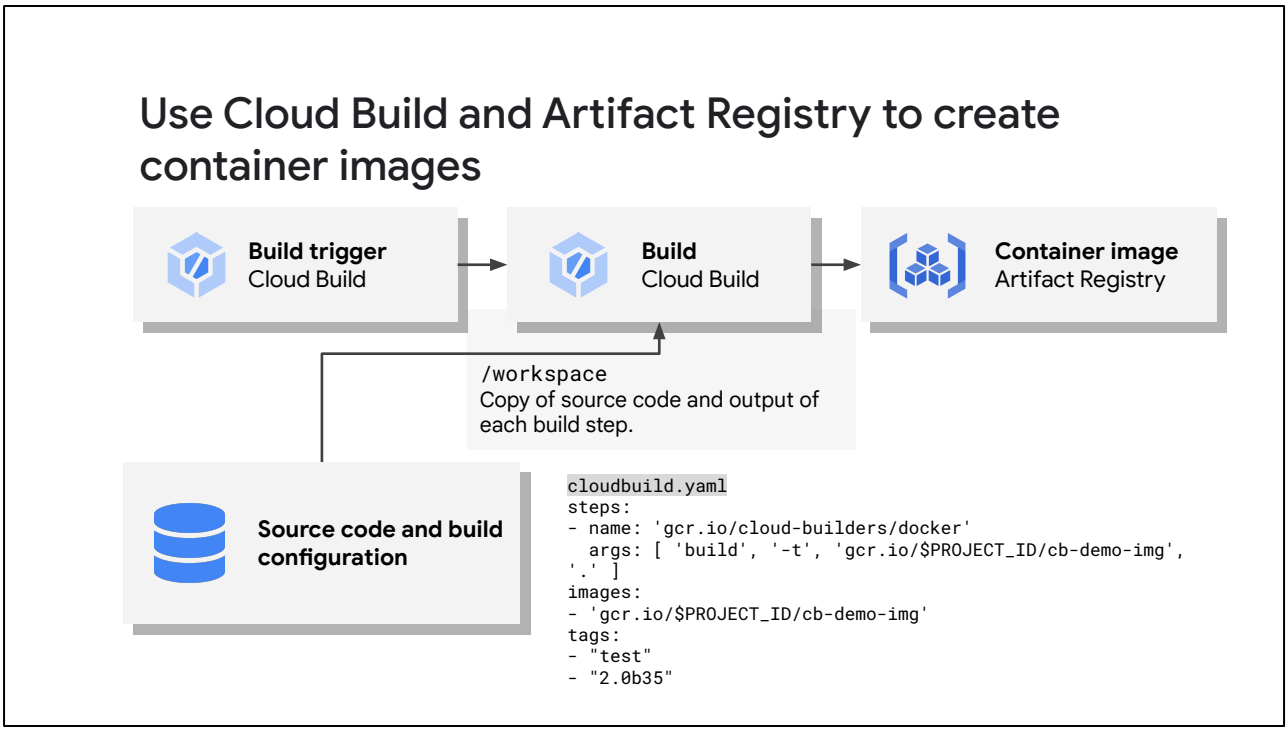


So what does a container provide that a virtual machine does not?

First, containers provide a clear separation of responsibility. Developers can focus on the application logic and any dependencies that are required for the application. The IT operations teams that will deploy and manage the application don't need to worry about application details like software versions and configurations.

Second, containers provide workload portability. Containers are lightweight and can run virtually anywhere, from a developer's laptop to a VM running on-premises or in any cloud. The same application that is tested by a developer on their laptop and tested in an integration environment can run in the production environment. This workload portability simplifies promotion of the app during the development lifecycle and lets you move workloads between clouds and data centers with minimal effort.

Third, containers provide application isolation. Containers virtualize CPU, memory, storage, and network resources at the operating system level. Applications are effectively running in their own environments, which lets containerized applications running on the same hardware use different versions of dependencies without affecting each other. By abstracting just the OS instead of the whole virtual computer, a container can "boot" in a fraction of a second. A virtual machine typically takes a minute or more to boot.



The container image for your application is a complete package that contains the application binary and all the software required for the application to run. When you deploy the same container image in your development, test, and production environments, your application will perform the same way in each environment. Cloud Build is a fully managed service that lets you set up build pipelines to create a Docker container image for your application and push the image to Artifact Registry. You don't need to download build tools and container images to a build machine or manage your own build infrastructure.

By using Artifact Registry and Cloud Build, you can create build pipelines that are automatically triggered when you commit code to a repository. In Cloud Build, you can create a build trigger that is executed based on a trigger type. A trigger type specifies whether a build should be triggered based on commits to a particular branch in a repository or commits that contain a particular tag.

You create a build configuration file that specifies the steps in the build pipeline. Steps are analogous to commands or scripts that you execute to build your application. Each build step is a Docker container that's invoked by Cloud Build when the build is executed. The step name identifies the container to invoke for the build step. The images attribute contains the name of the container image to be created by this build configuration. Cloud Build lets you use different code repositories, tag container images to enable searches, and create build steps that perform operations like downloading and processing data. The build configuration can be specified in a YAML or JSON format.

Cloud Build mounts your source code into the “/workspace” directory of the Docker container associated with a build step. The artifacts produced by each build step are persisted in the “/workspace” folder and can be used by the following build step. Cloud Build automatically pushes the built container image to Artifact Registry. In Artifact Registry, you can view the status and history of builds for a given container. Cloud Build can also publish build status notifications to Pub/Sub. You can subscribe to these notifications to act based on build status or other attributes.

[Cloud Build overview: https://cloud.google.com/build/docs/overview
Artifact Registry overview: https://cloud.google.com/artifact-registry/docs/overview
Interacting with Docker Hub images:
https://cloud.google.com/build/docs/interacting-with-dockerhub-images
Build container images: https://cloud.google.com/build/docs/building/build-containers
Subscribing to build notifications:
https://cloud.google.com/build/docs/subscribe-build-notifications]

In this module, you learned ...

- How continuous integration and delivery can be used to build a repeatable process for building and deploying your applications.
- How to build and maintain container images by using **Cloud Build** and **Artifact Registry.**

In this module, you learned how continuous integration and delivery can be used to build a repeatable process for building and deploying your applications.

We also discussed how to build and maintain container images by using Cloud Build and Artifact Registry.