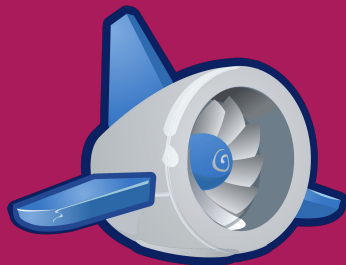


AppEngine

le Serverless

au service des développeurs !





Julien Landuré



Julien Landuré

CTO Zenika

GDG & DevFest Nantes Organizer
GDE Cloud

 @jlandure

History



Google App Engine Blog

News, notes, tips and tricks from the Google App Engine Team

Monday, April 7, 2008

Introducing Google App Engine + our new blog

Posted by Paul McDonald, Product Manager

At tonight's [Campfire One](#) we launched a [preview release](#) of Google App Engine -- a developer tool that enables you to run your web applications on Google's infrastructure. The goal is to make it easy to get started with a new web app, and then make it easy to scale when that app reaches the point where it's receiving significant traffic and has millions of users.

Google App Engine gives you access to the same building blocks that Google uses for its own applications, making it easier to build an application that runs reliably, even under heavy load and with large amounts of data. The [development environment](#) includes the following features:

- Dynamic webserving, with full support of common web technologies
- Persistent storage (powered by [Bigtable](#) and [GFS](#) with queries, sorting, and transactions)
- Automatic scaling and load balancing
- Google APIs for authenticating users and sending email
- Fully featured local development environment

Google App Engine packages these building blocks and takes care of the infrastructure stack, leaving you more time to focus on writing code and improving your application.

Today's launch is a preview release -- we're by no means feature-complete, and we're giving you early access because we really want your feedback. This preview of Google App Engine is available for the first 10,000 developers who sign up, and we plan to increase that number in near future.

During this preview period, applications are limited to 500MB of storage, 200M megacycles of CPU per day, and 10GB bandwidth per day. We expect most applications will be able to serve around 5 million pageviews per month. In the future, these limited quotas will remain free, and developers will be able to

More Blogs from Google

Visit our [directory](#) for more information about Google blogs.

Archives

April (7) ▼

AWS



Google Cloud



La promesse

- # Easy to build
- # Easy to manage
- # Easy to scale



Le principe

Exécution d'applications sur l'infrastructure Google en s'appuyant sur les standards actuels

L'infrastructure matérielle et logicielle est transparente pour le développeur et est gérée par Google

Adaptation et extensibilité de la plateforme en fonction de la charge et de la quantité de données à stocker

Fonctionnalités soumises à quotas.

SERVERLESS

So, what really is Serverless?

Operational model



No servers



Auto-Scales



Pay only for usage

Programming paradigm

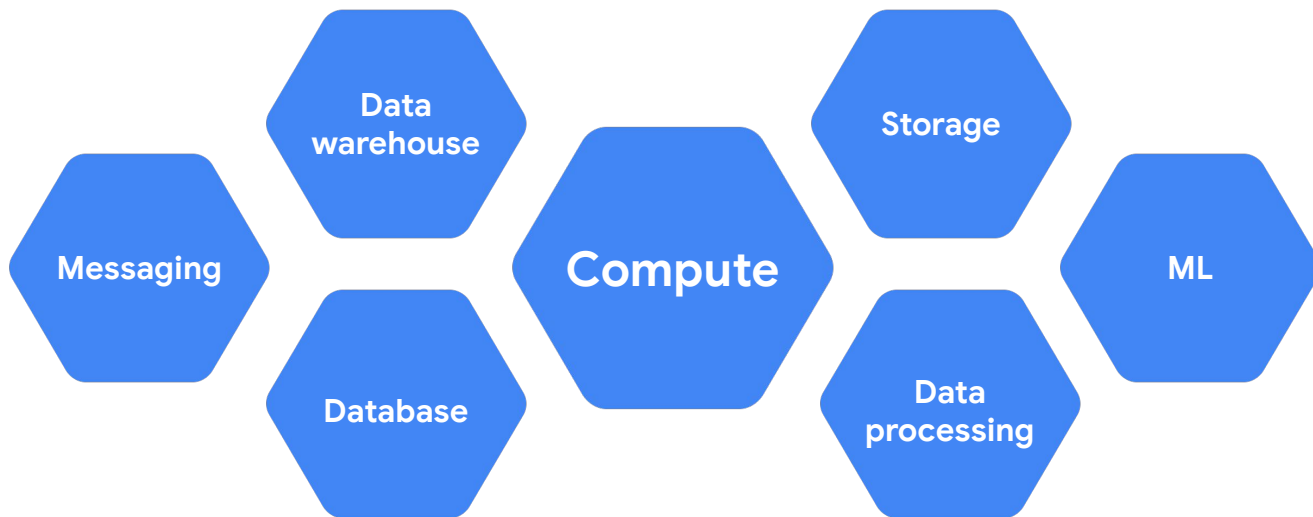


Service-based



Event-driven

Serverless is more than a set of functions



Serverless on GCP



Cloud PubSub



BigQuery



Cloud Datastore



Cloud Functions



App Engine



Cloud Storage



Cloud Dataflow



Cloud ML Engine

Serverless Compute on GCP



Cloud PubSub



BigQuery



Cloud Datastore



Cloud Functions



App Engine



Cloud Storage



Cloud Dataflow



Cloud ML Engine

Serverless compute options

Serverless functions



Cloud
Functions



Source code-based event
driven functions

Serverless web applications



App Engine



Source code-based web
apps and API backends

Serverless HTTP workloads



Cloud Run



HTTP containers
fully managed

HTTP containers
on GKE cluster

APPENGINE

App Engine



Unit: Application

Trigger: HTTP

Best used:

- When migrating existing application
- For HTTP services

App Engine: Build highly scalable applications

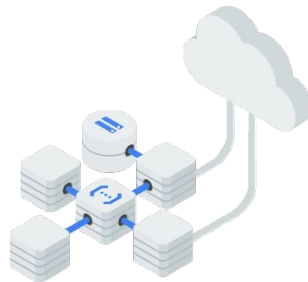
Familiar
dev languages



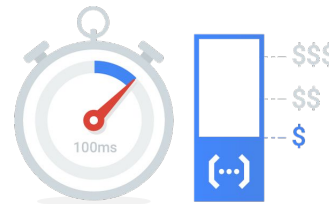
Zero config
deployments

gcloud app
deploy

Auto-scaling to support
peak traffic spikes



Pay only while
your code runs



App Engine Today



App Engine Standard (2nd gen)

- Open-source, idiomatic experience
- Use any extension, binary, or framework
- Supported Runtimes
 - **Python 3.7** (3.8 in beta)
 - **Java 11**
 - **Node.js 10** (12 in beta)
 - **PHP 7.3**
 - **Go 1.12**

2nd Generation idiomatic runtimes

gVisor based

Idiomatic dev experience

No more API restrictions

Supports native modules

Faster execution



2 modes



Mode Standard
(1st & 2nd generation runtime)

Mode Flexible

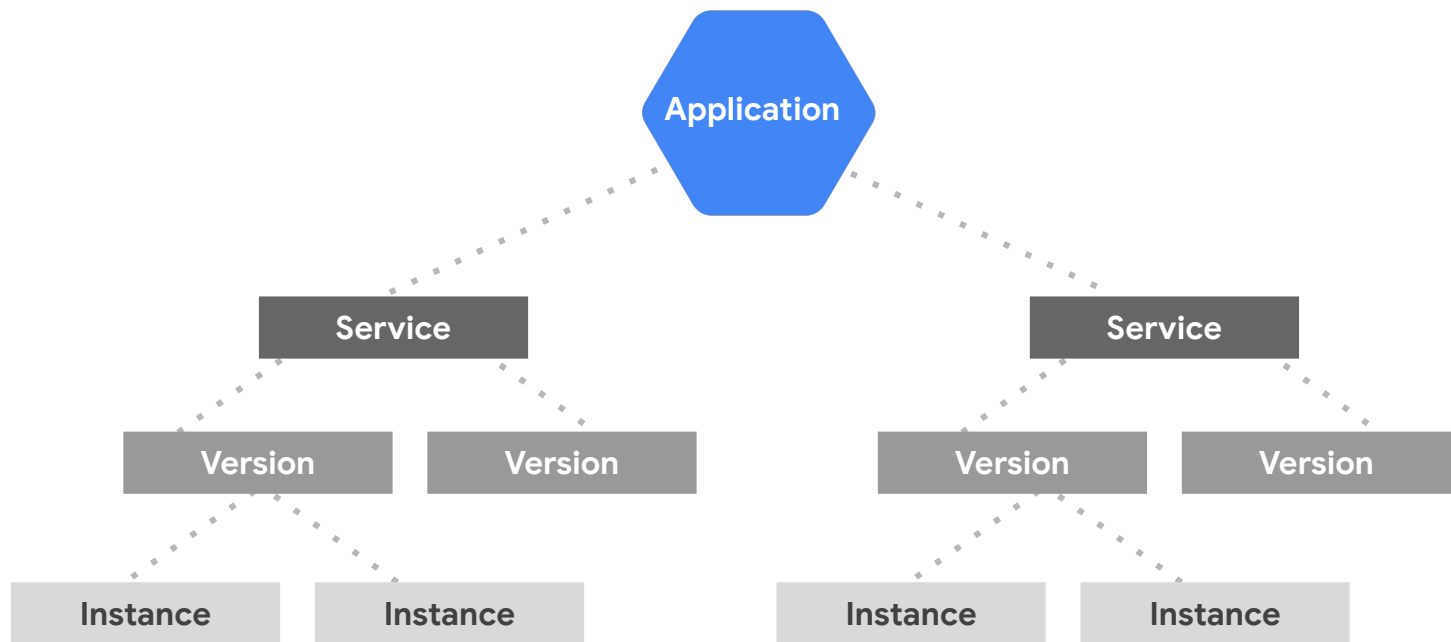
App Engine runtimes



Standard	Standard 2 nd generation	Flexible
Python 2.X, Java, PHP, Go	Node.js 10, Python 3.7, Go 1.12, PHP 7.3, Java 11	Node.js, Ruby, Java, Python, Go, PHP, .NET & custom container images
Sandboxed processes		Runs on managed VMs
GAE specific APIs + GCP APIs	Use GCP APIs directly	
No binary modules / libraries	any extension, binary, or framework open-source, idiomatic experience	

DEVELOPER EXPERIENCE

App Engine: Services and versions



App Engine: Traffic splitting

A/B testing and
Canary releasing
with a few clicks or
a single gcloud
command

The screenshot shows the 'Split traffic' configuration page in the Google Cloud App Engine console. On the left is a navigation menu with options: App Engine, Dashboard, Services, Versions (selected), Instances, Task queues, Cron jobs, Security scans, Firewall rules, and Quotas. The main content area is titled 'Split traffic' and includes a back arrow. Below the title is an explanatory text: 'You can split incoming traffic to different versions of your app. Traffic splitting is useful for slowly rolling out new versions or A/B testing different designs and features' with a 'Learn more' link. The 'Split traffic by' section has three radio buttons: 'IP address' (selected), 'Cookie', and 'Random'. The 'Traffic allocation' section shows two versions: version 3 (selected from a dropdown) which 'will receive the remaining' 10%, and version 2 (selected from a dropdown) which receives 90%. A slider visualizes this allocation. There is an '+ Add version' button and 'Save' and 'Cancel' buttons at the bottom.

App Engine

← Split traffic

You can split incoming traffic to different versions of your app. Traffic splitting is useful for slowly rolling out new versions or A/B testing different designs and features [Learn more](#)

Split traffic by

- ☒ IP address
- ☐ Cookie
- ☐ Random

Traffic allocation

3	will receive the remaining	10 %	×
2		90 %	×

+ Add version

Save Cancel

2nd Generation idiomatic runtimes

gVisor based

Idiomatic dev experience

No more API restrictions

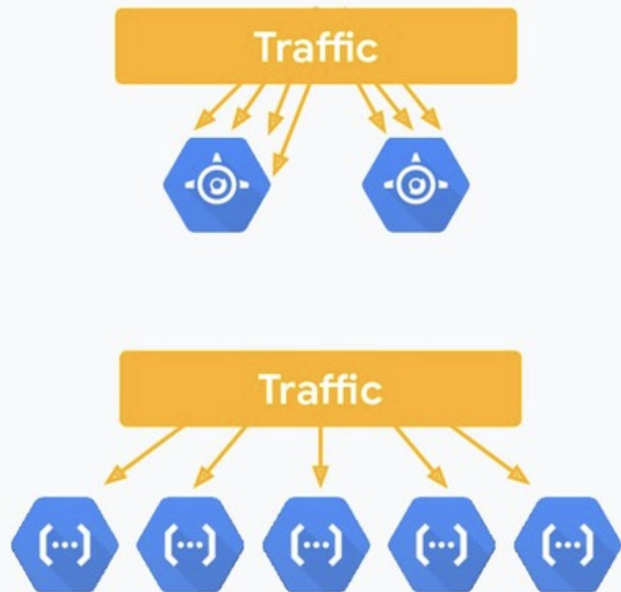
Supports native modules

Faster execution



Concurrency & scaling

- Scale up/down
- Scale-up may lead to "cold starts"
- One vs. multiple concurrent requests
- Traffic splitting across versions
- CDN static asset serving



Besoins



- ✓ Hébergement statique
- ✓ Stockage : Cloud Storage, Datastore, Cloud SQL
- ✓ Mémoire : Memcache
- ✓ Travail récurrent : Cronjob
- ✓ Gestion de file d'attentes : TaskQueue
- ✓ Recherche plein texte : Search

Les plus

Émulateur en local !

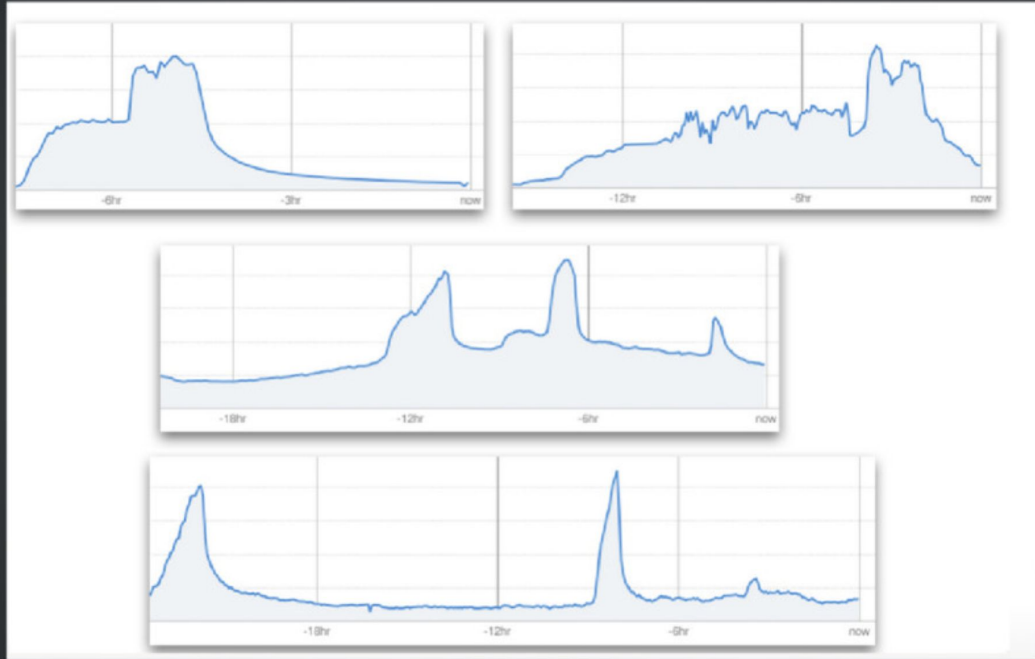
De la sécurité par défaut !

La mise à jour des protocoles réseau auto !

Des quotas généreux !



Pic de charge



En résumé

**Plusieurs langages supportés :
Python, Java, Go, PHP, NodeJS**

99.95% uptime SLA

Admin et Monitoring via une console web

Autoscaling de 0 à n

Gestion des versions d'une application

Un produit mature (beta en avril 2008)

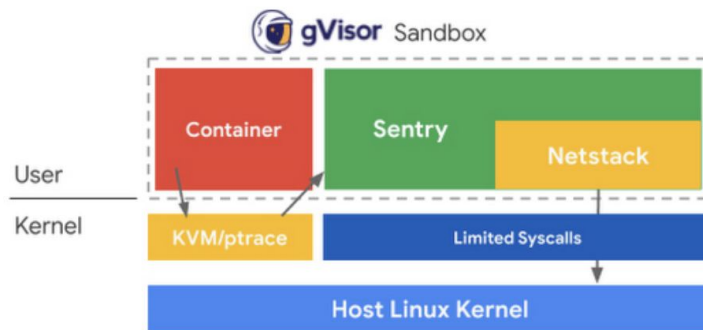
Disponible dans différentes régions

Les plus

Top des services managés ! avec *Cloud Run*

Mitigating CVE-2020-14386 with gVisor

If you saw the [Google Kubernetes Engine security bulletin](#), you may have noticed a line you hadn't seen before: **"Pods running in GKE Sandbox are not able to leverage this vulnerability."** If you're a user of **Cloud Run**, **Cloud Functions** or **App Engine standard environment**, you are protected from this vulnerability as well, and will not have experienced any service disruptions or been issued patching instructions. All these platforms utilize **gVisor** to securely "sandbox" workloads, which protected users from this vulnerability.



<https://cloud.google.com/blog/products/containers-kubernetes/how-gvisor-protects-google-cloud-services-from-cve-2020-14386>

Sides Projects

Julien Landuré
@jlandure

<https://github.com/jlandure/nuxt-static-deploy-appengine>

<https://github.com/jlandure/go-listing-appengine>

<https://github.com/jlandure/dummy-static-appengine>

<https://github.com/Zenika/alpine-firestore-backup>

<https://github.com/Zenika/alpine-chrome>

Thanks



Julien Landuré

@jlandure



App Engine

@gdgcloudabidjan @gdgnantes