Google Cloud

Partner Certification Academy

Professional Cloud Developer

The information in this presentation is classified:

Google confidential & proprietary

⚠ This presentation is shared with you under NDA.

- Do **not** record or take screenshots of this presentation.
- Do **not** share or otherwise distribute the information in this presentation with anyone **inside** or **outside** of your organization.

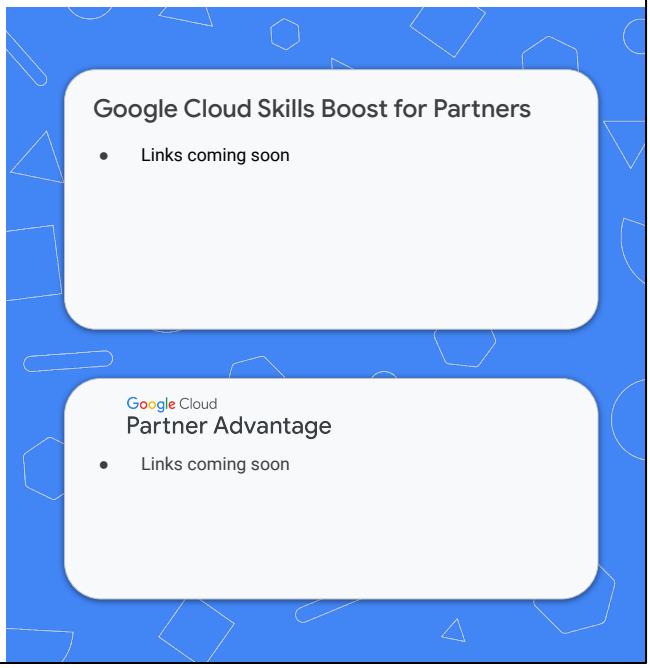
Thank you!

Google Cloud

Source Materials

Some of this program's content has been sourced from the following resources:

- Google Cloud certification site
- Google Cloud documentation
- Google Cloud console
- Google Cloud courses and workshops
- Google Cloud white papers
- Google Cloud Blog
- Google Cloud YouTube channel
- Google Cloud samples
- Google codelabs
- Google Cloud partner-exclusive resources



Session logistics

- When you have a question, please:
  - Click the Raise hand button in Google Meet.
  - Or add your question to the Q&A section of Google Meet.
  - Please note that answers may be deferred until the end of the session.
- These slides are available in the Student Lecture section of your Qwiklabs classroom.
- The session is **not recorded.**
- Google Meet does not have persistent chat.
  - If you get disconnected, you will lose the chat history.
  - Please copy any important URLs to a local text file as they appear in the chat.

Module Agenda

01 Synchronous communication (HTTP/S endpoints)

02 Asynchronous communication
(Cloud Tasks, Pub/Sub, Cloud Scheduler and Eventarc)

03 Cloud Build

04 Artifact Registry

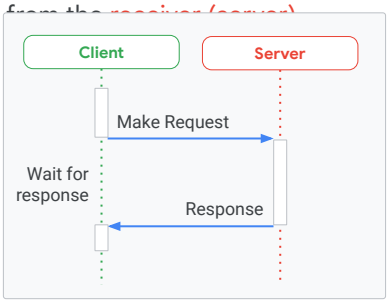
05 Binary Authentication

06 Operations

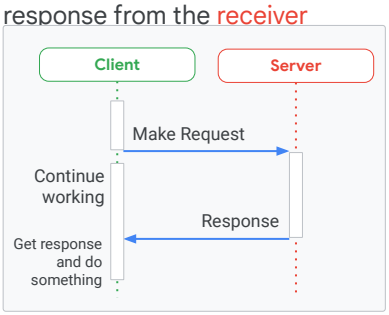
07 Next workshop's assigned content

Asynchronous vs Synchronous communication

Synchronous communication is a type of communication in which the sender of a message (client) waits for a response from the receiver (server)



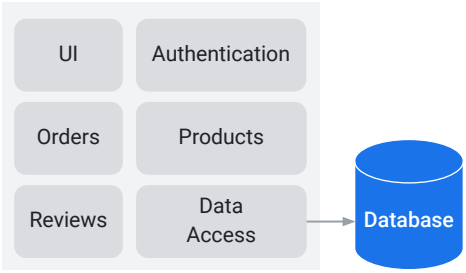
Asynchronous communication is a type of communication in which the sender of a message (client) does not wait for a response from the receiver



* Identity Platform may be used as well

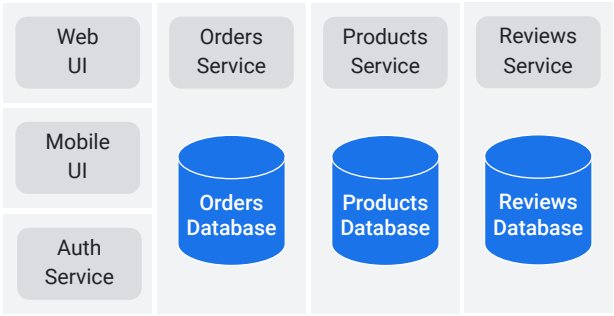
Asynchronous vs Synchronous communication between Monolithic or Microservices applications

Monolithic applications implement all features in a single code base with a database for all data.



Usually synchronous

Microservice have multiple code bases, and each service manages its own data.

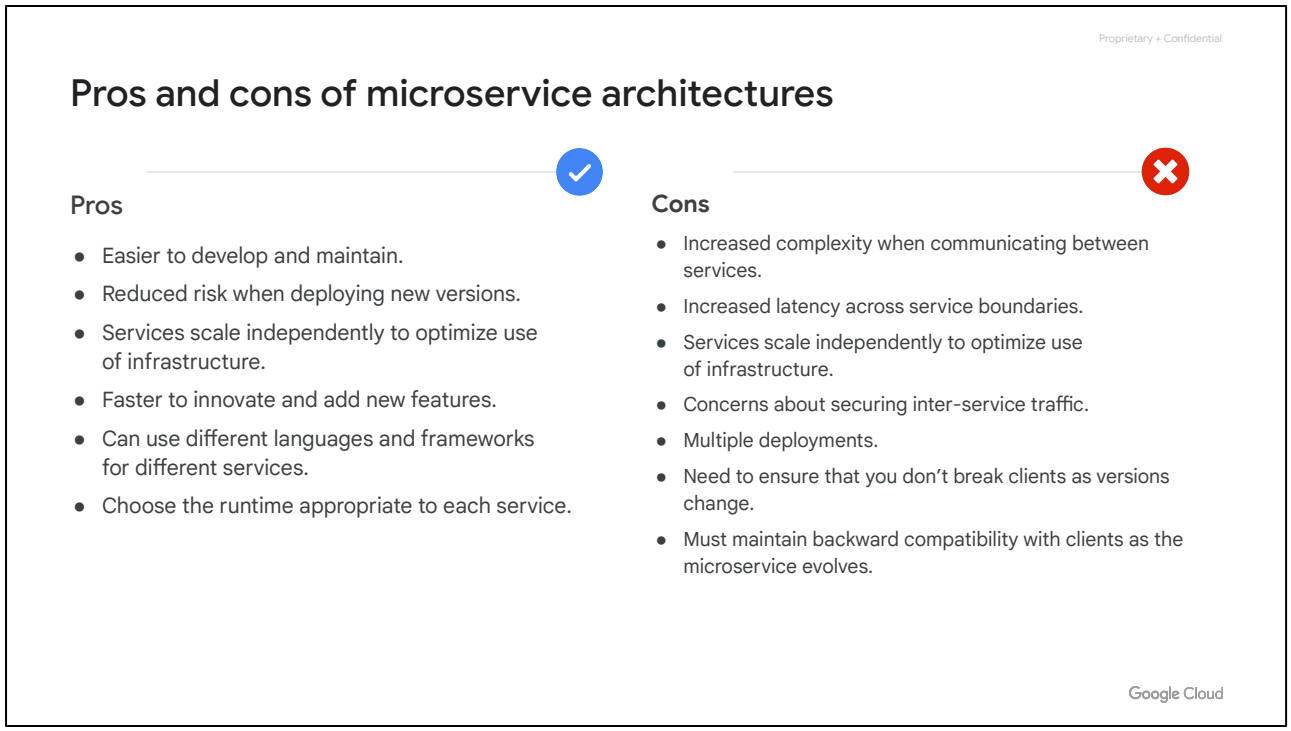


Usually asynchronous

Microservice architecture is the current trend. It is important to ensure that there is a good reason to select this architecture. The primary reason is to enable teams to work independently and deliver through to production at their own cadence. This supports scaling the organization: adding more teams increases speed. There is also the additional benefit of being able to scale the microservices independently based on their requirements.

Architecturally, a monolith or microservices should be modular components with clearly defined boundaries. With a monolith, the deployment is the grouping of the components, whereas with microservices, the individual components are deployable. Google Cloud provides several services for deploying microservices from App Engine, Cloud Run, GKE, and Cloud Functions. Each offers different levels of granularity and control and will be discussed later in the course.

To achieve independence on services, each service should have its own datastore. This lets the best datastore solution for that service be selected. Google Cloud is strong here, with a wide selection of datastores.



A properly implemented microservices-based application can achieve the following goals:

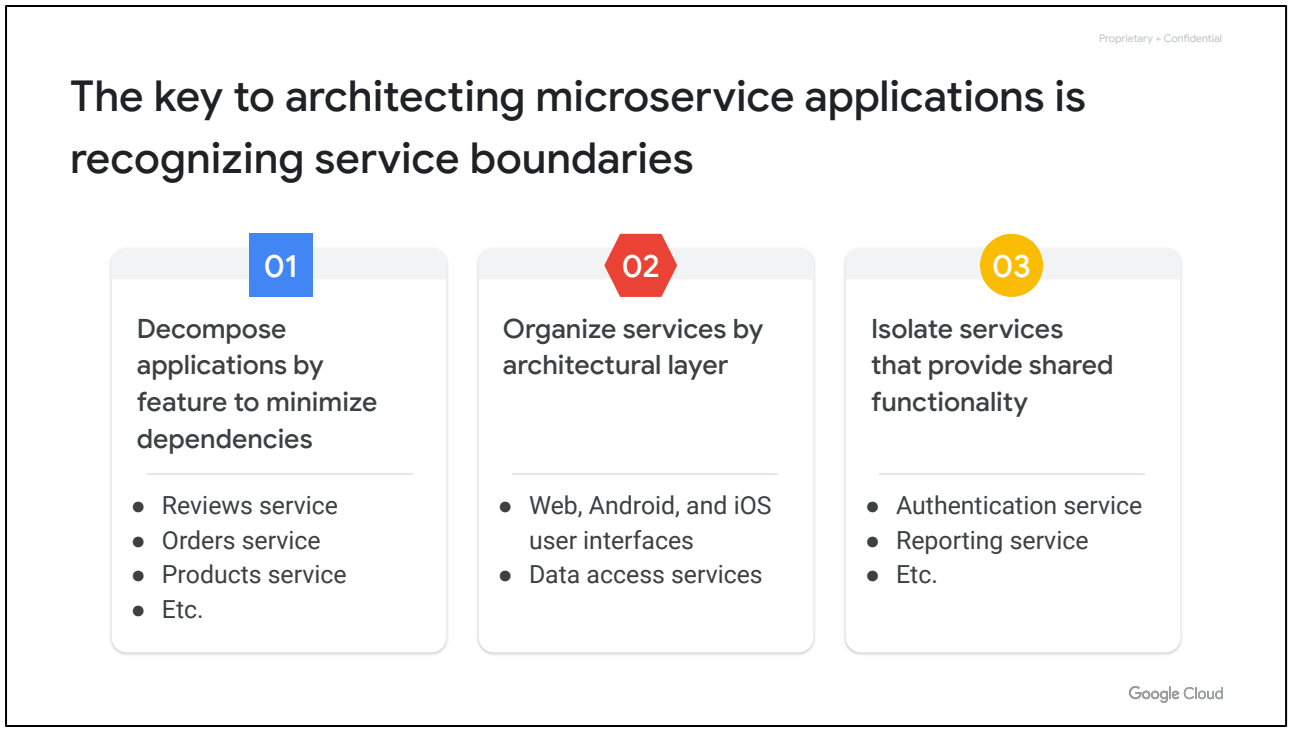
- Define strong contracts between the various microservices
- Allow for independent deployment cycles, including rollback
- Facilitate concurrent, A/B release testing on subsystems
- Minimize test automation and quality assurance overhead
- Improve clarity of logging and monitoring
- Provide fine-grained cost accounting
- Increase overall application scalability and reliability through scaling smaller units

However, the advantages must be balanced with the challenges this architectural style introduces. Some of these challenges include:

- Difficult to define clear boundaries between services to support independent development and deployment
- Increased complexity of infrastructure, with distributed services having more points of failure
- The increased latency introduced by network services and the need to build in resilience to handle possible failures and delays
- Due to the networking involved, there is a need to provide security for service-to-service communication, which increases complexity of infrastructure
- Strong requirement to manage and version service interfaces. With

- independently deployable services, the need to maintain backward compatibility increases.

More on the drawbacks can be found here
http://www.ptone.com/dablog/2015/07/microservices-may-be-the-new-premature-optimization/

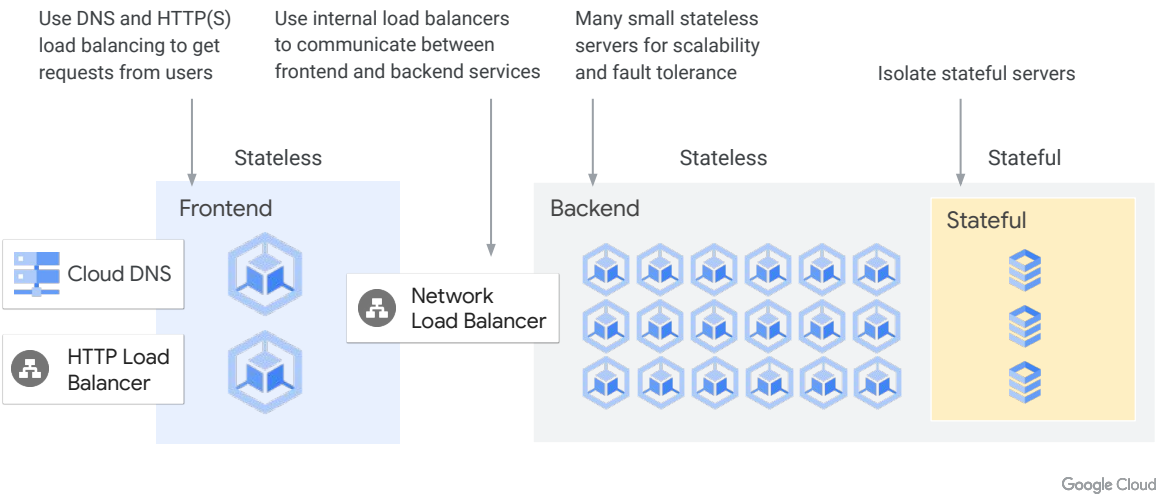


Decomposing applications into microservices is one of the biggest technical challenges of application design. Here techniques like domain-driven design are extremely useful in identifying logical functional groupings. https://en.wikipedia.org/wiki/Domain-driven_design

Consider, for example, an online retail application. Logical functional groupings could be product management, reviews, accounts, orders. These groupings then form mini applications which expose an API. Each of these mini applications will be implemented by potentially multiple microservices internally. Internally these microservices are organized by architectural layer, and each is independently deployable and scalable. Each in theory can be implemented in the language most suitable.

Any analysis will also identify shared services, such as authentication, which then are isolated and deployed separately from the mini applications.

A general solution for large-scale cloud-based systems



The slide displays a general solution that shows the separation of the frontend and backend processing stages. A load balancer distributes the load between the backend and frontend services. This allows the backend to scale if it needs to keep up with the demand from the frontend. In addition, the stateful servers/services are also isolated.

The layout above allows a large part of the application to make use of the scalability and fault tolerance of Google Cloud services as stateless services. By isolation of the stateful servers and services, the challenges of scaling and upgrading are limited to a subset of the overall set of services.

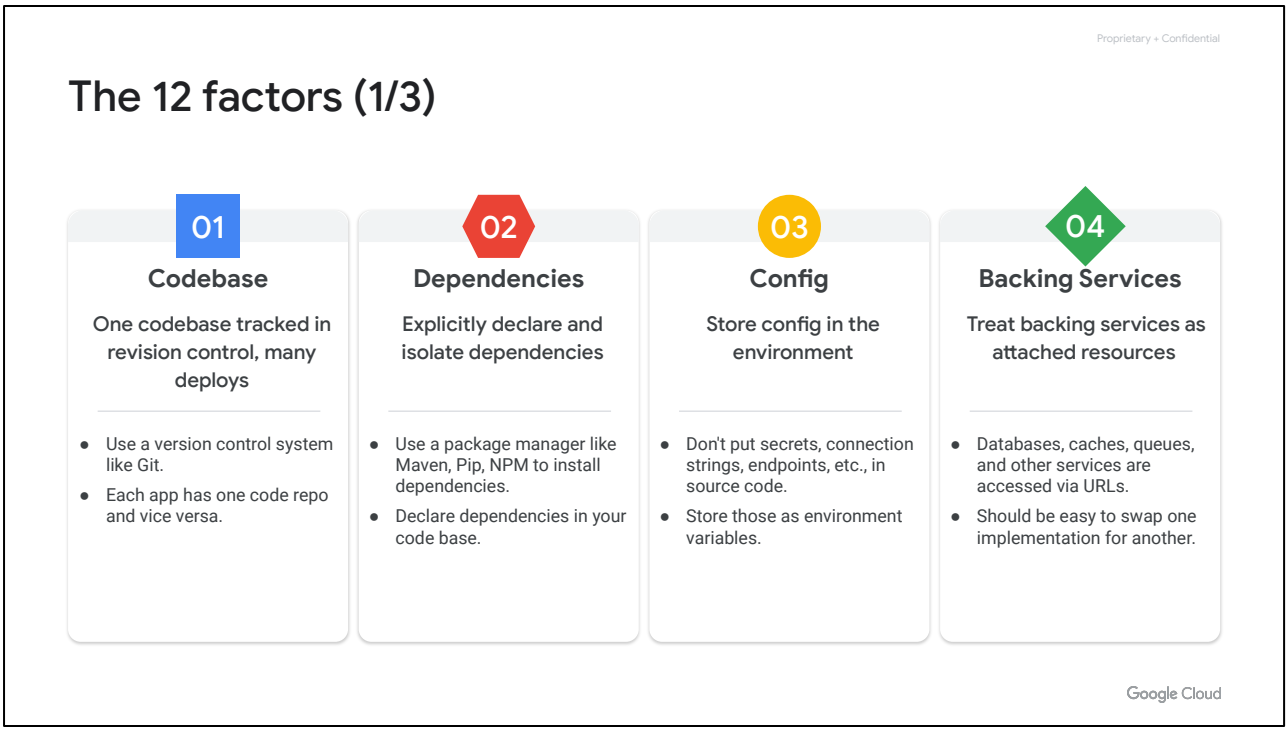
The Twelve-Factor App is a set of best practices for building web or software-as-a-service applications

- Maximize portability
- Deploy to the cloud
- Enable continuous deployment
- Scale easily

https://12factor.net

Twelve-factor design also helps you decouple components of the application, so that each component can be replaced easily or scaled up or down seamlessly. Because the factors are independent of any programming language or software stack, 12-factor design can be applied to a wide variety of applications. Based on the experience of building cloud-native applications, extra factors above the 12 mentioned on the slide are now openly discussed in the community. For example, factors such as an API first strategy, stateless services, telemetry/observability, and security are often added.

An example article can be found here:
https://content.pivotal.io/blog/beyond-the-twelve-factor-app



Codebase
The codebase should be in version control such as Git. Cloud Source Repositories provides fully featured private repositories.
https://cloud.google.com/source-repositories/

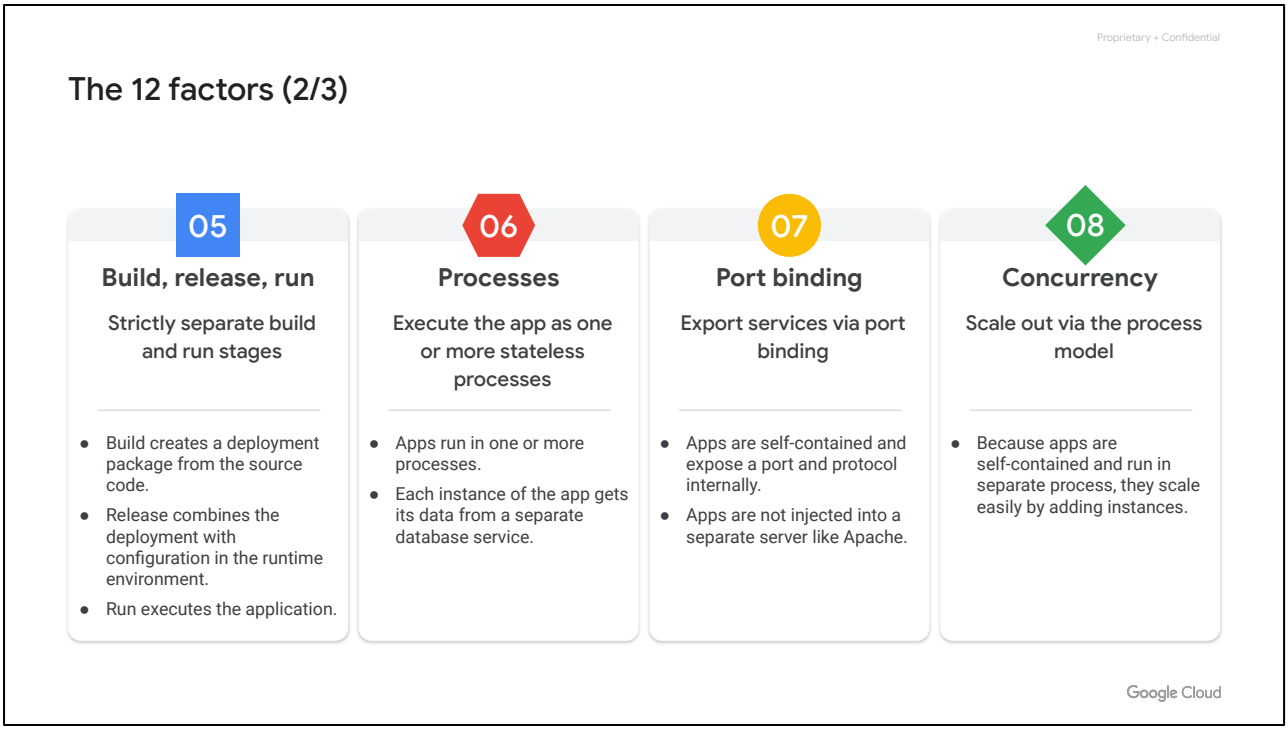
Dependencies
There are two considerations when it comes to dependencies for 12-factor apps: dependency declaration and dependency isolation.

Dependencies should be declared explicitly and stored in version control. Dependency tracking is performed by language-specific tools such as Maven for Java and Pip for Python. An app and its dependencies can be isolated by packaging them into a container. Container Registry can be used to store the images and provide fine-grained access control.

Config
Every app has configuration for different environments: test, production, development. This configuration should be external to the code and usually kept in environment variables.

Backing services
Every backing service, such as a database, cache, or message service, should be accessed via URLs and set by configuration. The backing services act as abstractions for the underlying resource.

For more details see:
https://cloud.google.com/solutions/twelve-factor-app-development-on-gcp

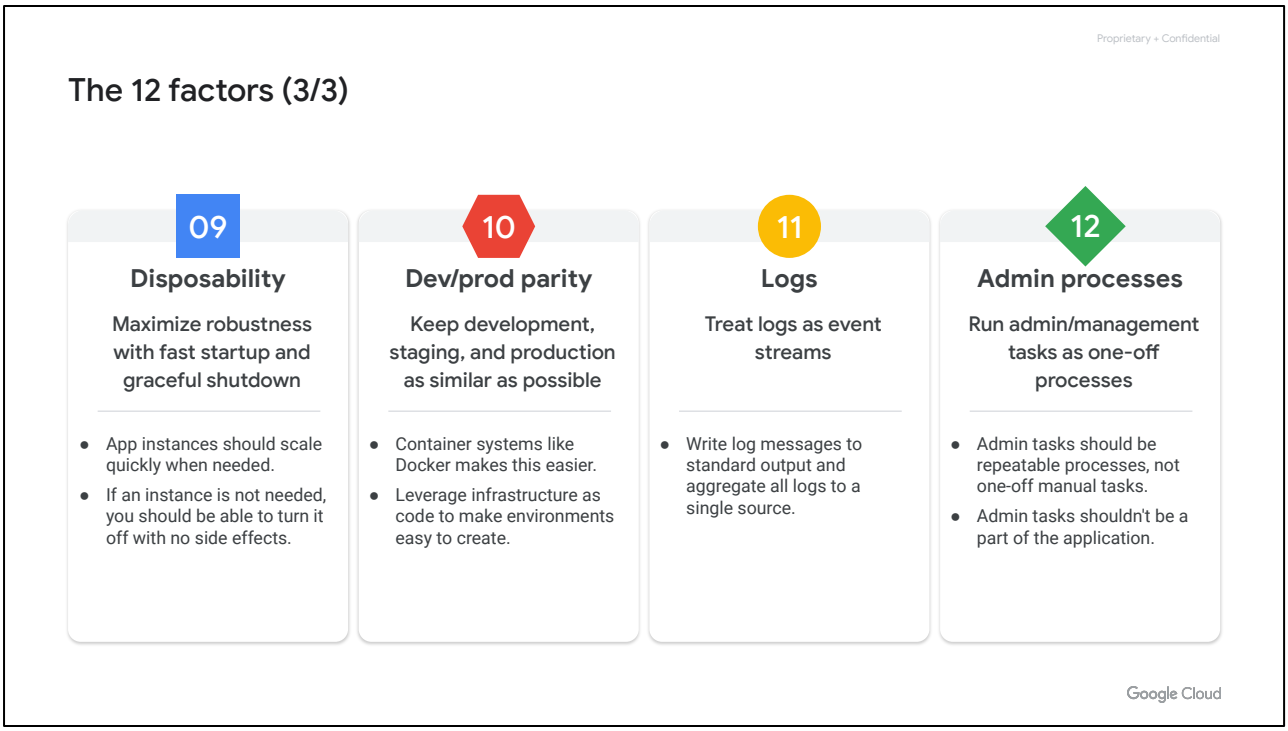


Build, release, run
The software deployment process should be broken into three distinct stages: build, release, and run. Each stage should result in an artifact that's uniquely identifiable. Every deployment should be linked to a specific release that's a result of combining an environment's configuration with a build. This allows easy rollbacks and a visible audit trail of the history of every production deployment.

Processes
Apps run as one or more stateless services. If state is required, then the technique discussed earlier in this module can be used. For example, use Memorystore to cache and share common data between services:
https://cloud.google.com/memorystore/

Port Binding
Services should be exposed using a port number. The apps bundle the web server as part of the app. In Google Cloud, such apps can be deployed on platform services such as Compute Engine, GKE, App Engine, or Cloud Run.

Concurrency
The app should be able to scale out by starting new processes and scale back in as needed to meet demand/load.



Disposability
Applications should be written to be more reliable than the underlying infrastructure they run on. This means they should be able to handle temporary failures in the underlying infrastructure and gracefully shut down and restart.

Dev/production parity
The aim should be to have the same environments used in development and test/staging as are used in production. Google Cloud provides several tools that can be used to build workflows that keep the environments consistent. These tools include Cloud Source Repositories, Cloud Storage, Container Registry, and Terraform. Terraform uses the underlying APIs of each Google Cloud service to deploy your resources.

Logs
Logs provide an awareness of the health of your apps. It's important to decouple the collection, processing, and analysis of logs from the core logic of your apps. Decoupling logging is particularly useful when your apps require dynamic scaling and are running on public clouds, because it eliminates the overhead of managing the storage location for logs and the aggregation from distributed (and often ephemeral) VMs or containers.

Google Cloud offers a suite of tools that help with the collection, processing, and structured analysis of logs.

Admin processes
These are usually one-off processes and should be decoupled from the application. Depending on your deployment on Google Cloud, there are many options for this including:
Cron jobs in GKE https://cloud.google.com/kubernetes-engine/docs/how-to/cronjobs
Cloud tasks on App Engine https://cloud.google.com/tasks/docs/
Cloud Scheduler https://cloud.google.com/scheduler/

Google Extends 12 factor
#13 - Think API First

- Use strong contracts
  To achieve independence, each microservice must provide a versioned, well-defined contract to its clients, which are other microservices.
- Make Microservices Addressable
- Use Versions of APIs
- Consider gRPC
  gRPC is a lightweight protocol for fast, binary communication between services or devices.

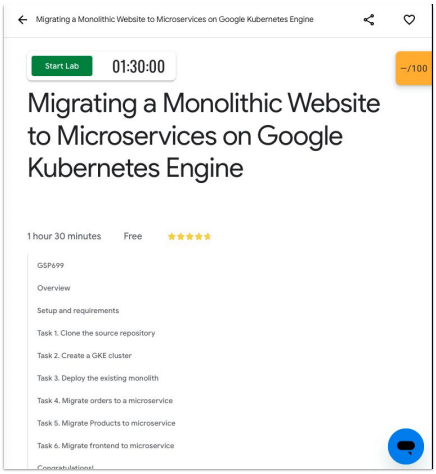
Apigee API Platform

https://cloud.google.com/appengine/docs/legacy/standard/python/designing-microservice-api#using_strong_contracts

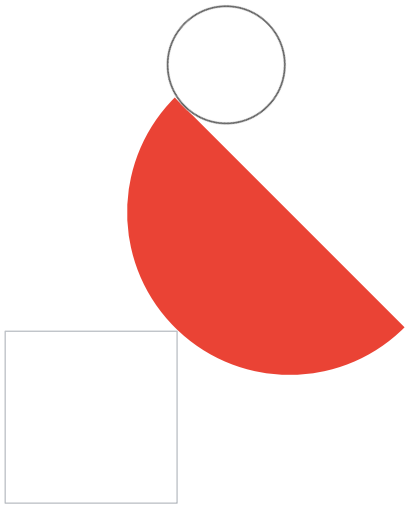
Lab - Migrating a Monolithic Website to Microservices on Google Kubernetes Engine

In this lab you'll start by breaking the monolith into three microservices, one at a time. The microservices include, Orders, Products, and Frontend. Build a Docker image for each microservice using Cloud Build, then deploy and expose the microservices on Google Kubernetes Engine (GKE) with a Kubernetes service type LoadBalancer.

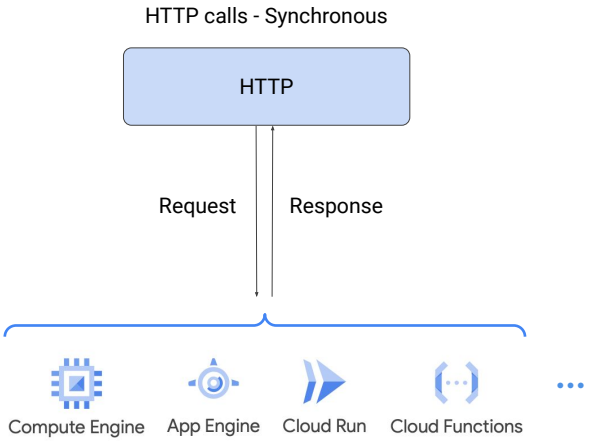
https://partner.cloudskillsboost.google/catalog_lab/2546



Synchronous communication



Synchronous communication



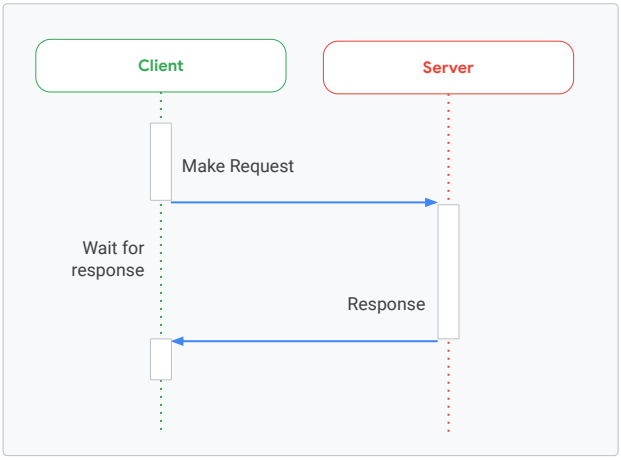
Recommendations

- **IAM** and a **service identity** based on a per-service user-managed **service account** with least privilege approach.
- Proof of the calling service's identity adding **Google-signed OpenID Connect ID token** as part of the request.

* Identity Platform may be used as well

Synchronous communication cases

- When updating a shared database, as the sender needs to make sure that the update is successful before sending another message.
- When making a payment, as the sender needs to make sure that the payment is processed before sending another message.
- When submitting a form, as the sender needs to make sure that the form is submitted successfully before sending another message.
- **And others when the tasks require tight coupling between services, such as updating a shared database.**



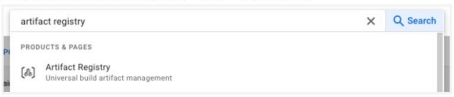
* Identity Platform may be used as well

Demo - Synchronous Communication

In this demo we will create two NodeJs servers, one as a backend and the other as a client.

In the first one, we will create an endpoint to return the response that we want the second server to read using a client load.

1. Create a Project on Google Cloud.
2. Search for **Artifact Registry** on the Search Bar



a. Click to access the Artifact Registry and click Create GCR.IO Repositories

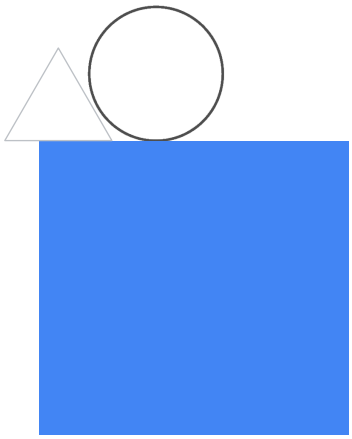
b. Click create

c. Click Create Repository

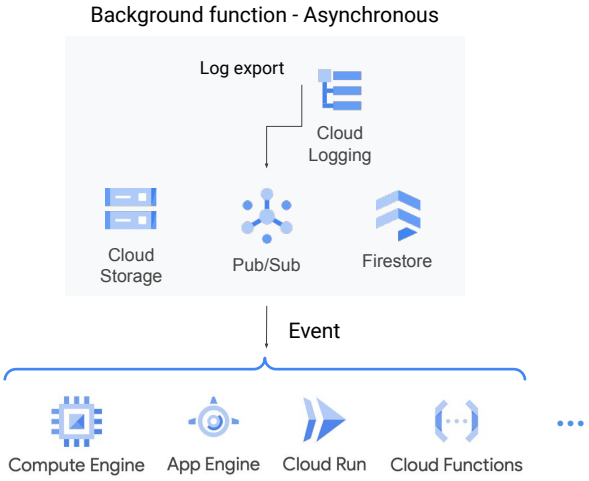
i. Set the name and location
ii. Click create

iii. Click on the repository that was created
iv. Click on Setup Instruction to copy the instructions

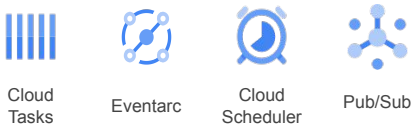
Asynchronous communication



Asynchronous communication



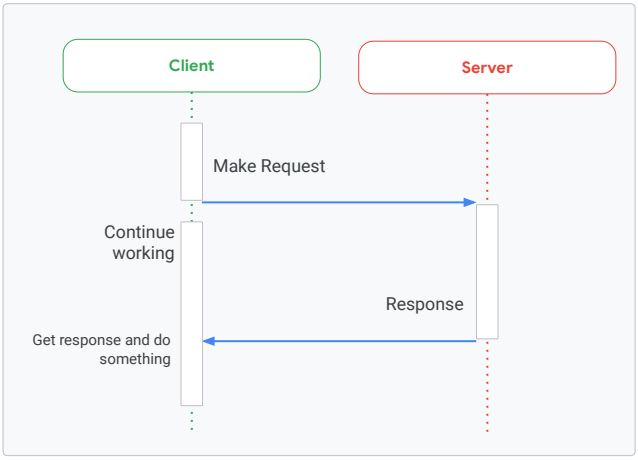
- In the context of microservices, synchronous communication is often used for tasks that require tight coupling between services
- There are cases where asynchronous communication is necessary, such as when a downstream application is often offline.



* Identity Platform may be used as well

Asynchronous communication cases

- When sending a notification, as the sender does not need to wait for the recipient to read the notification before sending another message.
- When scheduling a job, as the sender does not need to wait for the job to be completed before sending another message.
- When fetching data from a slow API, as the sender can continue sending other messages while waiting for the data to be fetched.
- **And others when the tasks require loose coupling between services, such as sending a notification or scheduling a job.**



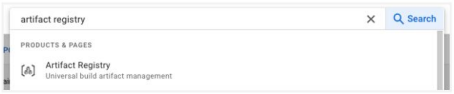
* Identity Platform may be used as well

Demo - Asynchronous Communication

In this demo you will create a service in Cloud Run with an endpoint waiting for a message to be pushed via Pub/Sub.

Then we will create a topic and a push subscription to send the message to the endpoint in your Cloud Run.

1. Create a Project on Google Cloud.
2. Search for **Artifact Registry** on the Search Bar



a. Click to access the Artifact Registry and click Create GCR.IO Repositories

b. Click create

c. Click Create Repository

i. Set the name and location
ii. Click create

iii. Click on the repository that was created
iv. Click on Setup Instruction to copy the instructions

Asynchronous communication

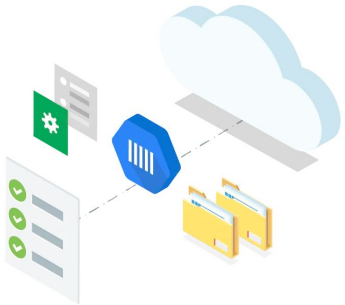
- Cloud Tasks
- Pub/Sub
- Cloud Scheduler
- Eventarc

Asynchronous communication

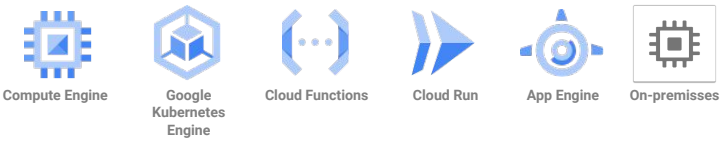
- Cloud Tasks
- Pub/Sub
- Cloud Scheduler
- Eventarc

Cloud Tasks

- Asynchronous task execution - a fully managed service
- Task Deduplication
- Guaranteed Delivery
- Schedule when a task is run
- Distributed Task Queues
- Decouple and scale micro services
- Manage resource utilization



HTTP Targets in GCP or On-Premise



Cloud Tasks overview | Cloud Tasks Documentation | Google Cloud
Cloud Tasks Service For Asynchronous Execution | Google Cloud

https://cloud.google.com/tasks/docs
Cloud Tasks is a fully managed service that allows you to manage the execution, dispatch and delivery of a large number of distributed tasks. You can asynchronously perform work outside of a user request. Your tasks can be executed on App Engine or any arbitrary HTTP endpoint.

So not just Google Cloud, can be anything that has an HTTP endpoint

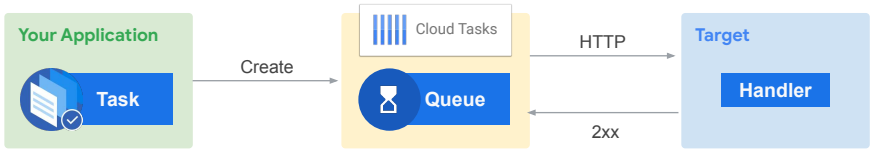
It falls under Developer Tools

GA Apr 9, 2019

- https://cloud.google.com/tasks/docs/release-notes#April_09_2019
- Initially for AppEngine

Cloud Tasks with HTTP Queues

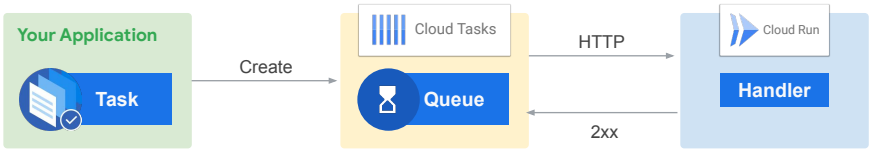
Cloud Task Service forwards request to any generic HTTP target
Target must manage scaling workers and cleaning up tasks once they are complete



Cloud Tasks overview | Cloud Tasks Documentation | Google Cloud
Cloud Tasks Service For Asynchronous Execution | Google Cloud

Cloud Tasks with Cloud Run Targets

Cloud Task Service forwards request to the worker
Cloud Task Service can handle much of the process management for the task, scaling workers and deleting completed tasks.



[Cloud Tasks overview | Cloud Tasks Documentation | Google Cloud]
[Cloud Tasks Service For Asynchronous Execution | Google Cloud]

Recommended demo

Cloud Tasks and Cloud Scheduler demonstration and explanation

https://www.youtube.com/watch?v=P9MCC9KmM_8

Asynchronous communication

- Cloud Tasks
- **Pub/Sub**
- Cloud Scheduler
- Eventarc

Pub/Sub offers reliable, real-time messaging

Distributed messaging with Pub/Sub

- At-least-once delivery
- No provisioning, auto-everything
- Open APIs
- Global by default
- End-to-end encryption

One tool to handle distributed message-oriented architectures at scale is Pub/Sub. The name is easy to remember because it's Publisher / Subscriber or publish messages to subscribers.

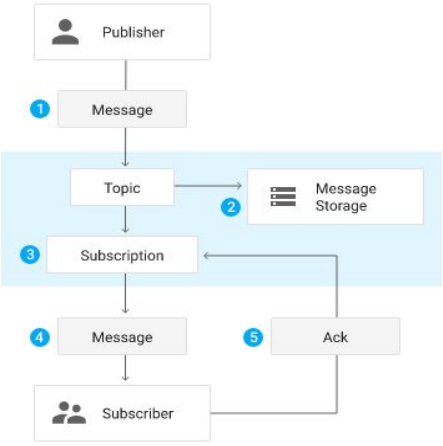
Pub/sub is a distributed messaging service that can receive messages from a variety of device streams like gaming events, IoT devices, application streams and more.

It ensures at-least once delivery of messages received to subscribing applications and no provisioning is required -- whether its a ton of messages or none pub/sub will scale to meet it

The APIs are open and the service is global by default and offers end-to-end encryption.

https://cloud.google.com/pubsub/docs/overview#scenarios
https://cloud.google.com/pubsub/

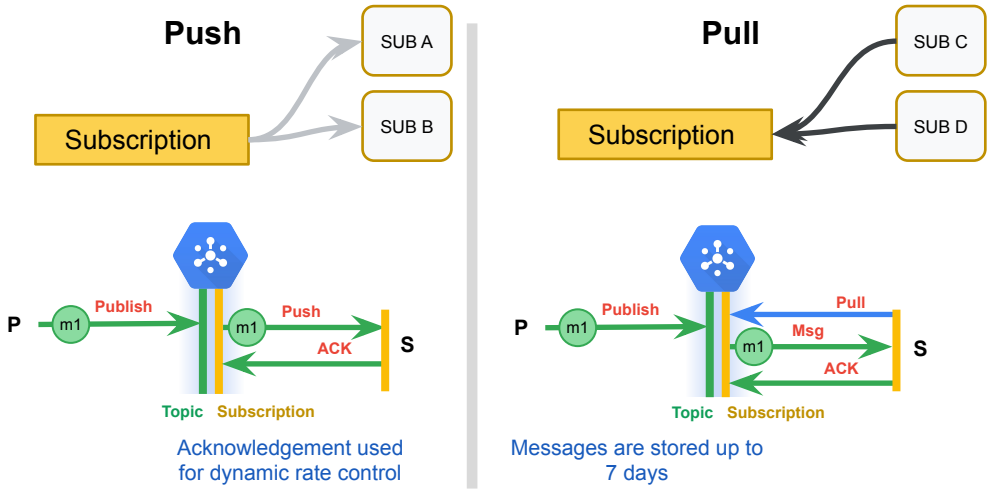
Pub/Sub Concepts and Flow



- **Message** can contain data and attributes
- **Topic** are named endpoints where messages are sent.
- **Subscriptions** represent a stream of messages within a topic
  - **Topics** can contain multiple **subscriptions**
  - Each **subscription** belongs to **one topic**
  - Two types of subscriptions, **push** and **pull**
- **Subscribers** get messages from **subscriptions**, they are applications that process **Pub/Sub messages**

https://cloud.google.com/pubsub/docs/overview#concepts

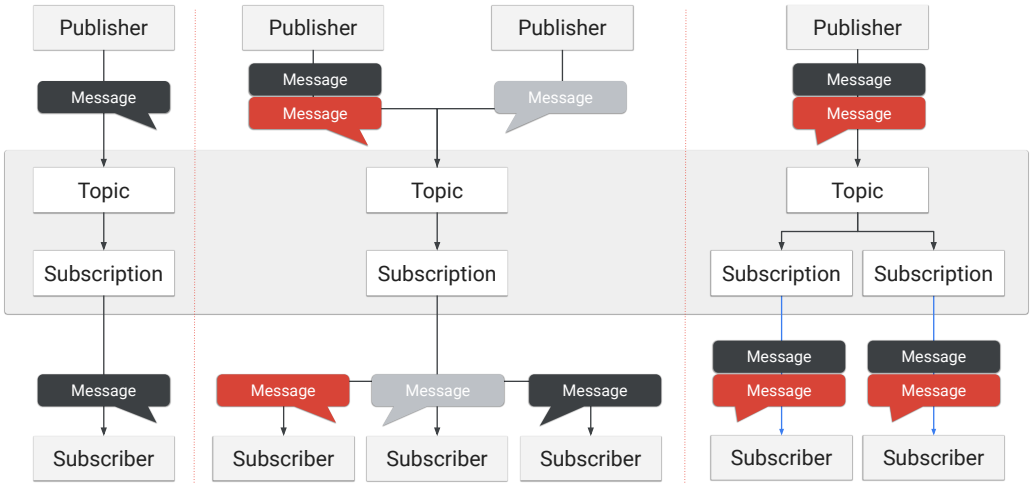
Cloud Pub/Sub provides both Push and Pull delivery



In the Push model, it actually uses an HTTP endpoint. You register an webhook as your subscription, and Pub/Sub infrastructure itself will call you with the latest messages. In the case of Push, you just respond with ‘status 200 ok’ for the HTTP call, and that tells Pub/Sub the message delivery was successful.

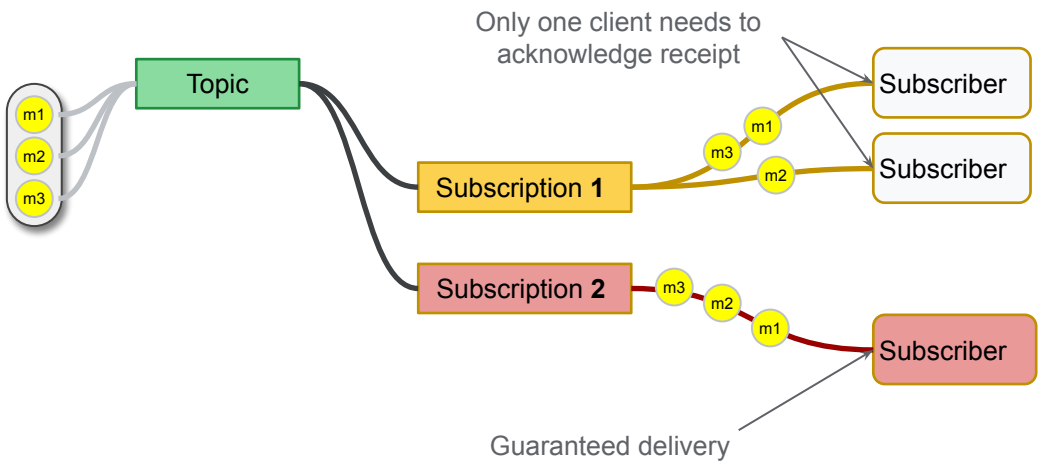
It will actually use the rate of your success responses to self limit so that it doesn’t overload your worker.

Publish/Subscribe patterns



You can use this distribution of messages, the publish and subscribe patterns, to do fan in or fan out. What you see here, the different colors represent different messages.

Subscribers can work as a team or separately

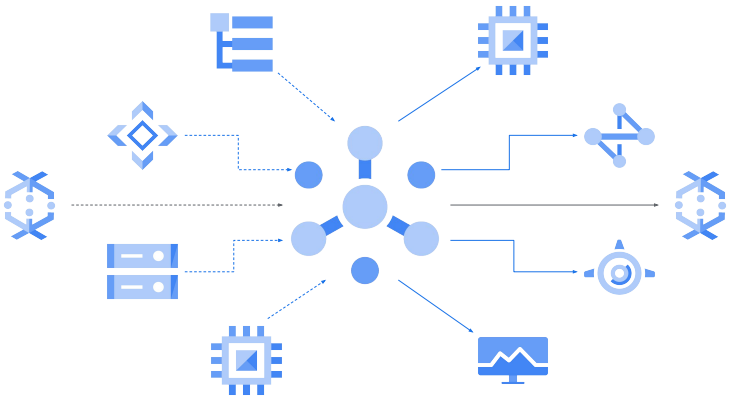


Subscribers can work as a individually or as a group. If we have just one subscriber, it is going to get every message delivered through that subscription. However, you can set up worker pools by having multiple subscribers sharing the same subscription.

In this case, it is going to distribute the message, so one and three go to Subscription 1, and two goes to Subscription 2. And it is just random based on when it pulls from messages throughout the day

In the case of a Push subscription, you only have one web inpoint, so you will only have one subscriber typically. But, that one subscriber could be a app engine standard app, or cloud run container image, which autoscales. So, it is one web endpoint, but it can have autoscale workers behind the scenes. And that is actually a very good pattern.

Pub/Sub offers reliable, real-time messaging to connect asynchronously services



One tool to handle distributed message-oriented architectures at scale is Pub/Sub. The name is easy to remember because it's Publisher / Subscriber or publish messages to subscribers.

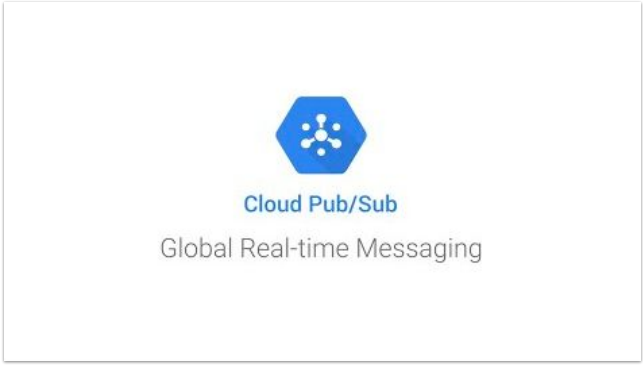
Pub/sub is a distributed messaging service that can receive messages from a variety of device streams like gaming events, IoT devices, application streams and more.

It ensures at-least once delivery of messages received to subscribing applications and no provisioning is required -- whether its a ton of messages or none pub/sub will scale to meet it

The APIs are open and the service is global by default and offers end-to-end encryption.

https://cloud.google.com/pubsub/docs/overview#scenarios
https://cloud.google.com/pubsub/

Recommended demo



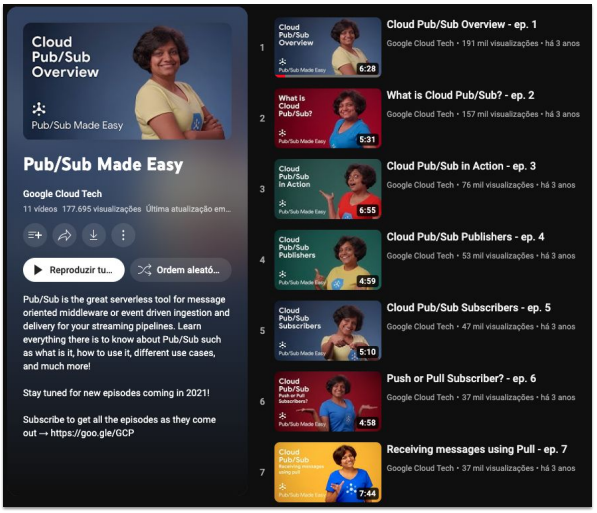
This video walks you through how to create a Google Cloud Pub/Sub topic. Pub/Sub allows you to create global real-time messaging and subscribe to topics you want to follow to stay up to date.

https://www.youtube.com/watch?v=pU1zA-DMlWk

Pub/Sub Made Easy

Learn everything there is to know about Pub/Sub such as what is it, how to use it, different use cases, and much more!

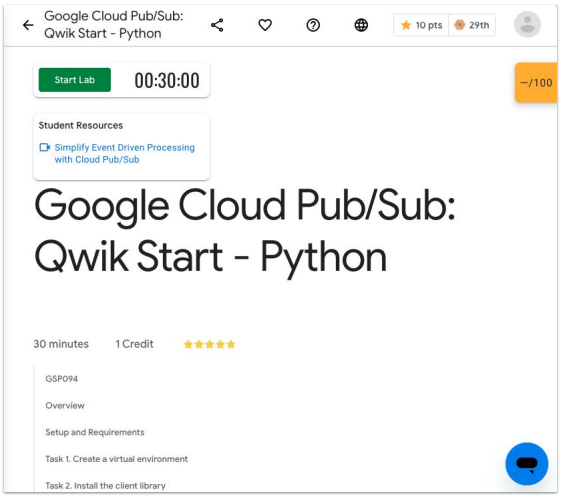
https://www.youtube.com/playlist?list=PLlivdWyY5sqKwVLe4BLJ-vlh9r9zCdOse



Lab - Google Cloud Pub/Sub: Qwik Start - Python

In this lab, you will learn how to get started publishing messages with Cloud Pub/Sub using the Python client library.

https://www.cloudskillsboost.google/focuses/2775?parent=catalog



Guide: Pub/Sub

- Overview
- Pub/Sub Publishers
- Pub/Sub Subscribers
- Push or Pull Subscriber
- Receiving messages using Pull
- Receiving Messages using Push
- Using Cloud Pub/Sub with Cloud Run
- Choosing Pub Sub or Pub Sub Lite?

Pub/Sub in a minute

Pub/Sub is an asynchronous messaging service that decouples services that produce events from services that process events.

Cloud Pub/Sub in a minute

Pub/Sub Made Easy

Learn everything there is to know about Pub/Sub such as what it is, how to use it, different use cases, and much more.

Pub/Sub

Asynchronous communication

- Cloud Tasks
- Pub/Sub
- **Cloud Scheduler**
- Eventarc

Cron Job

A cron job is a task scheduler that runs a program on a server at a specified interval. The cron utility has a highly customizable format, called a crontab file, which allows us to run essentially any command at whatever interval we want.

Cron jobs are a powerful tool that can be used to automate tasks on a server. They are easy to set up and use, and they can be used to run any command at whatever interval we want.

```
# 0 0 * * * /usr/local/bin/application.py
# │ │ │ │ │ └── run this script
# │ │ │ │ └──── every year
# │ │ │ └────── every month
# │ │ └──────── every day
# │ └────────── at hour zero
# └──────────── at minute zero
```

It has one huge downside: We need to keep my server up and running to ensure that code will run

Cloud Scheduler

Is a fully managed enterprise-grade cron job scheduler that allows you to schedule virtually any job, including **batch**, **big data** jobs, cloud **infrastructure operations**, and more.

You can automate everything, including retries in case of failure to reduce manual toil and intervention. **Cloud Scheduler** even acts as a single pane of glass, allowing you to manage all your automation tasks from one place.

Cloud Scheduler

Each cron job created using Cloud Scheduler is sent to a target according to a specified schedule, where the work for the task is accomplished. The target must be one of the following types:

- **Publicly available HTTP/S endpoints**
- **Pub/Sub topics**
- **App Engine HTTP/S applications**

```
$ gcloud beta scheduler jobs create http send_pythons_job \
    --schedule="0 0 * * *" \
    --uri=https://us-central1-<PROJECT_ID>.cloudfunctions.net/send_pythons
```

Cloud Functions Endpoint URI

Your Project ID

Cloud Scheduler and Cloud Functions endpoint

Recommended demo

Code from this demo: https://dev.to/googlecloud/moving-your-cron-job-to-the-cloud-with-google-cloud-functions-1ecp

In this video, we'll show you the dexterity of Python, and how you can easily run jobs on a schedule with Cloud Functions and Cloud Scheduler.

https://www.youtube.com/watch?v=7Z1mgOxWTs8

Asynchronous communication

- Cloud Tasks
- Pub/Sub
- Cloud Scheduler
- Eventarc

Eventarc

Eventarc is a fully managed event delivery service that lets you build **event-driven architectures** without having to implement, customize, or maintain the underlying infrastructure. Eventarc offers a standardized solution to manage the flow of state changes, called events, between **decoupled microservices**.

Delivery guarantees
At-least-once or exactly-once delivery guarantees.

Security
Encrypted in transit and at rest.

Authorization
Authorized using IAM roles and permissions.

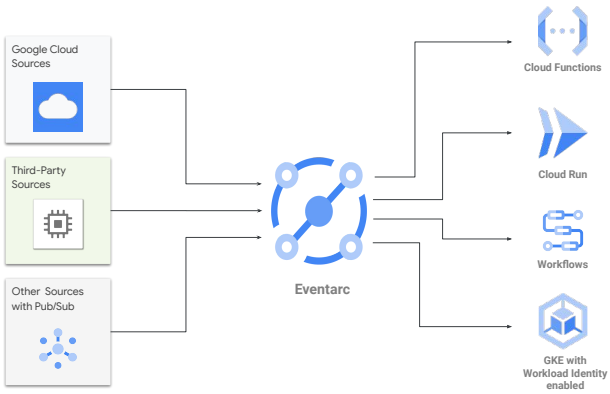
Observability
Logging and monitoring, to help you troubleshoot problems.

Error handling
Can retry failed deliveries.

https://cloud.google.com/eventarc/docs/event-providers-targets

Eventarc

Eventarc is a fully managed event delivery service that lets you build **event-driven architectures** without having to implement, customize, or maintain the underlying infrastructure. Eventarc offers a standardized solution to manage the flow of state changes, called events, between **decoupled microservices**.



https://cloud.google.com/eventarc/docs/event-providers-targets

Eventarc - use cases

Configure and monitor	Harmonize	Analyze
• **System configuration:** Install a configuration management tool on a new VM when it is started. • **Automated remediation:** Detect if a service is not responding properly and automatically restart it. • **Alerts and notifications:** Monitor the balance of a cryptocurrency wallet address and trigger notifications.	• **Directory registrations:** Activate an employee badge when a new employee joins a company. • **Data synchronization:** Trigger an accounting workflow when a prospect is converted in a CRM system. • **Resource labeling:** Label and identify the creator of a VM when it is created.	• **Sentiment analysis:** Use the Cloud Natural Language API to train and deploy an ML model that attaches a satisfaction score to a customer service ticket when it is completed. • **Image retouching and analysis:** Remove the background and automatically categorize an image when a retailer adds it to an object store.

https://cloud.google.com/eventarc/docs/event-providers-targets

Eventarc

Events occur whether or not a target destination reacts to them. You create a response to an event with a trigger. A trigger is a declaration that you are interested in a certain event or set of events. When you create a trigger, you specify filters for the trigger that let you capture and act on those specific events, including their routing from an event source to a target destination.

https://cloud.google.com/eventarc/docs/targets#triggers

```
gcloud functions deploy <YOUR_FUNCTION_NAME> \
  --gen2 \
  --trigger-event-filters=<EVENTARC_EVENT_FILTERS> \
  [--trigger-event-filters-path-pattern=<EVENTARC_EVENT_PATH_PATTERN>] \
  [--trigger-location=<EVENTARC_TRIGGER_LOCATION>] \
  [--trigger-service-account=<EVENTARC_TRIGGER_SERVICE_ACCOUNT>] \
  [--retry] \
...
```

Eventarc Trigger and Cloud Functions

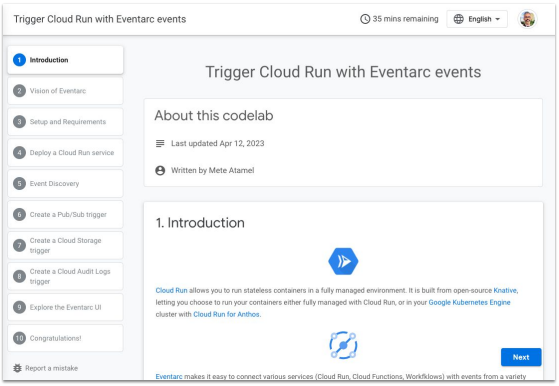
https://cloud.google.com/eventarc/docs/event-providers-targets

Codelab: Trigger Cloud Run with Eventarc events

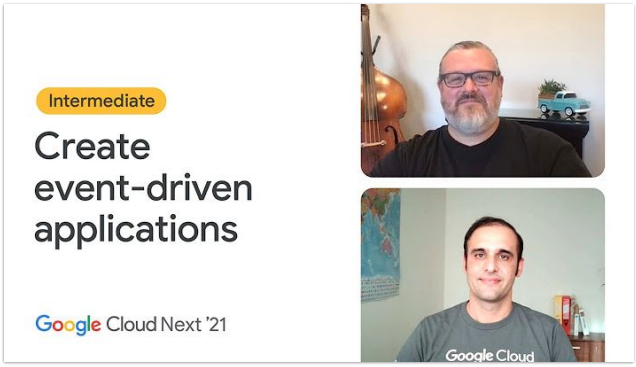
https://codelabs.developers.google.com/codelabs/cloud-run-events

You'll learn:

- Vision of Eventarc
- Discover events in Eventarc
- Create a Cloud Run sink
- Create a trigger for Pub/Sub
- Create a trigger for Cloud Storage
- Create a trigger for Cloud Audit Logs
- Explore the Eventarc UI



Case study demo

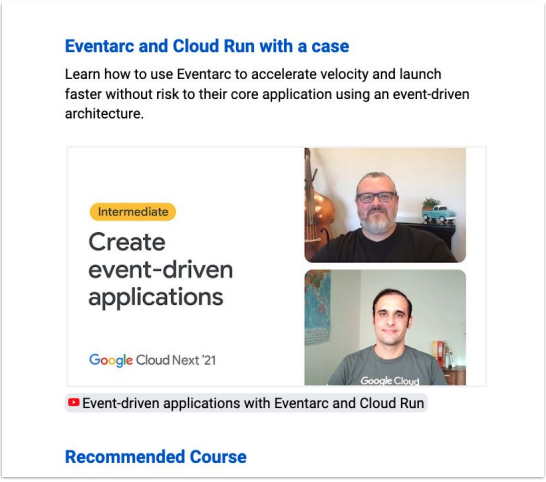


Learn how Vivint used Eventarc to accelerate velocity and launch faster without risk to their core application using an event-driven architecture.

https://www.youtube.com/watch?v=za82mvjur5E

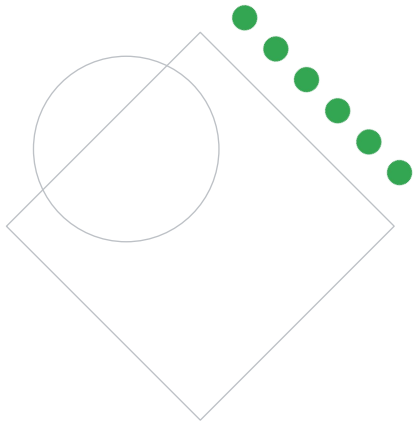
Guide: Eventarc

- Event-driven architecture
- Event providers and destinations
- Eventarc triggers
- CloudEvents
- Source
- Target
- Integrating Eventarc



Eventarc

Cloud Build



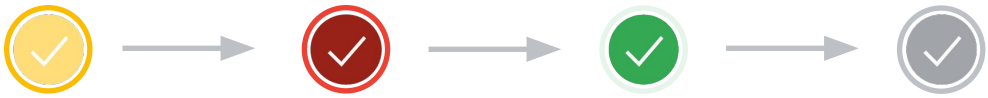
Cloud Build

- Can import source code from a variety of **repositories** or **cloud storage** spaces, execute a build to your specifications, and produce artifacts such as **Docker containers** or **Java** archives.
- Build software quickly across all programming languages, including Java, Go, Node.js, and more
- Deploy across multiple environments such as Vm instances, Cloud Run, Cloud Functions, App Engine and Kubernetes

The build steps from Cloud Build:

- Pre-built steps
- Community-made steps
- Your own steps

Continuous integration pipelines automate building applications



Developers check-in code

Use a Git repo for each microservice and branches for versions.

Run unit tests

If the tests don't pass, stop.

Build deployment package

Create a Docker image.

Deploy

Save your new Docker image in a container registry.

Cloud Source Repositories

- A managed Git repository, act as standalone repository or connect it to an existing **GitHub** or **Bitbucket** repository (Automatically synchronizes)
- When code is pushed to the repository
  - Can publish messages to a specified **Pub/Sub** topic
  - Can also trigger a **Cloud Build** process to
    - Build the code
    - Run unit tests
    - Put the artifacts in the **Artifact Registry**
    - Deploy the code to **Cloud Run**, **GKE**, etc.

Cloud Source Repositories
https://cloud.google.com/source-repositories

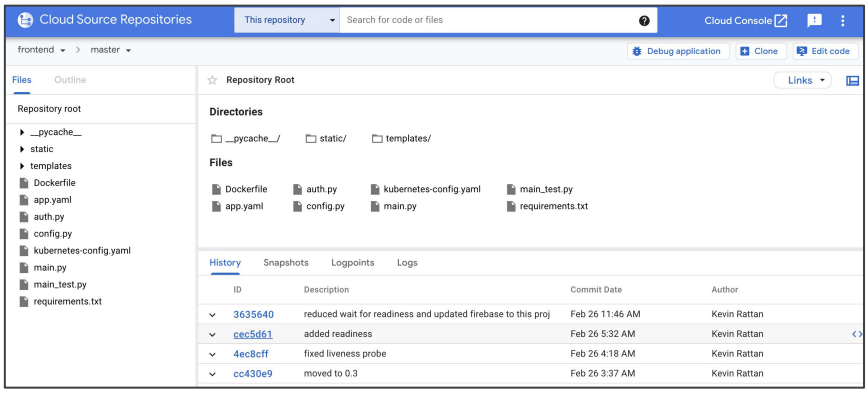
You can use Cloud IAM to add team members to your project and to grant them permissions to create, view, and update repositories.

Repositories can be configured to publish messages to a specified Pub/Sub topic. Messages can be published when a user creates or deletes a repository or pushes a commit.

Some other features of Cloud Source Repositories include the ability to debug in production using Cloud Debugger, audit logging to provide insights into what actions were performed where and when, and direct deployment to App Engine. It is also possible to connect an existing GitHub or Bitbucket repository to Cloud Source Repositories. Connected repositories are synchronized with Cloud Source Repositories automatically.

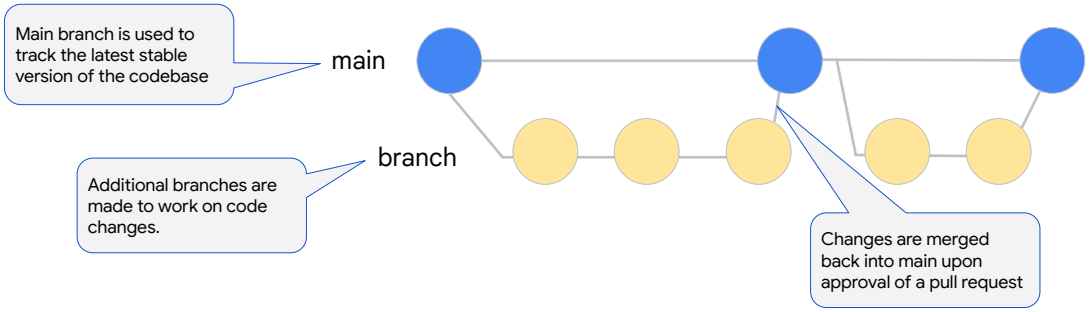
Cloud Source Repositories provides managed Git repositories

Control access to your repos using IAM within your Google Cloud projects.



Source control management is an essential part of cloud native development

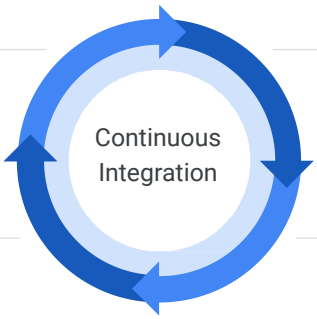
- Git is a distributed version control system that helps developers track changes to code over time
  - One of the most popular version control systems



Google provides the components required for a continuous integration pipeline

Cloud Source Repositories
Developers push to a central repository when they want a build to occur.

Artifact/Container Registry
Store your Docker images or deployment packages in a central location for deployment.



Cloud Build
Build system executes the steps required to make a deployment package or Docker image.

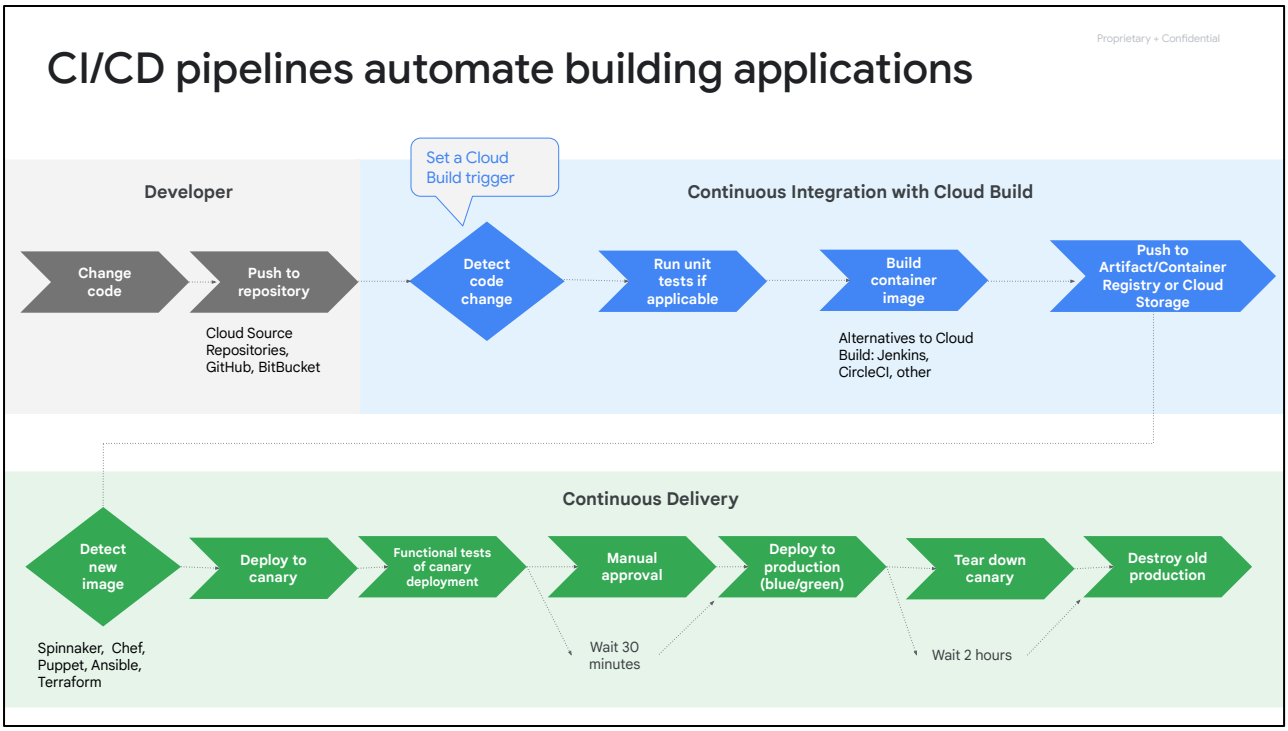
Build triggers
Watches for changes in the Git repo and starts the build.

Cloud Build Serverless CI/CD platform

Cloud Build documentation
https://cloud.google.com/build/docs

Cloud Build - Create a build configuration file (lots of good sections to review - look at the different topics on the left)
https://cloud.google.com/build/docs/configuring-builds/create-basic-configuration

Cloud Build Youtube video
https://www.youtube.com/watch?v=2TZXSnCTd7E&list=PLTWE_Imu2InBzuPmOcgAYP7U80a87cpJd



Source :

Cloud Foundations: Continuous Integration and Delivery (CI/CD) Technical Deep Dive | PSO | Y21
https://docs.google.com/presentation/d/1l0r-fLPHZwBBKUYerGxh4xJQ1d3Nk3TjG7DtzKVo640/edit#slide=id.gf86e597b00_0_1452

Source image edited by BT

TL;DR / Purpose of the slide:

- Define the objective: discussing and designing the CI/CD architecture that will be implemented

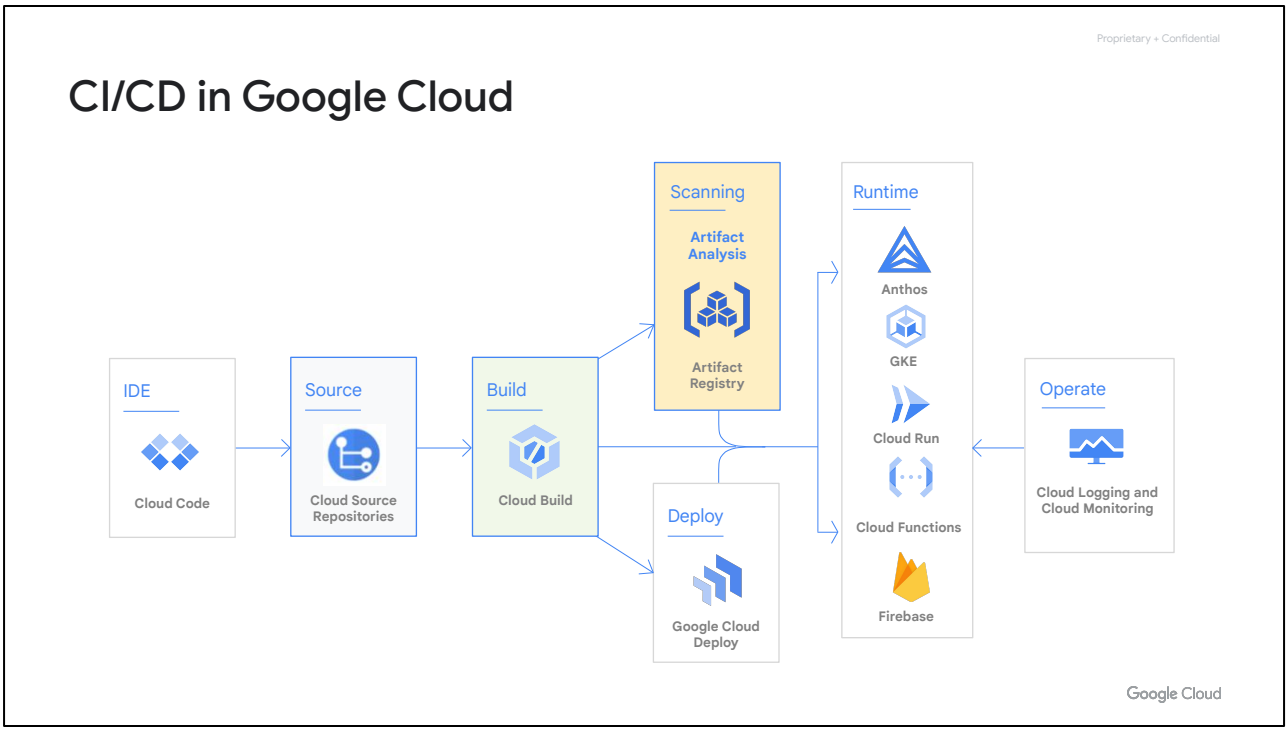
Key points

- The following slides discuss how this can be **achieved using managed and unmanaged solutions**:
  - **CI/CD terminology**
  - **CI/CD high-level architecture on Google Cloud**
  - **Review some of the common managed and unmanaged solutions**

Probing questions (optional):

- Verify with customer that **objective is relevant**

- **Confirm** that deploying a **CI/CD solution** into **Cloud** is part of the **customer's goal**, otherwise skip the section



Google Cloud offers a complete, fully managed, serverless, DevOps pipeline with pluggable systems that can be replaced by any existing third-party tooling. Let's look at the individual components:

- Cloud Code is a plugin to IDEs like VS Code and IntelliJ that allows you to write, debug, and deploy your Kubernetes applications. It also has pre-configured samples so you can start running them in seconds.
- Cloud Source Repositories is a fully managed Git service that allows you to store your code and configure triggers to your CI/CD pipelines.
- Cloud Build is a completely serverless CICD platform. Build, test, and deploy applications in the cloud at any scale.
- Artifact Registry stores, manages, scans, and secures your container images and packages.
- Google Cloud Deploy is a declarative continuous delivery system.
- Finally, after your code is in production in a platform of your choice, you can use Cloud Logging and Cloud Monitoring for observing your applications.

Let's look closer at the CI/CD components.

Cloud Deploy

Cloud Deploy is a fully managed continuous delivery platform that makes it easy to deploy releases to **GKE**, **Cloud Run**, and **Anthos** in a streamlined and controlled way.

It is tightly integrated with Google Cloud and can be integrated with popular **DevOps tools**.

Google Cloud offers a complete, fully managed, serverless, DevOps pipeline with pluggable systems that can be replaced by any existing third-party tooling. Let's look at the individual components:

- Cloud Code is a plugin to IDEs like VS Code and IntelliJ that allows you to write, debug, and deploy your Kubernetes applications. It also has pre-configured samples so you can start running them in seconds.
- Cloud Source Repositories is a fully managed Git service that allows you to store your code and configure triggers to your CI/CD pipelines.
- Cloud Build is a completely serverless CICD platform. Build, test, and deploy applications in the cloud at any scale.
- Artifact Registry stores, manages, scans, and secures your container images and packages.
- Google Cloud Deploy is a declarative continuous delivery system.
- Finally, after your code is in production in a platform of your choice, you can use Cloud Logging and Cloud Monitoring for observing your applications.

Let's look closer at the CI/CD components.

Cloud Deploy

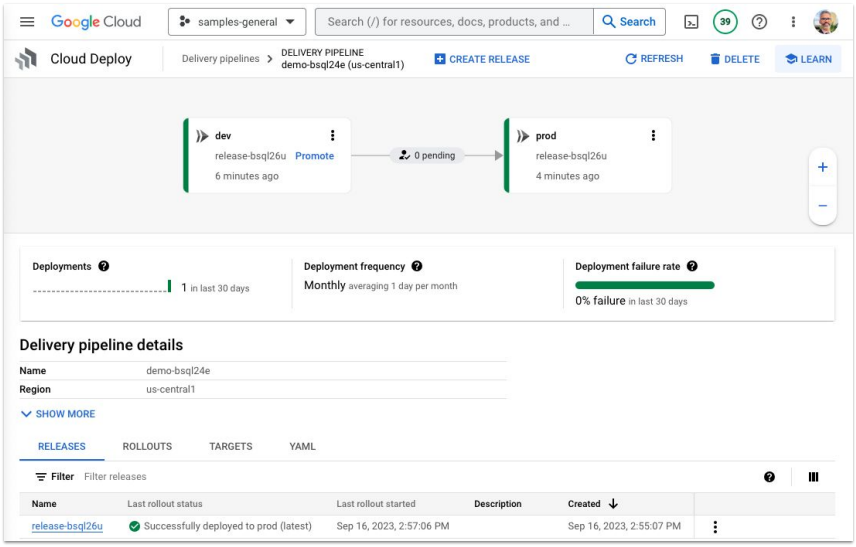
- **Streamlined continuous delivery:** Makes it easy to define and progress releases through environments such as test, stage, and production.
- **Fully managed:** You don't have to worry about setting up or managing infrastructure.
- **Single pane of glass:** A single pane of glass to monitor and control release candidates organization-wide.
- **Tightly integrated with Google Cloud:** You can lockdown release progression via IAM, monitor release events with Cloud Logging, and achieve traceability with Cloud Audit Logs.
- **Integrates with the tools you love:** Can be integrated with popular DevOps tools such as CI and ticketing.

Google Cloud offers a complete, fully managed, serverless, DevOps pipeline with pluggable systems that can be replaced by any existing third-party tooling. Let's look at the individual components:

- Cloud Code is a plugin to IDEs like VS Code and IntelliJ that allows you to write, debug, and deploy your Kubernetes applications. It also has pre-configured samples so you can start running them in seconds.
- Cloud Source Repositories is a fully managed Git service that allows you to store your code and configure triggers to your CI/CD pipelines.
- Cloud Build is a completely serverless CICD platform. Build, test, and deploy applications in the cloud at any scale.
- Artifact Registry stores, manages, scans, and secures your container images and packages.
- Google Cloud Deploy is a declarative continuous delivery system.
- Finally, after your code is in production in a platform of your choice, you can use Cloud Logging and Cloud Monitoring for observing your applications.

Let's look closer at the CI/CD components.

Cloud Deploy

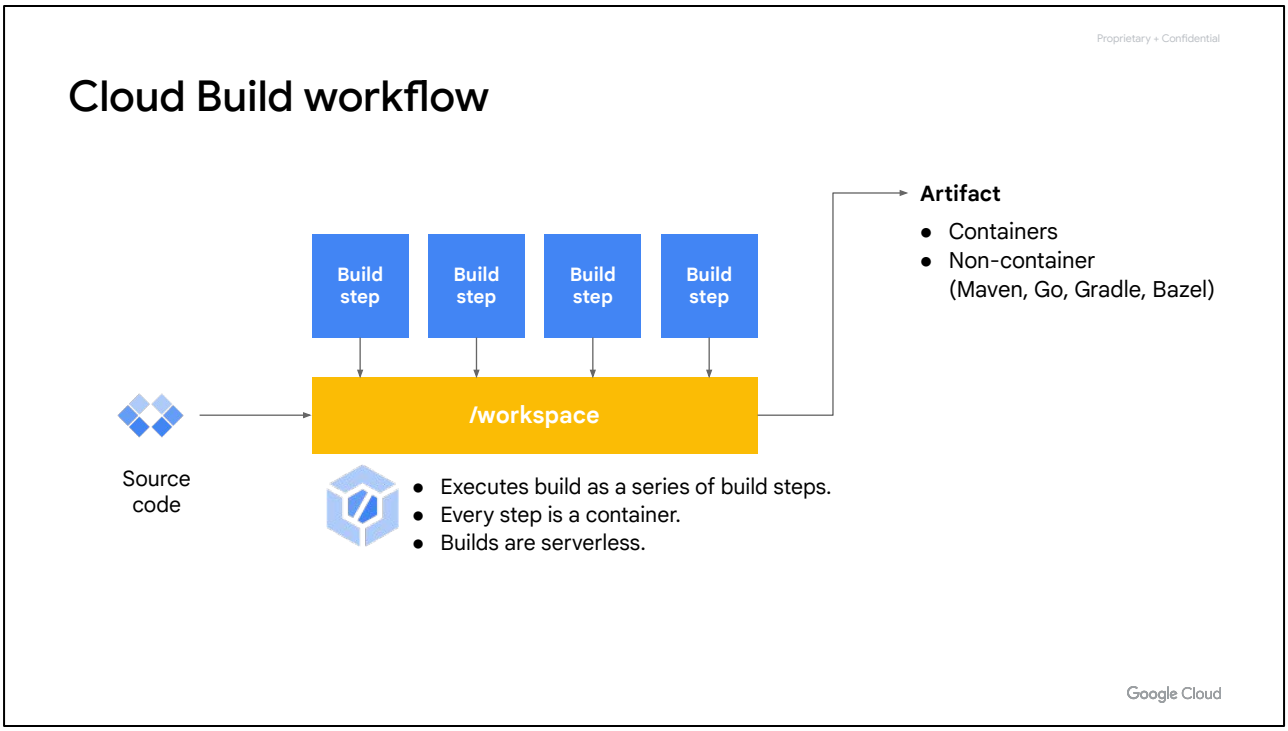


Google Cloud

Google Cloud offers a complete, fully managed, serverless, DevOps pipeline with pluggable systems that can be replaced by any existing third-party tooling. Let's look at the individual components:

- Cloud Code is a plugin to IDEs like VS Code and IntelliJ that allows you to write, debug, and deploy your Kubernetes applications. It also has pre-configured samples so you can start running them in seconds.
- Cloud Source Repositories is a fully managed Git service that allows you to store your code and configure triggers to your CI/CD pipelines.
- Cloud Build is a completely serverless CICD platform. Build, test, and deploy applications in the cloud at any scale.
- Artifact Registry stores, manages, scans, and secures your container images and packages.
- Google Cloud Deploy is a declarative continuous delivery system.
- Finally, after your code is in production in a platform of your choice, you can use Cloud Logging and Cloud Monitoring for observing your applications.

Let's look closer at the CI/CD components.



Cloud Build
https://cloud.google.com/cloud-build

Documentation
https://cloud.google.com/build/docs

Cloud Build is a service that executes your builds on Google Cloud Platform infrastructure. Cloud Build can import source code from Cloud Storage, Cloud Source Repositories, GitHub, or Bitbucket, execute a build to your specifications, and produce artifacts such as Docker containers or Java archives.

Cloud Build executes your build as a series of *build steps*, where each build step is run in a Docker container. A build step can do anything that can be done from a container irrespective of the environment. To perform your tasks, you can either use the supported build steps provided by Cloud Build or write your own build steps.

- Supported Build Steps → https://cloud.google.com/build/docs/deploying-builds/deploy-cloud-run
- Write your own build steps → https://cloud.google.com/build/docs/configuring-builds/use-community-and-custom-builders

cloudbuild.yaml file specifies the build steps

A build config file defines the fields that are needed for Cloud Build to perform your tasks.

You can write the build config file using the YAML or the JSON syntax

```
steps:
- name: 'gcr.io/cloud-builders/npm'
  args: ['install']
- name: 'gcr.io/cloud-builders/npm'
  args: ['test']
- name: 'gcr.io/cloud-builders/docker'
  args: ['build', '-t', 'us-central1-docker.pkg.dev/$PROJECT_ID/nodejs/helloworld', '.']
- name: 'gcr.io/cloud-builders/docker'
  args: ['push', 'us-central1-docker.pkg.dev/$PROJECT_ID/nodejs/helloworld']
```


The Cloud Build steps are defined in a configuration file, which takes the name of cloudbuild.yaml file by default. Here, you can see that we are using an NPM container builder to install and test the application and a Docker container builder to build the application container and push it to Artifact Registry.

Cloud Build - Deploy to Cloud Run example

Can trigger Cloud Build automatically via a push to a repository or run it manually `gcloud builds submit...`

```
steps:
# Build the docker image and tag it
- name: 'gcr.io/cloud-builders/docker'
  args: ['build', '-t', 'us-central1-docker.pkg.dev/$PROJECT_ID/nodejs/helloworld:latest', '.']

# Push the container image to Artifact Registry
- name: 'gcr.io/cloud-builders/docker'
  args: ['push', 'us-central1-docker.pkg.dev/$PROJECT_ID/nodejs/helloworld:latest']

# Deploy container image to Cloud Run
- name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'
  entrypoint: 'gcloud'
  args: ['run', 'deploy', 'space-invaders-from-cloud-build', '--image',
'us-central1-docker.pkg.dev/$PROJECT_ID/nodejs/helloworld:latest', '--region', 'us-central1']

# Name of the image in the Artifact Registry
images:
- 'us-central1-docker.pkg.dev/$PROJECT_ID/nodejs/helloworld:latest'
```

Cloud Build has multiple "builders": docker, go, kubectl, maven (mvn), Node.js (npm), cloud-sdk, git, and more*

Next topic

Using the cloud-sdk builder

*Complete list of Google Cloud Build images:
https://cloud.google.com/build/docs/cloud-builders

Google Cloud

Cloud Build - Create a build configuration file
https://cloud.google.com/build/docs/configuring-builds/create-basic-configuration

There are many good sections to review in this site - look at the different topics on the left.

Cloud builders

Containers with common languages and tools that you can use in build steps:

docker	gcr.io/cloud-builders/docker	docker example
go	gcr.io/cloud-builders/go	go example
gcloud	gcr.io/cloud-builders/gcloud	gcloud example
gradle	gcr.io/cloud-builders/gradle	gradle example
maven	gcr.io/cloud-builders/mvn	maven example
kubectl	gcr.io/cloud-builders/kubectl	kubectl example
npm	gcr.io/cloud-builders/npm	npm example

And many more: https://github.com/GoogleCloudPlatform/cloud-builders

The containers that build, test, and deploy your application are pre-built and curated by Google and can be used without modification in your pipeline. Additionally, you can use any container as a build container in Cloud Build.

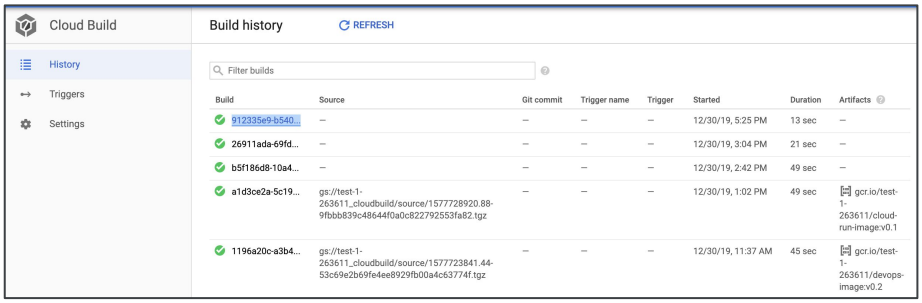
Four common types of testing

Type of Testing	Purpose	Example	Tools
Unit Testing	To verify if individual units of code work correctly Does not test external dependencies	Testing a function that adds two numbers	Cloud Build
Integration Testing	To verify if different units of the software work well together Tests external dependencies	Testing how a database communicates with an application	Google Application Integration
Performance Testing	To verify if the software meets the performance requirements	Testing how fast the application can handle a certain number of requests	Cloud Trace, Apache Benchmark, New Relic, ...
Load Testing	To verify if the software can handle a high volume of traffic	Testing how the application responds to a sudden surge in traffic	Apache Benchmark, Selenium, ...

Cloud Build lets you build software quickly across all languages

- Google-hosted Docker build service
  - Alternative to using Docker build command
- Use the CLI to submit a build

```
$ gcloud builds submit --tag us-central1-docker.pkg.dev/$PROJECT_ID/nodejs/helloworld
```



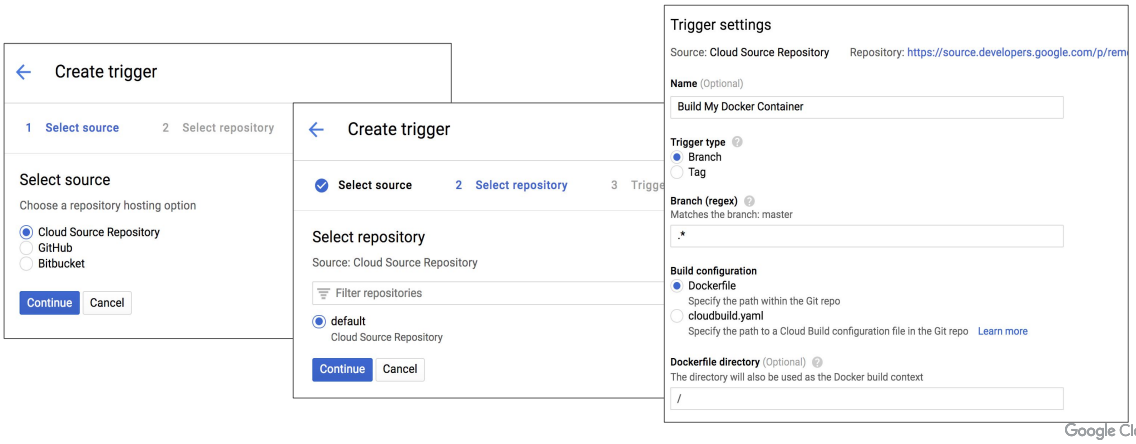
Os desenvolvedores ganham controle total sobre a definição de fluxos de trabalho para criar, testar e implantar em vários ambientes, incluindo VMs, sem servidor e Kubernetes. Não há mais necessidade de provisionar ou manter ambientes de construção: tudo é feito pelo Cloud Build. Você escreve uma configuração de compilação para fornecer instruções ao Cloud Build sobre quais tarefas executar. Eles são definidos como uma série de etapas. Cada etapa é executada por um construtor de nuvem. Os construtores de nuvem são contêineres com linguagens e ferramentas comuns instaladas neles.
Os construtores podem ser configurados para buscar dependências, executar testes de unidade, análises estáticas e testes de integração e criar artefatos com ferramentas de construção, como docker, gradle, maven, bazel e gulp. O Cloud Build executa as etapas de criação que você define. Executar etapas de construção é semelhante a executar comandos em um script.

Você pode usar as etapas de compilação fornecidas pelo Cloud Build e a comunidade Cloud Build ou escrever suas próprias etapas de compilação personalizadas:

Os construtores disponíveis podem ser encontrados aqui:
https://github.com/GoogleCloudPlatform/cloud-builders

Build triggers watch a repository and build a container whenever code is pushed

Supports Maven, custom builds, and Docker



A Cloud Build trigger automatically starts a build whenever a change is made to source code. It can be set to start a build on commits to a particular branch or on commits that contain a particular tag.You can specify a regular expression with the branch or tag value to match. The syntax for the regular expression is at https://github.com/google/re2/wiki/Syntax.

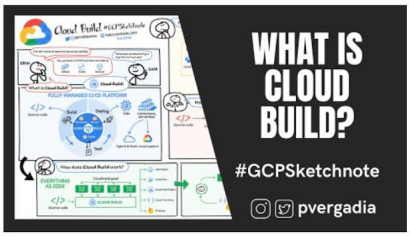
The build configuration can be specified either in a Dockerfile or a Cloud Build file. The configuration required is shown in the slide.

Guide: Cloud Build and Artifact Registry

- Cloud Build
- CI/CD
- Artifact Registry
- Serverless Deployment
- Manage containers
- Trigger Cloud Build from Cloud Source Repositories

What is Cloud Build

A quick overview of Cloud Build, a serverless CI/CD tool that helps build, test and deploy code at scale.



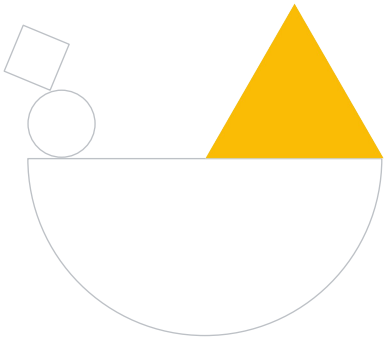
What is Cloud Build? #GCPSketchnote

CI/CD Google Cloud using Cloud Build

Understand CI/CD to automate with Cloud Build.

Cloud Build and Artifact Registry

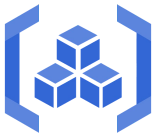
Artifact Registry



Artifact Registry

Stores multiple artifact formats

- Docker images
- OS packages for Linux distributions
- Language packages for Python, Java, and Node



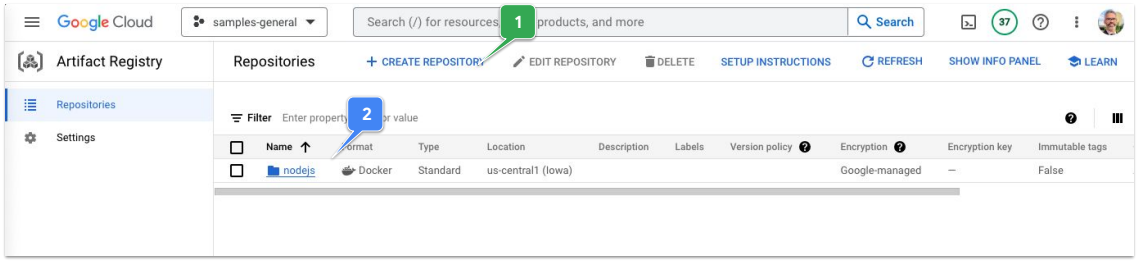
Provides

- Built-in vulnerability scanning of container images
- Controlled access with fine-grained control (IAM)
- Regional and multi-regional repositories
- Can have multiple per project

https://cloud.google.com/artifact-registry

Artifact Registry has its own IAM roles and can be deployed multi-regionally.

Artifact Registry



Artifact Registry

1. Create Repository
2. List of repositories

To start working with Artifact Registry, you first need to create a repository within Artifact Registry and then interact with it.

Each repository is associated with a specific artifact format. For example, a Python repository stores Python packages.

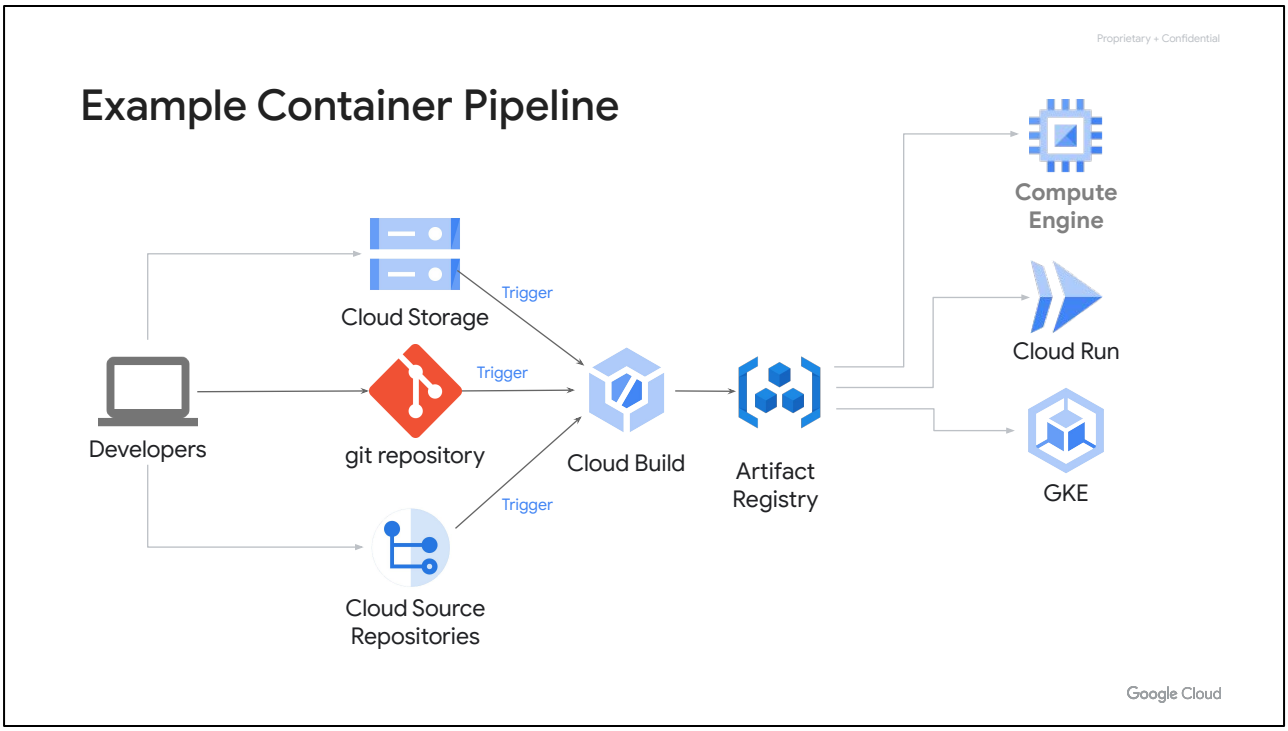
Artifact Registry has its own IAM roles and can be deployed multi-regionally.

Cloud Build pushes a container to Artifact Registry

```
steps:
- name: 'gcr.io/cloud-builders/npm'
  args: ['install']
- name: 'gcr.io/cloud-builders/npm'
  args: ['test']
- name: 'gcr.io/cloud-builders/docker'
  args: ['build', '-t', 'us-central1-docker.pkg.dev/$PROJECT_ID/nodejs/helloworld', '.']
- name: 'gcr.io/cloud-builders/docker'
  args: ['push', 'us-central1-docker.pkg.dev/$PROJECT_ID/nodejs/helloworld']
```

Artifact Registry securely stores and analyzes your containers.

The last step of this continuous integration pipeline is pushing the container to Artifact Registry, which securely stores and analyzes your containers.



Continuous integration (CI)
https://cloud.google.com/solutions/continuous-integration

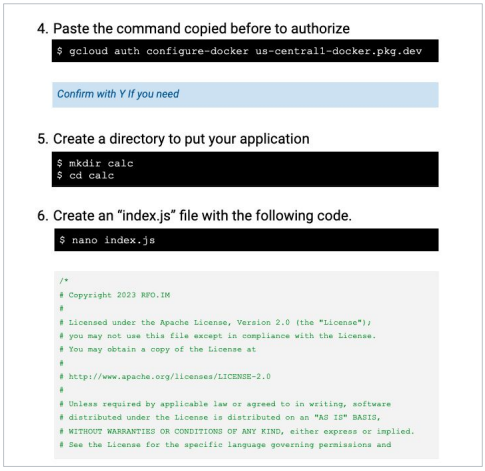
Building an automated serverless deployment pipeline with Cloud Build
https://cloud.google.com/blog/topics/developers-practitioners/building-automated-serverless-deployment-pipeline-cloud-build

Cloud Build can retrieve the source code for your builds from a variety of storage locations: Cloud Source Repositories, Cloud Storage, or git-compatible repositories like GitHub and Bitbucket.

To generate a build with Cloud Build, you define a series of steps. For example, you can configure build steps to fetch dependencies, compile source code, run integration tests, or use tools such as Docker, Gradle, and Maven. Each build step in Cloud Build runs in a Docker container.

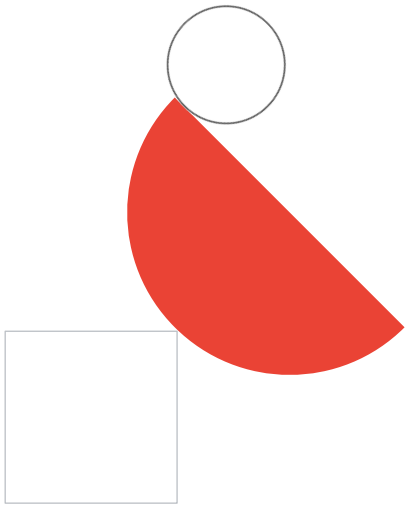
Demo: Cloud Build - NodeJs build and test

- In this demonstration, you will create an application and perform unit tests on it and automate the creation of the application container, build and testing it and send it to the Artifact Registry .



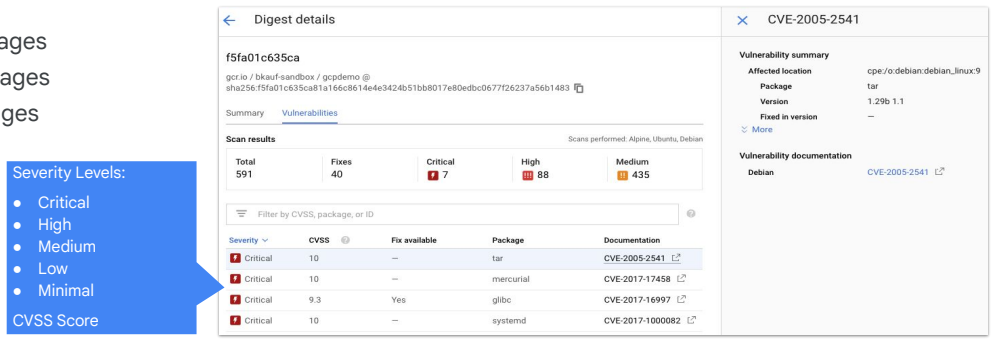
Cloud Build - NodeJs build and test

Binary Authentication



Artifact Registry provides container analyses and vulnerability scanning

- Scans images and installed packages for Common Vulnerabilities and Exposures (CVEs) when:
  - An image is added to the registry.
  - There is an update to the database.
- Works for:
  - Debian images
  - Ubuntu images
  - Alpine images



The Container Analysis API allows you to store metadata information associated with a resource. This metadata can be later retrieved to audit your software supply chain.

Container Analysis is built on top of Grafeas, an open source component metadata API that can work as a centralized authority for tracking and enforcing policies. Build, auditing, and compliance tools can use Grafeas to store, query, and retrieve comprehensive metadata about software components.

Because Grafeas is open source, you are not committed to a specific vendor. Grafeas uses a unique software identifier to associate metadata. It decouples the artifact storage so you can store metadata about components from many different repositories. The same principles apply to Container Analysis: you can use it as a centralized universal metadata store for software components in Artifact Registry or any other location.

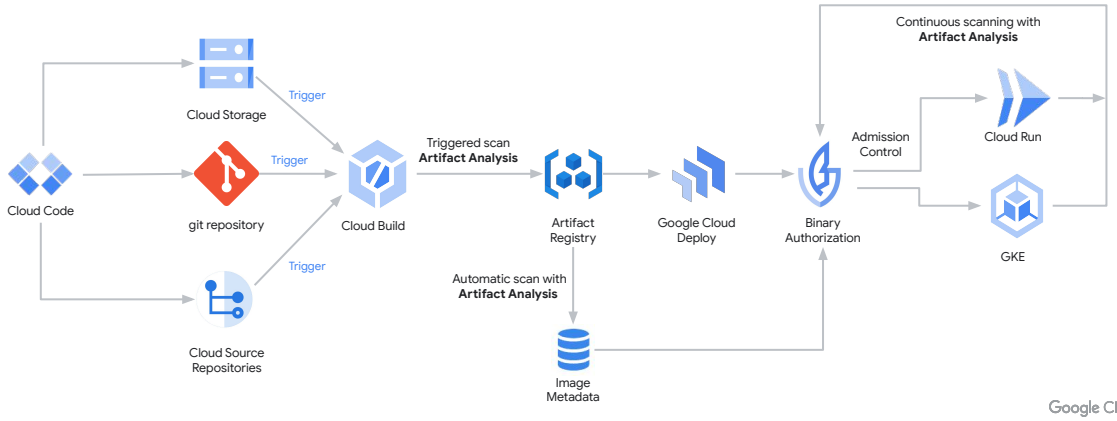
When an image is stored in Artifact Registry, the registry automatically scans it for known Common Vulnerabilities and Exposures, or CVEs. Artifact Registry examines images and packages installed in images and works for Debian, Ubuntu, and Alpine images.

Images are scanned when:

- They are added to the registry.
- The vulnerability database is updated.

Container pipeline with Binary Authorization

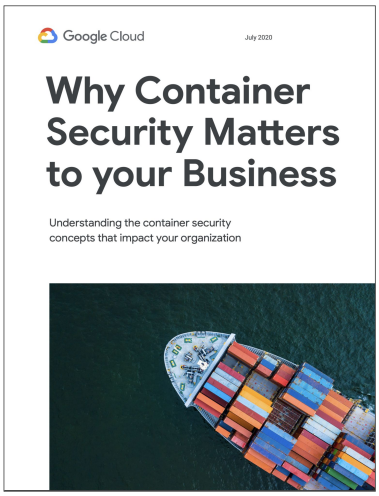
Artifact Analysis (family of services that provide software composition analysis, metadata storage and retrieval) creating and interacting with metadata across source, build, storage, deployment and runtime environments, working with Binary Authorization.



Container Security

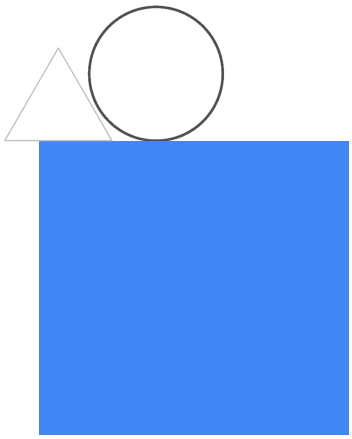
Shared Responsibility Model

- The security of cloud services is a shared responsibility between the Google and the user
- Security and incident response teams need to understand what they are responsible for hardening and protecting vs Google's responsibility

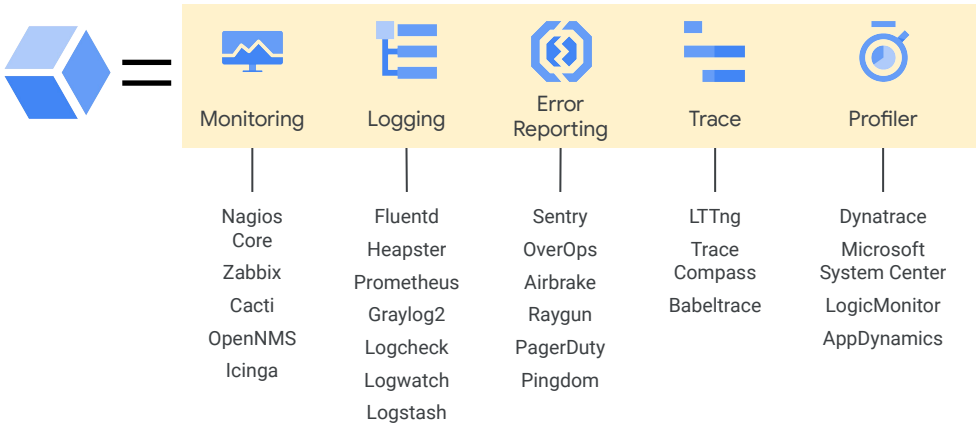


https://services.google.com/fh/files/misc/why_container_security_matters_to_your_business.pdf

Operations

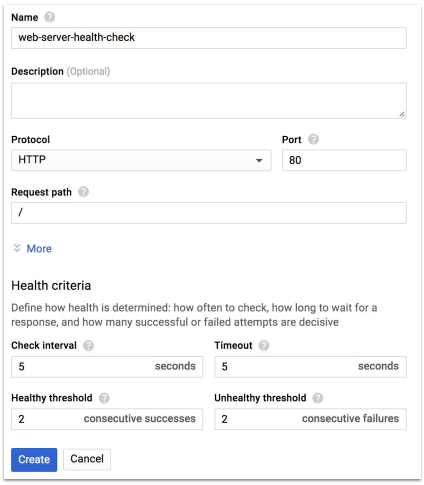


Google Cloud's operations suite unifies operations tools



Health Checks

- Used to ensure machines are ready to take requests
  - Instance groups will create new machines if existing ones aren't healthy
  - Load balancers use health checks to determine if they can send requests to machines



When you set up a load balancer, you need a health check to ensure the load balancer doesn't send requests to unhealthy instances.

If you are creating an instance group, you need a health check to set up the auto healing feature.

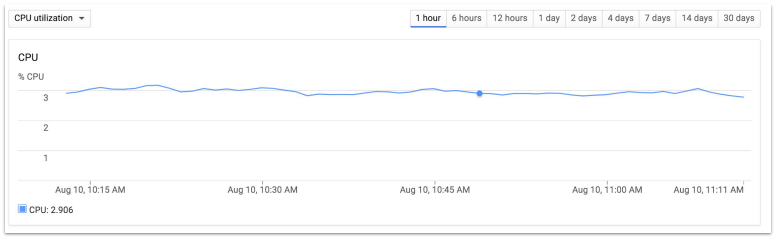
Health checks are easy. Just make a request to the service and see if it works. Some of the parameter values may not be entirely obvious though. It's possible for a single request to fail on a healthy machine. The opposite might be true too. One request might succeed once, but the machine might not really be healthy. That's what the Healthy and Unhealthy threshold values are for. You might set these values a bit higher than 1 just to be sure.

The check interval determines how often requests are made. Be aware, there is more than one server on Google's side than runs the health checks, so there are actually more health checks than what you specify in the interval.

One other thing to be aware of, if your service is not public and the health checks never seem to run, it's likely you need to create a firewall rule to open your service ports to the health checkers. See the documentation for the IP address range of the health check servers.

Resource Monitoring

- Operations can be used to monitor many different resources
  - Cloud Run, App Engine, virtual machines, and many more
  - In addition to Google Cloud resources, AWS resources can be monitored as well
- Different metrics can be monitored depending on the resource
  - CPU usage, network traffic, uptime, message count, etc.
  - Custom metrics can be created



If you're creating virtual machines you are paying for CPUs, memory, and disk space. You should monitor those things to make sure you aren't paying for more resources than you need. This will optimize your spend.

There are lots of other things you may want to monitor. What are you spending on your App Engine application? Maybe you are allowing users to upload pictures in a Cloud Storage bucket. You might want to know how much data is in the bucket.

You're paying for network traffic out. Maybe you want to keep an eye on that and turn on the CDN if you get too much traffic. You might also want to know where your requests are coming from, and deploy your servers closer to users.

Operations has many built-in metrics that you can monitor for most every service. You can also use Operations to monitor AWS resources.

You can even create your own custom metrics programmatically. Thus, what you can monitor is really only limited by your imagination. For example, if you were an online game developer, you could create a custom metric to monitor the number of game users and the number of moves.

Monitoring Agent

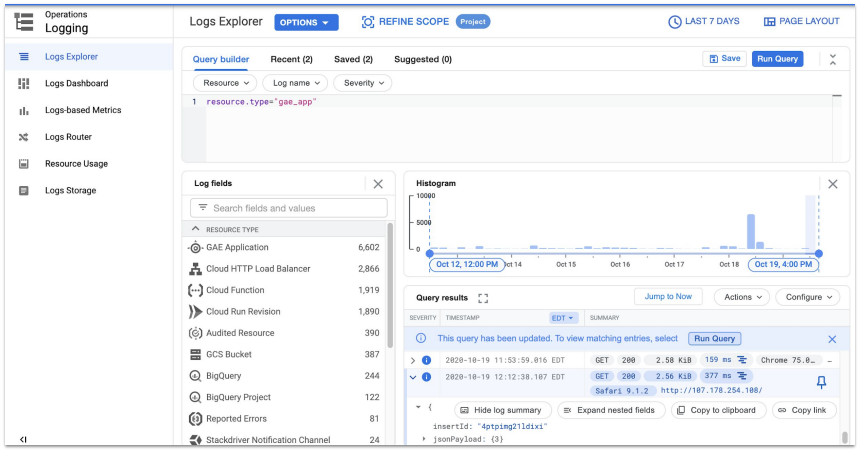
- Installed on VMs to provide additional metrics not available externally
  - Memory usage and uptime, for example
- Also provides metrics on common applications
  - Apache, Cassandra, CouchDB, HBase, IIS, JVM, Kafka, etc.
- See the docs: https://cloud.google.com/monitoring/agent/

Sometimes Operations needs a little help from inside a machine to monitor certain metrics. Memory usage might be the most common example of this. To solve this issue, you just install the Operations monitoring agent on the virtual machine you want to monitor.

The monitoring agent will also detect common applications on your machines and allow you to monitor those as well. Applications that are automatically supported include Apache Web Server, Cassandra, Internet Information Services, and many more.

See cloud.google.com/monitoring/agent for more information.

Logs



Every Google Cloud request is logged automatically. You can also write messages from your applications to the logs. Like the monitoring agent, there is a Operations logging agent you can install on virtual machines to make this easy.

You can search and filter the logging information in the UI.

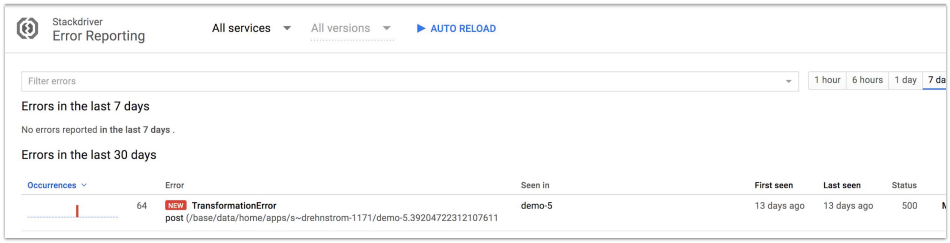
Log Exports

- Logs can be exported for further analysis
- Export destinations include:
  - BigQuery
  - Cloud Storage
  - Pub/Sub
- Can create advanced filters to specify which log entries to export
- Set a destination to Pub/Sub to immediately respond to a log export

You can also set up log exports. Log data can be exported to Cloud Storage if you want to archive it. Or, the logs can be exported to BigQuery if you want to write SQL queries to analyze the logs. You can also export log data into Pub/Sub and get real time log analysis if you like.

Error Reporting

- Automatically set up with App Engine services
  - Can enable for services running in Compute Engine
  - Can integrate with Operations logs
- Can enable automatic notification for errors



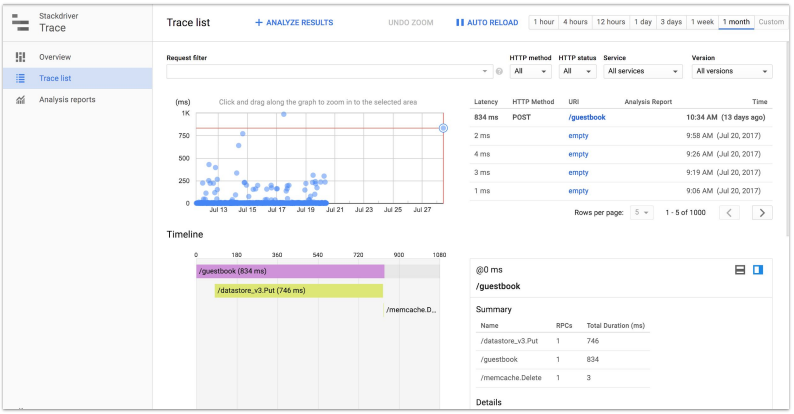
The Google SRE team developed a separate Error Reporting service to make it easier to monitor your applications for errors. This just works if you are deploying your applications using App Engine.

You can also set up notifications so you are aware if your applications start generating errors.

Trace

Displays requests along with their timings

- Useful for debugging performance problems



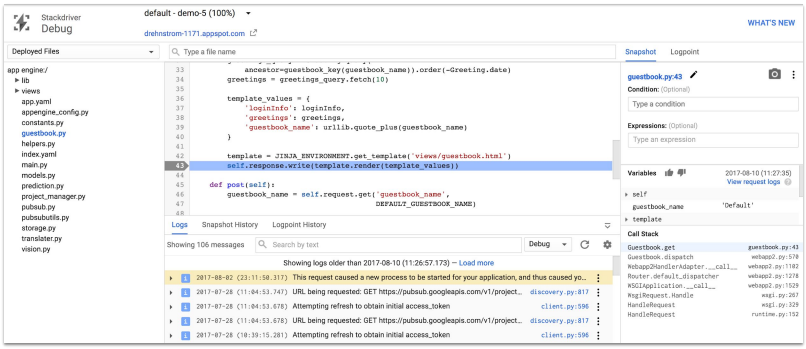
If you are a programmer, you might be trying to debug a performance problem and sometimes this can be difficult. It may not be obvious what is specifically taking a long time within a single service request.

The Trace service makes this easier. Whenever a request is made to an App Engine application, the request is added to the Trace. The Trace breaks down exactly what happened within the request and how long each piece took.

Debugging

Log points can be added to code of deployed applications

- Creates a snapshot of the running app when a breakpoint is hit



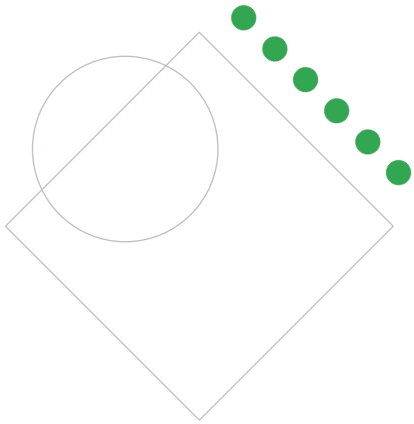
Google Cloud

Sometimes an application appears to work. It runs on your developer machine, it runs in the test environment, and it passes all its units tests. So, you deploy it to the cloud where it stops working. Frustrating! The Operations Debugger can help.

You can set what is called a log point in the source code and when that line of code is hit the debugger will create a snapshot of the application at that moment.

The programmer can then go back and look at values of variables and things and hopefully this will give him/her insight into why the application is failing.

Next workshop's assigned content

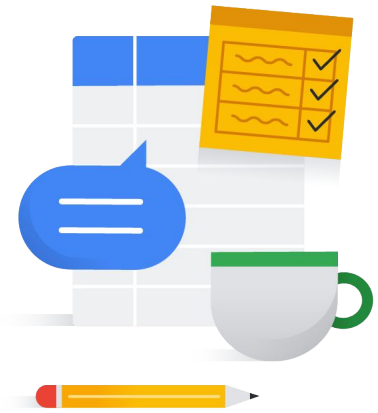


Assignments for the next workshop

-
  - Skills Boost:
    -
- Guide assignments
  - ???
  - etc.

Program issues or concerns?

- Problems with **accessing** Cloud Skills Boost for Partners
  - cloud-partner-training@google.com
- Problems with **a lab** (locked out, etc.)
  - support@qwiklabs.com
- Problems with accessing Partner Advantage
  - https://support.google.com/googlecloud/topic/9198654




Google Cloud