Execution Environments for Your Application

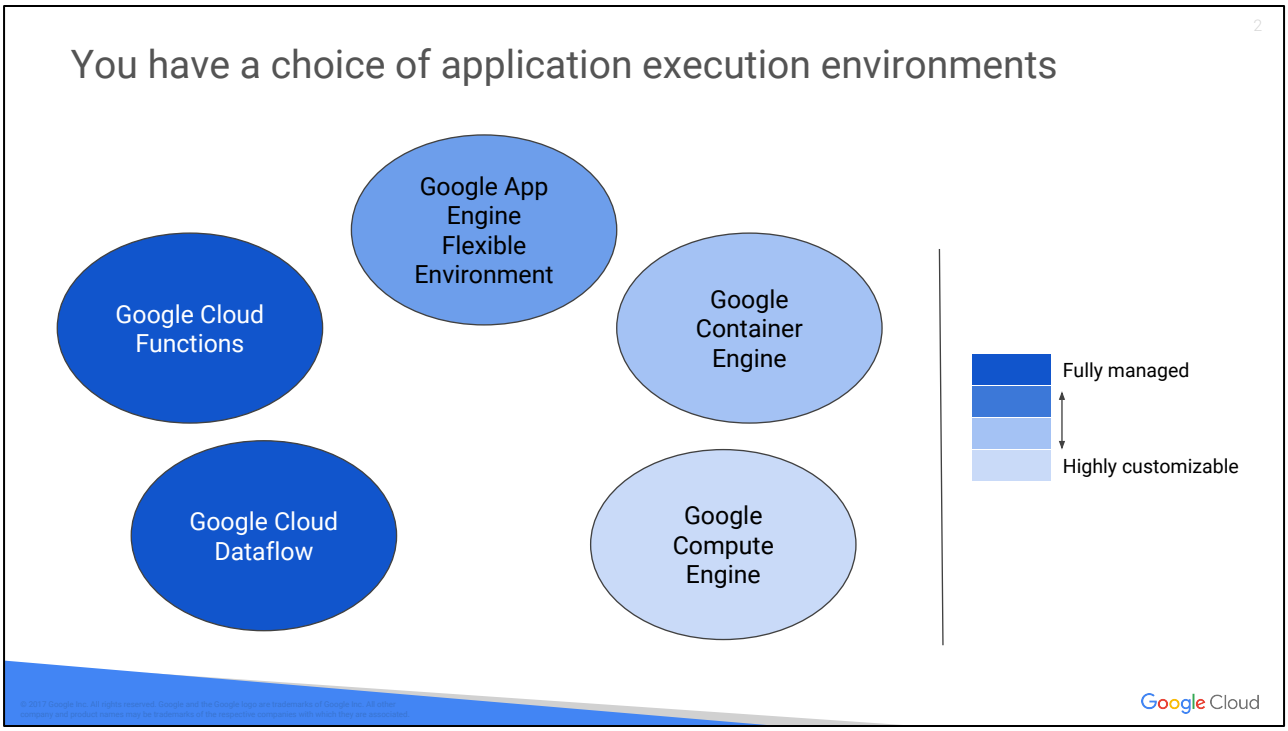
Developing Applications with Google Cloud Platform

COMPUTE ENGINE, CONTAINER ENGINE, APP ENGINE FLEXIBLE ENVIRONMENT, CLOUD FUNCTIONS, CLOUD DATAFLOW

DEPLOYING THE APPLICATION INTO APP ENGINE FLEXIBLE ENVIRONMENT

Version 2.0
Last modified: 2017-09-25

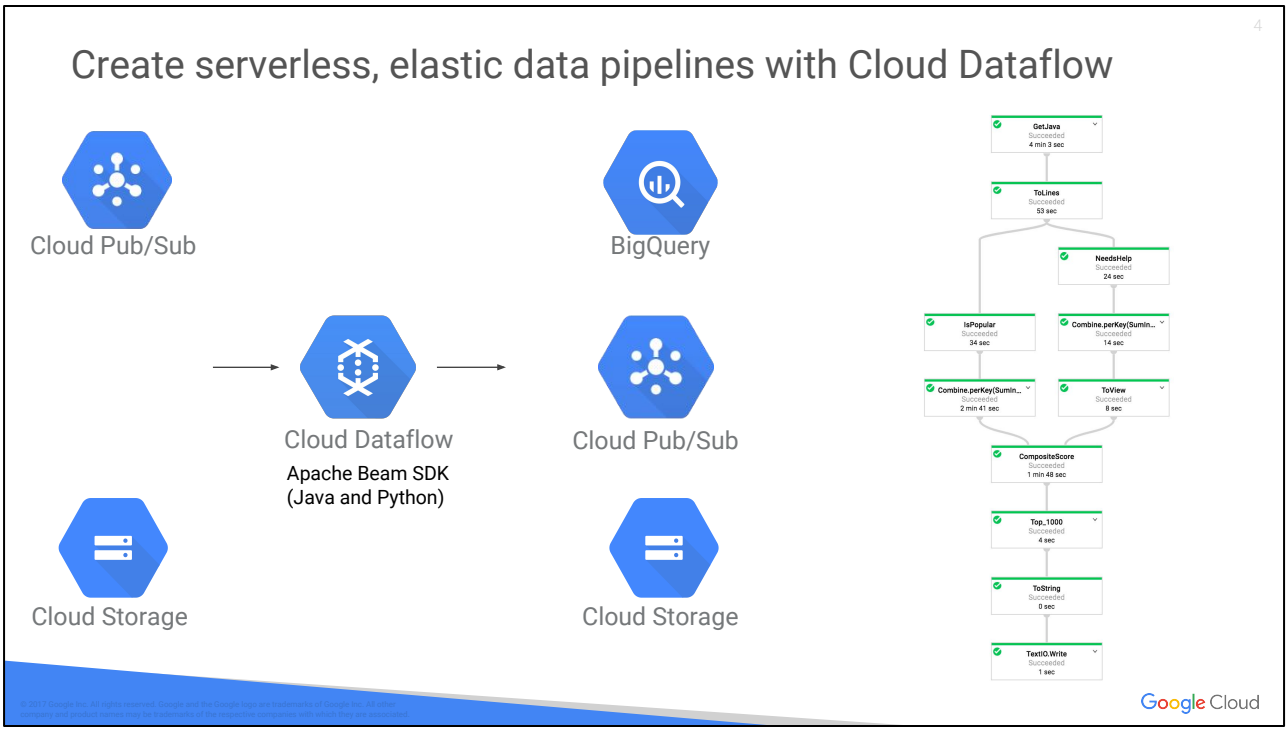
60 minutes



Google Cloud Platform provides a range of options to run your application code. These options range from services like Cloud Functions and Cloud Dataflow, which are fully managed, to App Engine, Container Engine, and Compute Engine, which offer steadily increasing abilities to customize your execution environment. Fully managed compute environments require minimal setup and operations. Highly customizable environments require greater operational and management effort to keep the application running optimally.

Cloud Dataflow

Google Cloud



Cloud Dataflow is a serverless execution engine or runner for executing parallel data processing pipelines that are developed using Apache Beam SDKs.

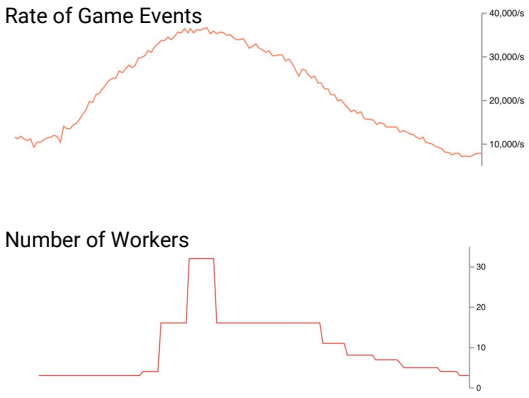
Cloud Dataflow supports fast, simplified pipeline development by using expressive Java and Python APIs in the Apache Beam SDK. Cloud Dataflow integrates with Google Cloud Platform services for streaming events ingestion (Cloud Pub/Sub), data warehousing (BigQuery), machine learning (Cloud Machine Learning), and more.

Apache Beam supports Java and Python SDKs to develop pipelines. A pipeline comprises your entire data processing task including reading input data, transforming the data, and writing output data. The Cloud Dataflow service uses other managed services such as Compute Engine, Stackdriver Logging, Google Cloud Storage, Google Cloud Storage JSON, BigQuery, Google Cloud Pub/Sub, and Google Cloud Datastore APIs to execute your pipelines.

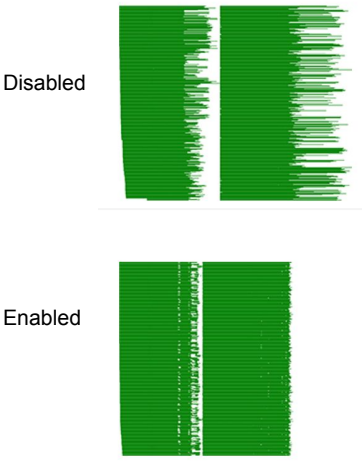
For more information, see:
Cloud Dataflow: https://cloud.google.com/dataflow/
Apache Beam: https://beam.apache.org/documentation/

Cloud Dataflow removes operational overhead

Autoscaling



Dynamic Workload Rebalancing

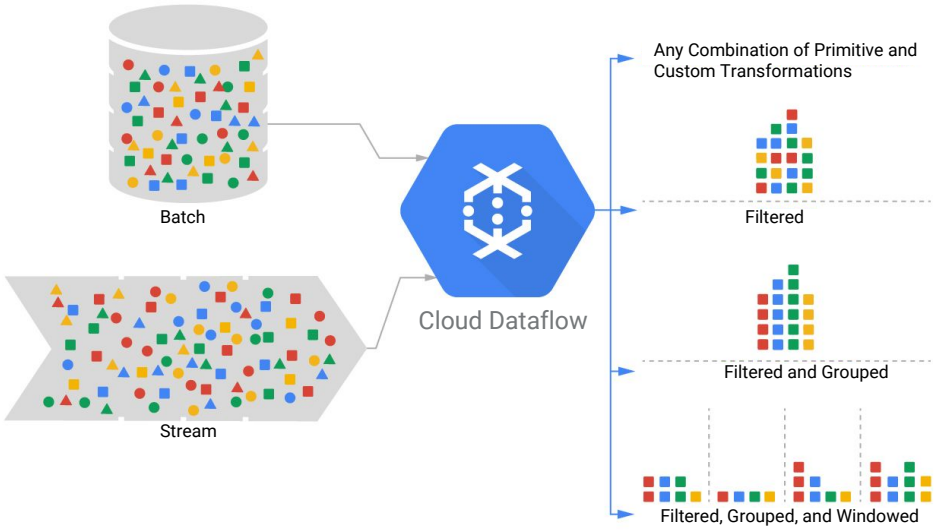


Google Cloud

With systems such as Spark and Hadoop, users spend a significant amount of time tuning their jobs. They have to estimate the number of workers to use for their job/cluster. A single configuration might not be sufficient because resource needs can dynamically change over the lifetime of the job. With Cloud Dataflow, you don't need to specify worker counts. Cloud Dataflow supports simple configuration and optimizes the worker count over time. Cloud Dataflow autoscales based on metrics such as CPU utilization, throughput, and the amount of work remaining (or backlog). It adds workers when CPU utilization and backlog increase and removes them when these metrics decrease.
For more information about autoscaling in Cloud Dataflow, see https://cloud.google.com/blog/big-data/2016/03/comparing-cloud-dataflow-autoscaling-to-spark-and-hadoop.

Dynamic workload rebalancing reduces overall completion time, reduces cost by keeping workers busy, and offers better predictability of overall job completion time.
For more information about dynamic workload rebalancing, see https://cloud.google.com/blog/big-data/2016/05/no-shard-left-behind-dynamic-work-rebalancing-in-google-cloud-dataflow.

Use Cloud Dataflow for high-volume data processing



Google Cloud

Cloud Dataflow is ideal for streaming and batch data pipelines. It enables use cases such as the following:

- Clickstream, point-of-sale, and segmentation analysis in the retail industry
- Fraud detection in the financial services industry
- Personalized user experience in the gaming industry
- IoT analytics in manufacturing, healthcare, and logistics industries

Consider other compute environments

Dataflow SDK 2.x for Java and Python

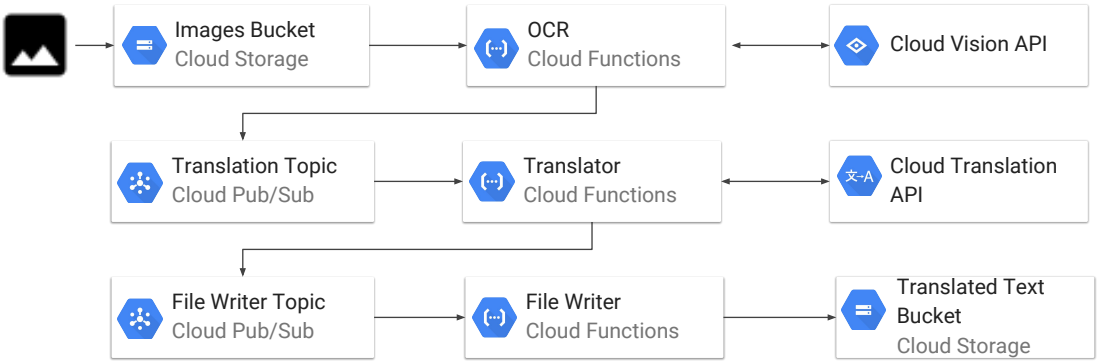
Google Cloud

The Dataflow SDK 2.x, based on Apache Beam SDK, is available for Java and Python programming languages. Consider other compute environments if you need to execute code in other programming languages.

Cloud Functions

Google Cloud

Develop event-driven, serverless, highly scalable microservices with Cloud Functions



Google Cloud

With Cloud Functions, you can develop an application that is event-driven, serverless, and highly scalable. Each function is a lightweight microservice that enables you to integrate application components and data sources. Cloud Functions are ideal for microservices that require a small piece of code to quickly process data related to an event. Cloud Functions are priced according to how long your function runs, the number of times it is invoked, and the resources that you provision for the function.

Consider an application that allows users to upload images that contain text to a bucket in Google Cloud Storage. The OCR Cloud Function is triggered when a new image is uploaded to the Images bucket. This function uses the Google Cloud Vision API to extract text from images and then queues up the text in Cloud Pub/Sub for translation. The Translator Cloud Function, triggered by the Cloud Pub/Sub topic, invokes the Google Cloud Translation API to translate the text. It queues up the translated text in the File Writer topic in Cloud Pub/Sub. The File Writer Cloud Function writes the translated text for each image as separate files in Cloud Storage.

In this example, all components of the application use fully managed services and APIs such as Cloud Storage, Cloud Pub/Sub, Cloud Functions, Cloud Vision API, and Cloud Translation API. These services scale automatically depending on the volume of incoming data and required compute resources. This scalability and reliability enables you to focus on your application code.

For more information about Cloud Functions, see https://cloud.google.com/functions/.

Focus on code: Node.js and Google Cloud Client Libraries

index.js

```
/**
 * Triggered from a message on a Cloud Pub/Sub topic.
 *
 * @param {!Object} event The Cloud Functions event.
 * @param {!Function} The callback function.
 */
exports.subscribe = function subscribe(event, callback) {
  // The Cloud Pub/Sub Message object.
  const pubsubMessage = event.data;

  // We're just going to log the message to prove that
  // it worked.
  console.log(Buffer.from(pubsubMessage.data, 'base64').toString());

  // Don't forget to call the callback.
  callback();
};
```

package.json

```
{
  "name": "sample-pubsub",
  "version": "0.0.1"
}
```

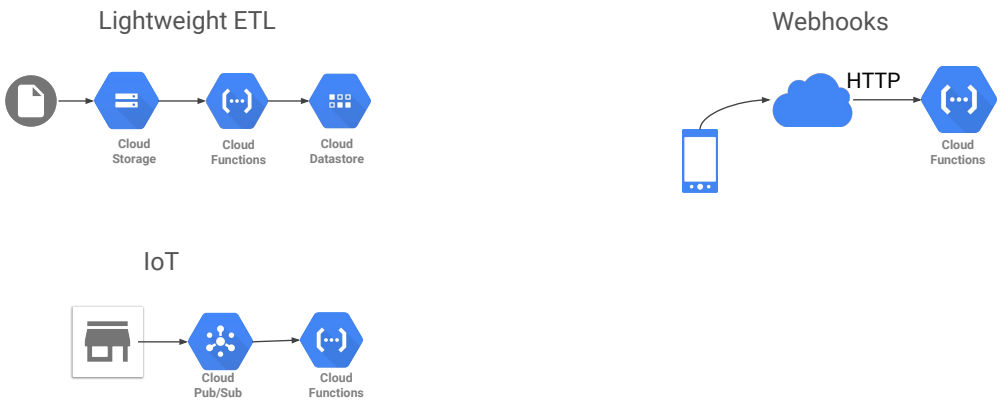

Google Cloud

Cloud Functions currently support a Node.js runtime environment. The function's source code must be exported in a Node.js module.

You do not need to upload zip files with packaged dependencies. You can specify any dependencies in a package.json file.The Cloud Functions service automatically installs all dependencies before running your code.

You can use Google Cloud Client Libraries to programmatically interact with other Google Cloud Platform services.

Use Cloud Functions to enable event-driven processing or to develop lightweight microservices



Google Cloud

Cloud Functions can be used in a variety of use cases that require event-driven processing or lightweight microservices.

You can use Cloud Functions for lightweight extract-transform-load (ETL) operations. For example, when a file is uploaded to Cloud Storage, a Cloud Function can be triggered to transform and upload the contents to a database. You can also use Cloud Functions to process IoT streaming data or other application messages that are published to a Cloud Pub/Sub topic.

Cloud Functions can serve as webhooks. For example, you can set up a webhook that is invoked automatically after each commit to a Git repository. External and internal clients can make direct HTTP calls to invoke microservices that are deployed as Cloud Functions.
For more information about Cloud Functions, see https://cloud.google.com/functions/.

Consider other compute environments

Large or complex codebase

Node.js runtime only

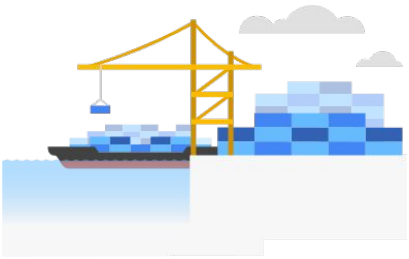
Google Cloud

Consider other compute environments if your application or microservice has a large and complex codebase or if you need to use runtime environments other than Node.js.

App Engine Flexible Environment

Deploy scalable web and mobile backends in any language with App Engine flexible environment

Native runtimes



Custom runtimes

Google Cloud

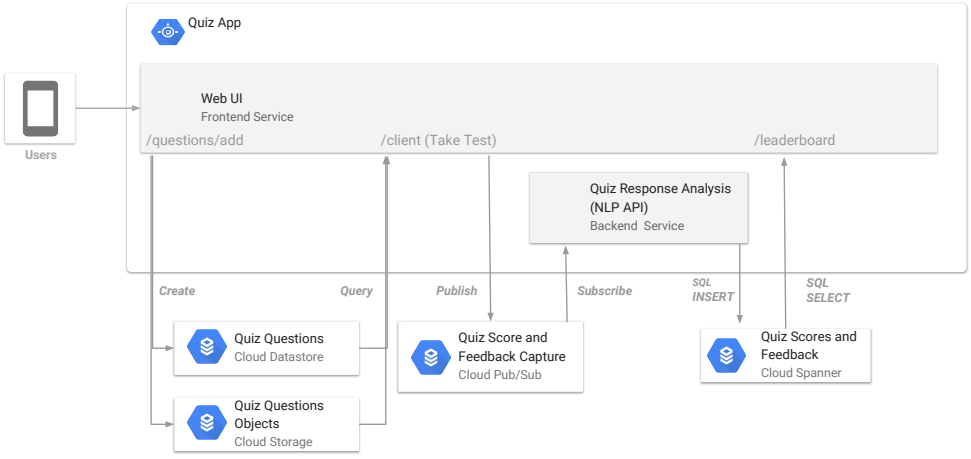
The App Engine flexible environment is an excellent option for deploying web applications, backends for mobile applications, HTTP APIs, and internal business applications.

App Engine flexible environment provides default settings for infrastructure components. You can customize settings such as network, subnetwork, port forwarding, and instance tags. SSH access to VM instances in the flexible environment is disabled by default. You can enable root access to the underlying VM instances. App Engine flexible environment runs a Docker image for your application. If needed, you can generate the Docker image of the application and run it in other container-based environments such as Google Container Engine.

App Engine flexible environment supports Java 8, Python 2.7 and Python 3.5, Node.js, Ruby, PHP, .NET core, Go, and other runtimes based on a custom Docker image or open-source Docker file. You can use Google Cloud Client Libraries to programmatically interact with other Google Cloud Platform services.

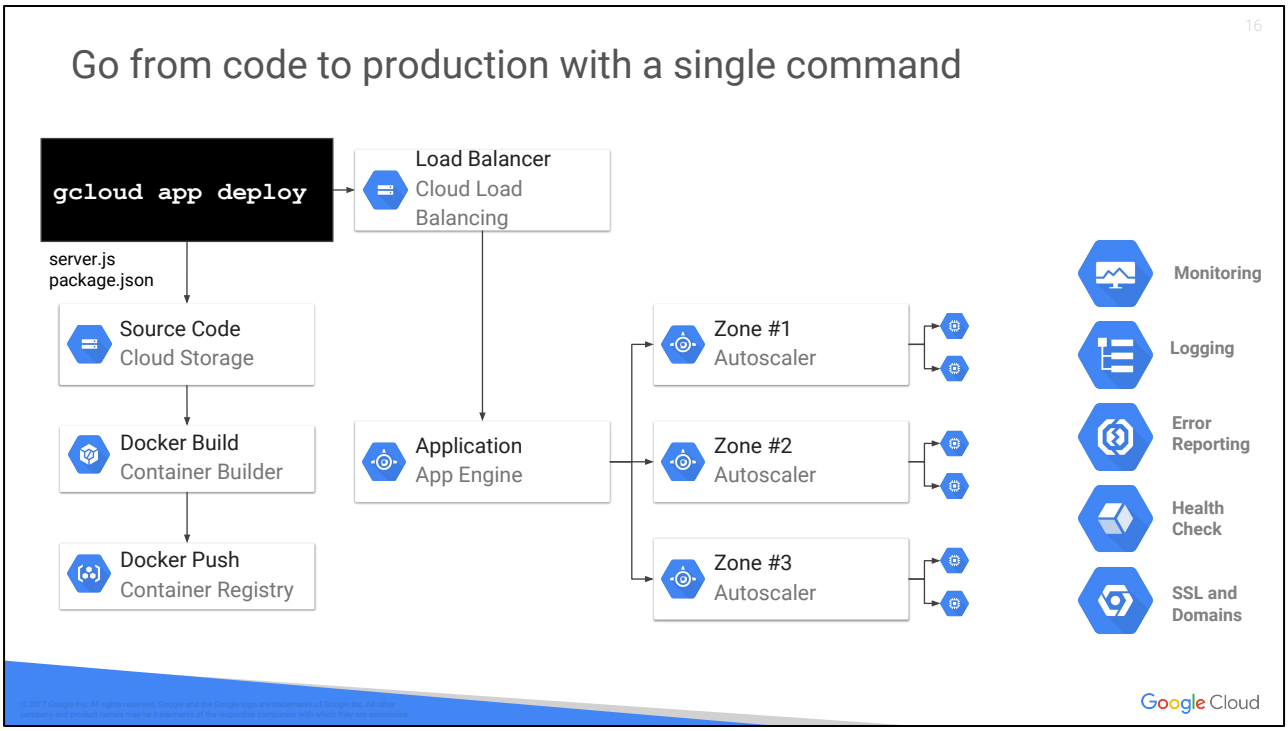
Images:
https://cloud.google.com/appengine/
https://cloud.google.com/container-engine/

Develop and deploy microservices



Google Cloud

You can use App Engine flexible environment to get started with a microservices architecture for your application. The application shown here deploys a frontend service and a backend service that are loosely coupled through Cloud Pub/Sub.



After you develop your application, you can deploy to your test, staging, or production environment with a single command: `gcloud app deploy`.

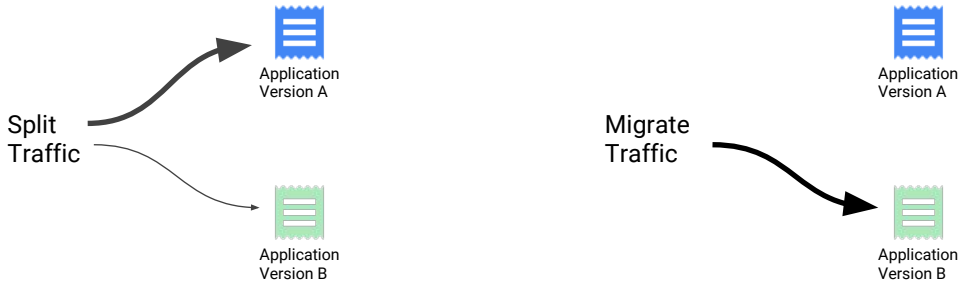
When you run this command, App Engine flexible environment automatically uploads your source code to Cloud Storage, builds a Docker image for your application with the runtime environment, and pushes the image to Google Container Registry. You do not need to set up any Docker scripts for applications that use one of the App Engine native runtime environments.

App Engine sets up a load balancer and runs your application in three zones to ensure that your application is always up and running. App Engine launches and autoscales Google Compute Engine instances to run your application and ensure that your application can scale up or down depending on traffic volume.

App Engine also sets up other services that are crucial for application monitoring and management, such as monitoring, logging, error reporting, health checks, and SSL.

App Engine flexible environment enables you to quickly deploy your application without manual infrastructure setup. It automatically applies critical, backward-compatible updates and security updates to the underlying operating system. If you need more control, you can directly ssh to the VM instances to work with custom runtimes and more.

Deploy safely with zero downtime



Google Cloud

App Engine flexible environment enables you to perform canary testing. You can verify a production release before serving any external traffic.

You can perform A/B testing of your application and deploy updates safely by easily splitting traffic between current and new versions. After you verify that the new version works, you can migrate all traffic to the new version without any downtime.

Use App Engine flexible environment for highly scalable web-focused applications

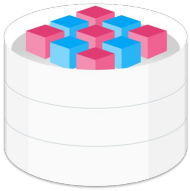
- ✓ Applications that are based on HTTP/s request-responses
- ✓ Applications that deploy public endpoints
- ✓ Continuous integration and delivery (CI/CD) pipelines with Jenkins or Spinnaker

Google Cloud

App Engine flexible environment is ideal for highly scalable web-focused applications. You can use App Engine flexible environment for applications that are based on HTTP/s request-responses and applications that deploy public endpoints. You can also implement CI/CD pipelines that use Jenkins or Spinnaker to deploy applications to App Engine.

Consider other compute environments

Protocols Other Than HTTP/S



Persistent Disks

Spiky or Very Low Traffic

Google Cloud

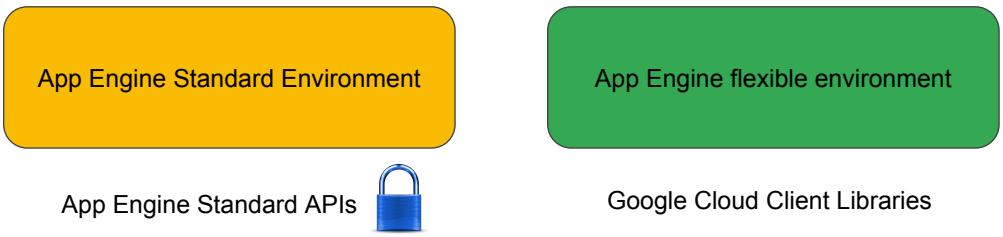
Consider other compute environments such as Container Engine or Compute Engine if your application needs to support ~~other~~ network protocols ~~besides~~ other than HTTP/S.

You cannot write data to persistent disks. App Engine flexible environment enables you to add volumes to tmpfs (files are in memory).

Currently, App Engine flexible environment is not ideal for applications with spiky or very low traffic. At least two instances are running at all times to serve traffic.

Images:
https://pixabay.com/en/success-curve-hand-finger-touch-1093889/
Other images from cloud.google.com

App Engine standard environment is an option for applications with spiky or very low traffic



Google Cloud

App Engine standard environment is an option for applications with spiky or very low traffic. Applications that run on App Engine standard environment must use App Engine Standard APIs. App Engine Standard APIs are supported only in the App Engine standard environment.

Applications that are developed using Google Cloud Client Libraries can be moved to another compute environment, such as Cloud Functions, Container Engine, or Compute Engine, if the needs of the application change.

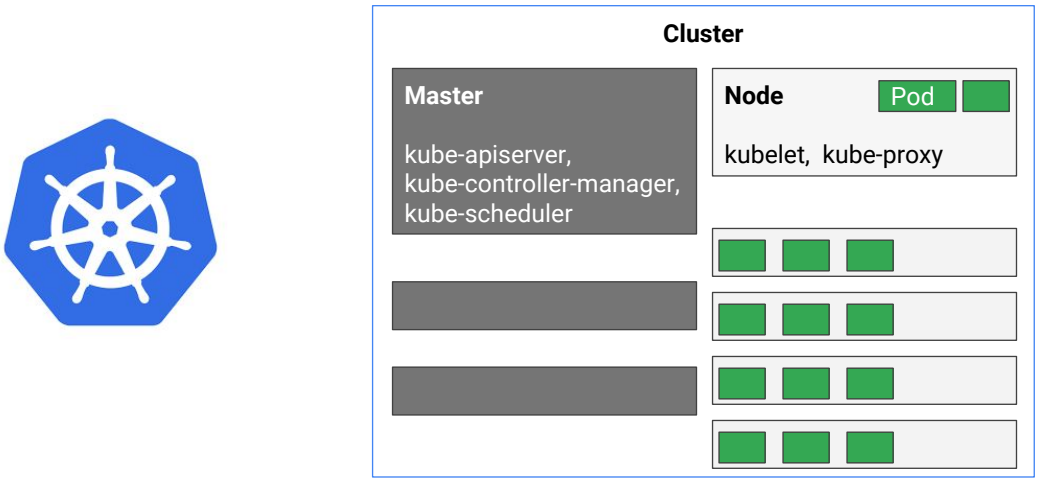
For more information about App Engine standard environment, see https://cloud.google.com/appengine/docs/standard/.

For more information about App Engine flexible environment, see:
App Engine flexible environment: https://cloud.google.com/appengine/docs/flexible/
You can run that on App Engine?: https://www.youtube.com/watch?v=sATG0OfdP4g

Container Engine

Google Cloud

Kubernetes is an open-source platform for deploying, scaling, and operating containers



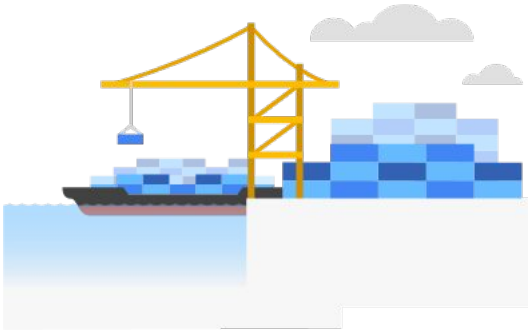
Google Cloud

Kubernetes is a leading open-source platform for deploying, scaling, and operating containers. Kubernetes, first developed at Google, is now a Cloud Native Computing Foundation project with a large and active community.

A cluster contains master and non-master nodes. The nodes in a cluster are the machines (VMs, physical servers, etc.) that run your applications. The Kubernetes master controls each node. A pod is a group of containers that share networking and storage resources on a node. A pod represents one or more related containers on your cluster. The Kubernetes Control Plane consists of the services running on the master and the kubelet processes running on the non-master nodes.

For more information, see https://kubernetes.io/docs/home/.

Container Engine is a managed service for Kubernetes



Google Cloud

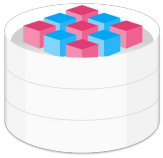
Container Engine is a managed service for Kubernetes.

For more information, see:
Container Engine: https://cloud.google.com/container-engine/
Container Engine Overview:
https://cloud.google.com/container-engine/docs/concepts/container-engine-overview
Upgrading a Container Cluster:
https://cloud.google.com/container-engine/docs/clusters/upgrade

Google Container Engine - The easiest way to use containers in production (Google Cloud Next '17): https://youtu.be/_yk1tTHYBvg

Images:
From cloud.google.com

Use Container Engine for complex, portable applications



Persistent Disks

Any Application Runtime Packaged as a Docker Container Image

Protocols Other Than HTTP/S

Google Cloud

With Container Engine, you can use persistent disks to build stateful applications.

Container Engine supports any application runtime that you can package as a Docker image. You can run applications that have been built by using the Google Cloud Client Libraries.

Container Engine is ideally suited for containerized applications, including third-party containerized software. You can run your container image on Kubernetes in a hybrid-cloud or multi-cloud environment. This is especially helpful when you have some parts of your application running on-premises and other parts in the cloud. You can use Container Engine to run applications that use network protocols other than HTTP/S, deploy private endpoints, or are based on a microservices architecture.

Images:
https://pixabay.com/en/usb-logo-input-flash-drive-memory-1773302/
Other images from cloud.google.com

Use Container Engine for greater control over how services are deployed and scaled

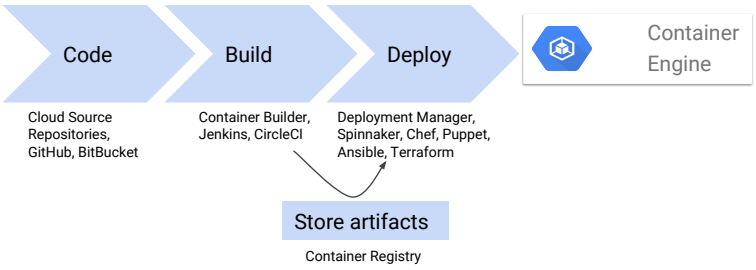
```
gcloud container clusters create

--machine-type=MACHINE_TYPE
--disk-size=DISK_SIZE
--num-nodes=NUM_NODES
…
```


Google Cloud

Container Engine provides greater control over how services are deployed and scaled. You can describe the compute, memory, and storage resources that your application containers need, and Container Engine will provision and manage the underlying cloud resources automatically. You can also specify limits on memory and CPU resources that a container can use. These limits enable Container Engine to better handle resource contentions.

Use Container Engine as a part of your CI/CD pipeline



As a part of a continuous integration and delivery (CI/CD) pipeline, you can generate a new Docker image for each code commit and automatically deploy the image to development, test, and production environments. Google Container Builder, Google Container Registry, and Container Engine can be be used to create a strong CI/CD system.

Consider other compute environments

If your application cannot be packaged as a container.

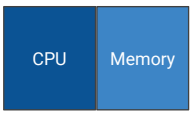
Google Cloud

Container Engine requires that your application be packaged and deployed as a container image.

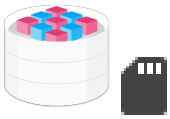
Compute Engine

Google Cloud

Run your application on high-performance, scalable VMs with Compute Engine

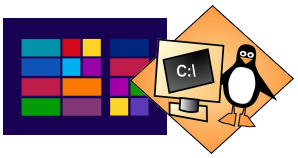


Predefined and Custom Machine Types



Persistent Disks and Local SSDs

Pre-emptible VMs



Windows, Linux OS, or Bring Your Own

Google Cloud

Compute Engine supports predefined machine types. You can also create custom machine types to create VMs with the optimal amount of CPU and memory for your workloads. Compute Engine supports persistent disks and local SSDs. You can also launch pre-emptible VMs for large compute and batch jobs.

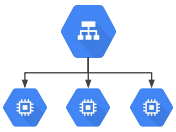
You can run your choice of OS, including Debian, CentOS, CoreOS, SUSE, Ubuntu, Red Hat, FreeBSD, or Windows Server 2008 R2, 2012 R2, and 2016. You can also use a shared image from the Cloud Platform community or bring your own operating system.

Images
https://material.io/icons/#ic_sd_storage
https://pixabay.com/en/background-windows-microsoft-720223/
https://pixabay.com/en/penguin-tux-linux-console-shell-146433/

Use Compute Engine for full control of infrastructure

Machine Type and OS

Software



Instance Groups with Global Load Balancing

Google Cloud

Compute Engine enables you to create highly customized VMs for specialized applications that have unique compute or operating system requirements.

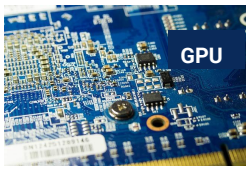
You can install and patch software running on the VM.

You can create managed instance groups based on an instance template. You can configure global load balancing and autoscaling of the managed instance groups. Compute Engine can perform health checks and replace unhealthy instances in an instance group. It autoscales instances based on the traffic volume in specific regions.

Images:
https://material.io/icons/#ic_file_download

Use Compute Engine for maximum flexibility

Third-party software



Graphics Processing Unit

Protocols Other Than HTTP/S

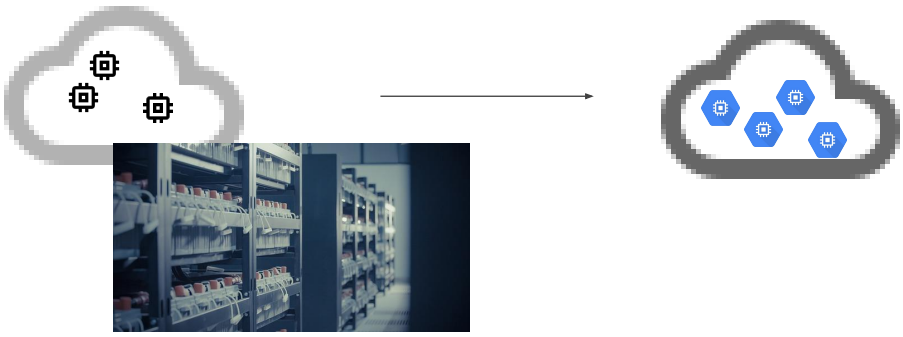
Google Cloud

Compute Engine offers you the most flexibility to configure your resources for the specific types of application that you need to run. You can run any third-party licensed software on Compute Engine.

You can attach GPUs to Compute Engine VMs to speed up machine learning and data processing workloads.

You can use Compute Engine for applications that require network protocols other than HTTP/S.

Use Compute Engine for lift-and-shift migration

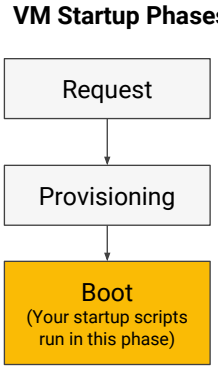


Google Cloud

Compute Engine is ideal for lift-and-shift migration. You can move virtual machines from your on-premises data center or another cloud provider to Google Cloud Platform without changing your application.

Images:
https://material.io/icons/#ic_cloud_queue
https://pixabay.com/en/data-center-the-engine-room-2476790/

Consider startup time



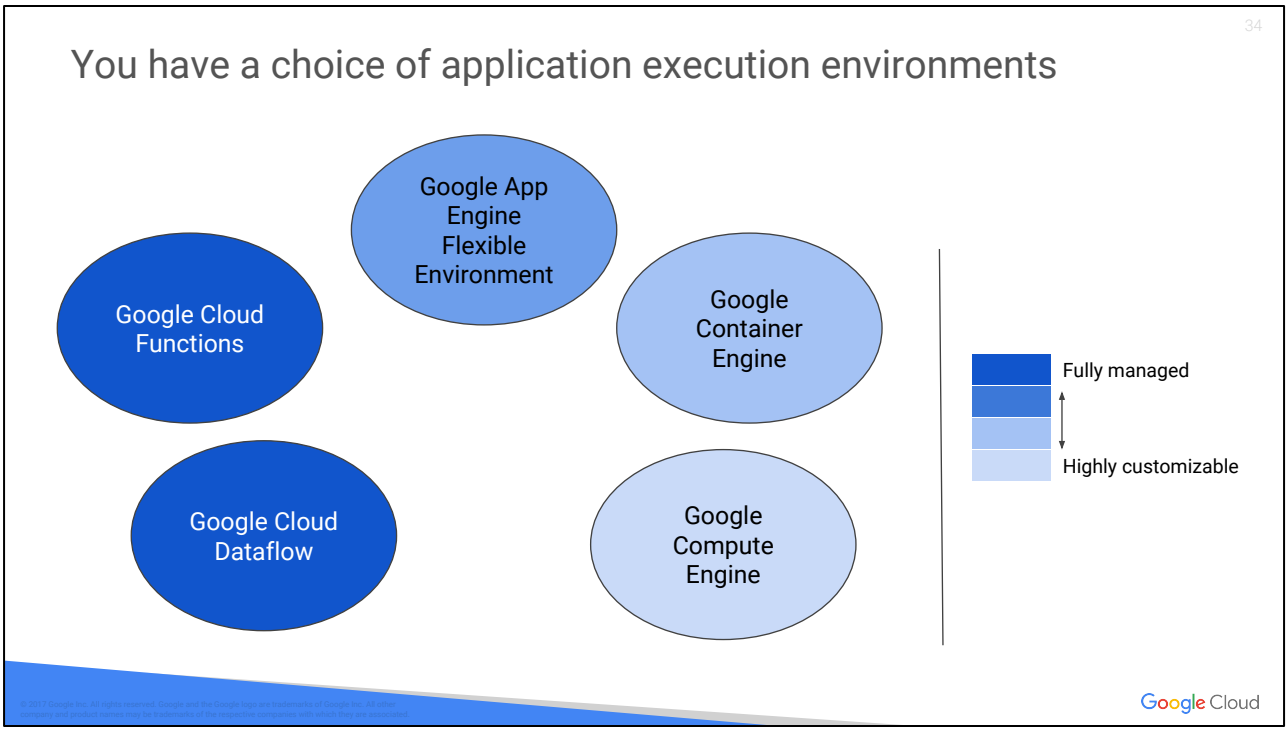
- Profile startup scripts.
- Consider custom image.
- Set appropriate target usage levels in autoscaling policy.

Google Cloud

A virtual machine can take about 60 seconds to spin up and become available. You can, however, launch hundreds of VMs within a few minutes.

To ensure that VM instances launch quickly, profile your startup scripts to identify and correct steps that take a long time to complete. If you are downloading and installing a lot of software, consider creating a custom image with all the software pre-installed. Plan ahead for bursts of traffic by setting appropriate target usage levels in your autoscaling policy.

For more information, see https://cloudplatform.googleblog.com/2017/07/three-steps-to-Compute-Engine-startup-time-bliss-Google-Cloud-Performance-Atlas.html.



To summarize, Google Cloud Platform provides a range of options to run your application code. These options range from services like Cloud Functions and Cloud Dataflow, which are fully managed, to App Engine, Container Engine, and Compute Engine, which offer steadily increasing abilities to customize your execution environment. Fully managed compute environments require minimal setup and operations. Highly customizable environments require greater operational and management effort to keep the application running optimally.

Choose the compute environment according to the requirements of your application and your team's ability to perform ongoing operations-related tasks.

For more information about choosing a compute option, see:

- Deciding between Compute Engine, Container Engine, App Engine and more (Google Cloud Next '17): https://youtu.be/g0dN8Hkh5H8
- Choosing the right compute option in GCP: a decision tree: https://cloudplatform.googleblog.com/2017/07/choosing-the-right-compute-option-in-GCP-a-decision-tree.html
- Choosing a Computing Option: https://cloud.google.com/docs/choosing-a-compute-option