GOOGLE APP ENGINE

CHAPTER 1

INTRODUCTION

What Is Google App Engine?....3
The App Engine runtime environment....3
The App Engine development environment....4
Quotas and limits....4
To get started....4
Overview of App Engine Features....5
Index of features....6
Languages and Runtimes....8
Data storage, retrieval, and search....11
Communications....15
Process management....18
Computation....19
App configuration and management....20

What Is Google App Engine?

Google App Engine is a Platform as a Service (PaaS) offering that lets you build and run applications on Google's infrastructure. App Engine applications are easy to build, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers for you to maintain. You simply upload your application and it's ready to go.

Google App Engine supports apps written in a variety of programming languages.

- Java: Using App Engine's Java runtime environment, you can build your application using standard Java technologies.
- Python: App Engine features a fast Python interpreter and standard Python libraries.
- PHP: App Engine uses Google's Cloud Platform services under the hood when you call standard PHP functions.
- Go: App Engine features a Go runtime environment that runs natively compiled Go code.

The App Engine runtime environment

Google App Engine makes it easy to build and deploy an application that runs reliably even under heavy load and with large amounts of data. It includes the following features:

- Persistent storage with queries, sorting, and transactions.
- Automatic scaling and load balancing.
- Asynchronous task queues for performing work outside the scope of a request.
- Scheduled tasks for triggering events at specified times or regular intervals.
- Integration with other Google cloud services and APIs.

Applications run in a secure, sandboxed environment, allowing App Engine to distribute requests across multiple servers, and scaling servers to meet traffic demands. Your application runs within its own secure, reliable environment that is independent of the hardware, operating system, or physical location of the server. For a full list of features, see our Features page.

The App Engine development environment

Software Development Kits (SDKs) for App Engine are available in all supported languages. Each SDK includes:

- All of the APIs and libraries available to App Engine.
- A simulated, secure sandbox environment, that emulates all of the App Engine services on your local computer.
- Deployment tools that allow you to upload your application to the cloud and manage different versions of your application.

The SDK manages your application locally, while the Administration Console manages your application in production. The Administration Console uses a web-based interface to create new applications, configure domain names, change which version of your application is live, examine access and error logs, and much more.

Quotas and limits

App Engine gives you 1 GB of data storage and traffic for free, which can be increased by enabling paid applications. However, some features impose limits unrelated to quotas to protect the stability of the system. For more details on quotas, including how you can edit them to suit your needs, see the Quotas page.

To get started...

1. Download the SDK.
2. Sign up for an account.
3. Read the getting started information for your language.
4. Check out the rest of the App Engine documentation.

Welcome to Google App Engine!

Overview of App Engine Features

This page summarizes App Engine's features. A feature may be available in every runtime language, or only in a subset of languages. The functionality of a feature is usually the same in all the runtimes where it's available, but there can be exceptions.

Every AppEngine feature is classified according to its status and availability:

- **General Availability** (GA) features are publicly available and are covered by App Engine's SLA and deprecation policy. The implementation of a GA feature is stable; any changes made to it will be backwards-compatible. Unless otherwise noted, the App Engine features described on this page are all in GA.

- **Beta** features are being developed to become GA features in a future App Engine release, but while they are in Beta their implementation may change in backward-incompatible ways. Beta features are publicly available.

- **Alpha** features may or may not become GA features in some future App Engine release, and their implementation may change in backward-incompatible ways. You must request access to use Alpha features.

Some features described here are provided by third-party vendors, others are open source projects. These will be marked *third-party* or *Open Source*.

The feature descriptions on this page are grouped according to the general functions they serve:

1. Languages and Runtimes
2. Data storage, retrieval, and search
3. Communications
4. Process Management
5. Computation
6. App configuration and management

Index of features

App Identity	A framework that provides access to the application's identity, and the ability to assert this identity using OAuth.
Appstats Analytics	Provides data visualization and analysis pertaining to the utilization of your application.
Blobstore	Allows your application to serve large data objects, such as video or image files, that are too large for storage in the Datastore service.
Capabilities	Detects outages and scheduled maintenance for specific services, so that an application may bypass them or notify users.
Channel	Creates a persistent connection between your application and JavaScript clients, so you can send messages in real time without polling.
Datastore	A schemaless object datastore, with scalable storage, a rich data modeling API, and an SQL-like query language.
Datastore Backup/Restore	Allows you to export or import data to or from your application's Datastore using the Admin Console.
Dedicated Memcache	Provides a fixed cache capacity assigned exclusively to your application.
Go Runtime	Build your application in the Go programming language.
Google Cloud Endpoints	Generates APIs for Android, iOS, and web clients, making it easier to create a web backend for your app.
Google Cloud SQL	A fully-managed web service that allows you to create, configure, and use relational databases that live in Google's cloud.
Google Cloud Storage Client Library	Read and write to Google Cloud Storage, with internal error handling and retry logic.
HRD Migration Tool	Migrates application data stored in the Blobstore or the deprecated Master/Slave Datastore into the GA High Replication Datastore.
Images	Manipulate, combine, and enhance images. Converts between image formats, access image metadata such as height and frequency of colors.
Java Runtime	Build your application in the Java programming language.
Logs	Programmatic access to application and request logs from within your application.
Mail	Send email messages on behalf of administrators and users with Google Accounts, and receive mail at various addresses.
MapReduce	An optimized adaptation of the MapReduce computing model for efficient distributed computing over large data sets.
Memcache	A distributed, in-memory data cache that can be used to greatly improve application performance.
Modules	Factor applications into logical components that can share stateful services and communicate in a secure fashion.
Multitenancy	Makes it easy to compartmentalize your data to serve many client organizations from a single instance of your application.
OAuth	Using Google Accounts and the OAuth API, any App Engine application can be an OAuth consumer.

OpenID	An open technology used for authenticating users across various web services.
PageSpeed	A family of tools that automatically optimizes the performance of your application.
Prospective Search	A querying service that allows your application to match search queries against real-time data streams.
PHP Runtime	Build your application in the PHP programming language.
Python Runtime	Build your application in the Python programming language.
Remote	Access App Engine services from any application. For example, access a production datastore from an app running on your local machine.
Scheduled Tasks	Configure tasks that run at defined times or regular intervals.
Search	Perform Google-like searches over structured data such as: plain text, HTML, atom, numbers, dates, and geographic locations.
SendGrid	Use SendGrid's library to send emails from your app and you can see statistics on opens, clicks, unsubscribes, spam reports and more.
Sockets	Supports outbound sockets using the language-specific, built-in libraries.
SSL for Custom Domains	Serve applications via HTTPS and HTTP from a custom domain rather than an`appspot.com` address.
Task Queue	Allows applications to perform work outside of a user request, using small, discrete tasks, that are executed later.
Task Queue REST API	Enables the use of an App Engine task queue over REST.
Task Queue Tagging	Leases up to a specified number of tasks with the same tag from the queue for a specified period of time.
Traffic Splitting	Routes incoming requests to different versions of your app, allowing you to do A/B testing and roll out new features incrementally.
Twilio	Enables your application to make and receive phone calls, send and receive text messages, and make VoIP calls from any phone, tablet, or browser.
URL Fetch	Uses Google's networking infrastructure to efficiently issue HTTP and HTTPS requests to URLs on the web.
Users	Allows applications to sign in users with Google Accounts or OpenID, and address these users with unique identifiers.
XMPP	Allows an application to send and receive chat messages to and from any XMPP-compatible chat messaging service.

Languages and Runtimes

Java Runtime

App Engine runs your Java web application using a Java 7 JVM in a safe sandboxed environment. App Engine invokes your app's servlet classes to handle requests and prepare responses in this environment.

App Engine uses the Java Servlet standard for web applications. You provide your app's servlet classes, JavaServer Pages (JSPs), static files and data files, along with the deployment descriptor (the web.xml file) and other configuration files, in a standard WAR directory structure. App Engine serves requests by invoking servlets according to the deployment descriptor.

The secured sandbox environment isolates your application for service and security. It ensures that apps can only perform actions that do not interfere with the performance and scalability of other apps. For instance, an app cannot spawn threads in some ways, write data to the local file system or make arbitrary network connections. An app also cannot use JNI or other native code. The JVM can execute any Java bytecode that operates within the sandbox restrictions.

The Google Plugin for Eclipse adds new project wizards and debug configurations to your Eclipse IDE for App Engine projects. App Engine for Java makes it especially easy to develop and deploy world-class web applications using Google Web Toolkit (GWT). The Eclipse plugin comes bundled with the App Engine and GWT SDKs. Third-party plugins are available for NetBeans and IntelliJ.

Java Developers Guide

Python Runtime

App Engine executes your Python application code using a pre-loaded Python interpreter in a safe sandboxed environment. Your app receives web requests, performs work, and sends responses by interacting with this environment.

A Python web app interacts with the App Engine web server using the WSGI protocol, so apps can use any WSGI-compatible web application framework. App Engine includes a simple web application framework, called webapp2, to make it easy to get started. For larger applications, mature third-party frameworks, such as Django, work well with App Engine.

The Python interpreter can run any Python code, including Python modules you

include with your application, as well as the Python standard library. The interpreter cannot load Python modules with C code; it is a "pure" Python environment.

The secured sandbox environment isolates your application for service and security. It ensures that apps can only perform actions that do not interfere with the performance and scalability of other apps. For instance, an app cannot write data to the local file system or make arbitrary network connections. Instead, apps use scalable services provided by App Engine to store data and communicate over the Internet. The Python interpreter raises an exception when an app attempts to import a module from the standard library known to not work within the sandbox restrictions.

Python Developers Guide

PHP Runtime **Beta**

The PHP runtime executes your application code in a sandboxed PHP 5.4 environment. Your app receives web requests, performs work, and sends responses by interacting with this environment. The runtime supports many of the standard PHP extensions. You cannot upload extensions written in C.

App Engine runs its own web server, which can be configured using an app.yaml file that's uploaded with your code. This file specifies how incoming HTTP requests to your application are directed to PHP scripts.

The sandbox isolates your application for reliability, scalability and security. For this reason, a small number of PHP functions are not available on App Engine, and others will raise an exception if used incorrectly. For instance, an app cannot write data to the local file system. Instead, apps can use scalable services provided by Google to store and process data and communicate over the Internet.

The PHP runtime provides a built-in Google Cloud Storage stream wrapper that allows you to use many of the standard PHP filesystem functions to access Google Cloud Storage.

PHP Developers Guide

Go Runtime **Beta**

App Engine runs Go version 1.2. The SDK provides an interface similar to the standard Go http package; writing Go App Engine apps is akin to writing stand-alone Go web servers.

The SDK includes the Go compiler and standard library; it has no additional dependencies. Because the Go runtime executes in a sandboxed environment, some

of the standard library functions will return errors and should not be used. For example, an os.ErrPermission will occur if an app tries to write to the local file system or make arbitrary network connections. You must use App Engine's scalable services to store data and communicate over the Internet.

You never need to invoke the Go compiler yourself. Your app is automatically re-built on the server side whenever you upload new code, and if you are running the local development server the SDK automatically recompiles sources on-the-fly when you change them.

The Go runtime environment for App Engine supports goroutines, but they are scheduled on a single thread. Goroutines can run concurrently but not in parallel. An application instance can handle multiple requests. For example, if one request is waiting for a datastore API call, the instance can process another request in the meantime.

Go Developers Guide

Managed VMs **Beta**

The Managed VM hosting environment lets you run App Engine applications on configurable Compute Engine Virtual Machines (VMs). This VM-based hosting environment offers more flexibility and provides more CPU and memory options. Applications that run on Managed VMs, are not subject to Java and Python runtime restrictions and they have access to all the Compute Engine machine types. You can also add third-party libraries and frameworks to your app.

Managed VMs Overview

Data storage, retrieval, and search

Datastore

App Engine Datastore is a schemaless NoSQL datastore providing robust, scalable storage for your web application, with the following features:

- No planned downtime
- Atomic transactions
- High availability of reads and writes
- Strong consistency for reads and ancestor queries
- Eventual consistency for all other queries

The Datastore holds data objects known as entities. An entity has one or more properties, named values of one of several supported data types: string, integer, or a reference to another entity. Each entity is identified by its kind, which categorizes the entity for the purpose of queries, and a key that uniquely identifies it within its kind. The Datastore can execute multiple operations in a single transaction. By definition, a transaction cannot succeed unless every one of its operations succeeds; if any of the operations fails, the transaction is automatically rolled back. This is especially useful for distributed web applications, where multiple users may be accessing or manipulating the same data at the same time.

Available in Python | Java | Go

Datastore Backup/Restore **Beta**

You can use the Datastore Admin tab of the Admin Console to backup selected kinds of entities, or to restore entities from a backup. You can backup using either Blobstore or Google Cloud Storage.

Available in Python | Java | Go

HRD Migration Tool

Use the migration tools found in the Application Settings tab of the Administration Console to migrate your application data from the deprecated Master/Slave Datastore to the High Replication Datastore (HRD). You must duplicate your application and then migrate the data. You may need to make some changes to your application to ensure an optimal migration.

Available in Python | Java | Go

Google Cloud SQL

Google Cloud SQL is a MySQL database that lives in Google's cloud. It has all the capabilities and functionality of MySQL, with a few additional features and a few unsupported features as listed below. Google Cloud SQL is easy to use, doesn't require any software installation or maintenance and is ideal for small to medium-sized applications.

Google Cloud SQL provides these features:

- Ability to host your MySQL databases in the cloud
- All data replicated in multiple locations for great availability and durability
- Choice of billing options:
- Per use option means you only pay for the time you access your data
- Package option allows you to control your costs for more frequent access
- Google Cloud SQL instances can have up to 16GB of RAM and 500GB data storage
- Create and manage instances in the Google Developers Console
- Data stored in datacenters in the EU or the US
- Cloud SQL customer data is encrypted when on Google’s internal networks and when stored in database tables and temporary files (with encryption of backups coming soon)
- Synchronous or asynchronous geographic replication
- Import or export databases using mysqldump
- Java and Python compatibility
- Support for MySQL wire protocol and MySQL connectors
- Support for connecting with the Secure Sockets Layer (SSL) protocol

Available in Python | Java | PHP | Go

Blobstore

The Blobstore API allows your application to serve data objects, called blobs, that are much larger than the size allowed for objects in the Datastore service. Blobs are useful for serving large files, such as video or image files, and for allowing users to upload large data files. Blobs are created by uploading a file through an HTTP request. Typically, your applications will do this by presenting a form with a file upload field to the user. When the form is submitted, the Blobstore creates a blob from the file's contents and returns an opaque reference to the blob, called a blob key, which you can later use to serve the blob. The application can serve the

complete blob value in response to a user request, or it can read the value directly using a streaming file-like interface.

Available in Python | Java | Go

Google Cloud Storage Client Library **Beta**

Google Cloud Storage is useful for storing and serving large files. Additionally, Cloud Storage offers the use of access control lists (ACLs), and the ability to resume upload operations if they're interrupted, and many other features. (The GCS client library makes use of this resume capability automatically for your app, providing you with a robust way to stream data into GCS.)

The GCS client library lets your application read files from and write files to buckets in Google Cloud Storage (GCS). This library supports reading and writing large amounts of data to GCS, with internal error handling and retries, so you don't have to write your own code to do this. Moreover, it provides read buffering with prefetch so your app can be more efficient.

Available in Python | Java | PHP | (Available in Go but no documentation exists)

Search

The Search API provides a model for indexing documents that contain structured data (text and HTML strings, atoms, dates, geopoints, numbers). Documents and indexes are saved in a separate persistent store optimized for search operations. You can search an index, and organize and present search results. The API supports partial text matching on string fields. The Search API can index any number of documents, however, a single search can return no more than 10,000 matching documents. The App Engine Datastore may be more appropriate for applications that need to retrieve very large result sets.

Available in Python | Java | Go

Prospective Search **Beta**

Prospective search is a querying service that allows your application to match search queries against real-time data streams. For every document presented, prospective search returns the ID of every registered query that matches the document.

Prospective search allows you to register a large set of queries and simultaneously match the queries against a single document. It is particularly useful for applications that process streaming data, for example:

Applications that match against all the updates on a social networking service, or

against high-frequency comments in a chat room. Applications that process data sources that provide notification, monitoring, or filtering services. To understand prospective search, it's helpful to compare it to the conventional retrospective search model. In a retrospective search application, such as Google search, the application must build, or have access to, an index of the data to be searched. Needing to pre-index the data makes it difficult and expensive to create real-time applications, because each query must be executed separately against a potentially large index.

In a prospective search application, such as Google Alerts, you register search queries and match them against new documents in real time, as the documents are inserted into your application. This allows you to create applications that efficiently monitor incoming live data. You are not limited to using existing, indexed data.

Applications often use both retrospective and prospective search capabilities to get the best of both worlds. For example, an application can use retrospective search to find matching documents indexed in the past while using prospective search to find matching documents as soon as they arrive.

Available in Python | Java

Memcache

App Engine supports two classes of the memcache service:

- Shared memcache is the free default for App Engine applications. It provides cache capacity on a best-effort basis and is subject to the overall demand of all applications served by App Engine.
- Dedicated memcache provides a fixed cache capacity assigned exclusively to your application. It's billed by the GB-hour of cache size. Having control over cache size means your app can perform more predictably and with fewer accesses to more costly durable storage. (Note that dedicated memcache is only available to HRD apps.)

Whether shared or dedicated, memcache is not a durable storage. Keys may be evicted when the cache fills up, according to the cache's LRU policy. Changes in the cache configuration or datacenter maintenance events may also flush some or all of the cache.

Available in Python | Java | PHP | Go

Dedicated Memcache

Dedicated memcache provides a fixed cache capacity assigned exclusively to your application. It's billed by the GB-hour of cache size. Having control over cache size

means your app can perform more predictably and with fewer accesses to more costly durable storage.

Available in Python | Java | PHP | Go

Logs

App Engine maintains two kinds of logs:

- Application logs contain arbitrary messages with a timestamp and log level.
- Request logs contain entries for each request handled by your app, with information such as the app ID, HTTP version, and so forth. Each request log includes a list of application logs associated with that request.

The Logs API provides access to your application's logs. You can also access the logs from the Admin Console.

Available in Python | Java | PHP | Go

Communications

Channel

The Channel API creates a persistent connection between an application and its users, allowing the application to send real time messages without the use of polling.

To use the API, you must add a JavaScript client in your web pages. The client performs these tasks:

- Connecting to the channel once it receives the channel’s unique token from the server
- Listening on the channel for updates regarding other clients and making appropriate use of the data (e.g. updating the interface, etc.)
- Sending update messages to the server so they may be passed on to remote clients

You application, acting as server, is responsible for:

- Creating a unique channel for individual JavaScript clients
- Creating and sending a unique token to each JavaScript client so they may

connect and listen to their channel

- Receiving update messages from clients via HTTP requests
- Sending update messages to clients via their channels
- Optionally, managing client connection state.

Available in Python | Java | Go

Google Cloud Endpoints

Google Cloud Endpoints consists of tools, libraries and capabilities that allow you to generate APIs and client libraries from an App Engine application, referred to as an API backend, to simplify client access to data from other applications. Endpoints makes it easier to create a web backend for web clients and mobile clients such as Android or Apple's iOS.

For mobile developers, Endpoints provides a simple way to develop a shared web backend and also provides critical infrastructures, such as OAuth 2.0 authentication, eliminating a great deal of work that would otherwise be needed. Furthermore, because the API backend is an App Engine app, the mobile developer can use all of the services and features available in App Engine, such as Datastore, Google Cloud Storage, Mail, Url Fetch, Task Queues, and so forth. And finally, by using App Engine for the backend, developers are freed from system admin work, load balancing, scaling, and server maintenance.

It is possible to create mobile clients for App Engine backends without Endpoints. However, using Endpoints makes this process easier because it frees you from having to write wrappers to handle communication with App Engine. The client libraries generated by Endpoints allow you to simply make direct API calls.

Available in Python | Java

Mail

App Engine applications can send email messages on behalf of the app's administrators, and on behalf of users with Google Accounts. Apps can receive email at various addresses. Apps send messages using the Mail service and receive messages in the form of HTTP requests initiated by App Engine and posted to the app.

Available in Python | Java | PHP | Go

SendGrid **Third-party**

You can use SendGrid to power your emails on Google App Engine. Using SendGrid can improve your deliverability and provide transparency into what actually happens to those emails your app sends. You can see statistics on opens, clicks, unsubscribes, spam reports and more through either the SendGrid interface or its API.

Available in Python | Java | PHP

Sockets **Beta**

App Engine supports regular outbound sockets in all runtimes, without requiring you to import any special App Engine libraries or add any special App Engine code. However, there are certain limitations and behaviors you need to be aware of when using sockets. The details vary depending on the runtime. Read the runtime documentation for more information.

Available in Python | Java | PHP | Go

Twilio **Third-party**

Twilio is powering the future of business communications, enabling developers to embed voice, VoIP, and messaging into applications. They virtualize all infrastructure needed in a cloud-based, global environment, exposing it through the Twilio communications API platform. Applications are simple to build and scalable. Enjoy flexibility with pay-as-you go pricing, and benefit from cloud reliability.

Twilio Voice enables your application to make and receive phone calls. Twilio SMS enables your application to send and receive text messages. Twilio Client allows you to make VoIP calls from any phone, tablet, or browser and supports WebRTC.

Available in Python | Java | PHP

URL Fetch

App Engine applications can communicate with other applications or access other resources on the web by fetching URLs. An app can use the URL Fetch service to issue HTTP and HTTPS requests and receive responses. The URL Fetch service uses Google's network infrastructure for efficiency and scaling purposes.

Available in Python | Java | PHP | Go

XMPP

An App Engine application can send and receive chat messages to and from any XMPP-compatible chat messaging service, such as Google Talk. An app can send and receive chat messages, send chat invites, request a user's chat presence and status,

and provide a chat status. Incoming XMPP messages are handled by request handlers, similar to web requests.

Some possible uses of chat messages include automated chat participants ("chat bots"), chat notifications, and chat interfaces to services. A rich client with a connection to an XMPP server (such as Google Talk) can use XMPP to interact with an App Engine application in real time, including receiving messages initiated by the app. (Note that such a client using Google Talk must use the user's password to make an XMPP connection, and cannot use a Google Accounts cookie.)

Currently, an app cannot participate in group chats. An app can only receive messages of types "chat" and "normal". An app can send messages of any type defined in RFC 3921.

Available in Python | Java | Go

Process management

Task Queue

With the Task Queue API, applications can perform work outside of a user request, initiated by a user request. If an app needs to execute some background work, it can use the Task Queue API to organize that work into small, discrete units, called tasks. The app adds tasks to task queues to be executed later.

App Engine provides two different queue configurations:

- Push queues process tasks based on the processing rate configured in the queue definition. App Engine automatically scales processing capacity to match your queue configuration and processing volume, and also deletes tasks after processing. Push queues are the default.

- Pull queues allow a task consumer (either your application or code external to your application) to lease tasks at a specific time for processing within a specific timeframe. Pull queues give you more control over when tasks are processed, and also allow you to integrate your application with non-App-Engine code using the Task Queue REST API. When using pull queues, your application needs to handle scaling of instances based on processing volume, and also needs to delete tasks after processing.

Available in Python | Java | PHP | Go

Task Queue REST API **Beta**

Provides a REST interface so any web client app can manage your application's task queues and the tasks in them.

Available in Python | Java | PHP | Go

Task Queue Tagging **Beta**

The tagging API lets pull queue consumers lease a specified number of tasks with the same tag from a pull queue.

Available in Python | Java | Go

Scheduled Tasks

The App Engine Cron Service allows you to configure regularly scheduled tasks that operate at defined times or regular intervals. These tasks are commonly known as cron jobs. These cron jobs are automatically triggered by the App Engine Cron Service. For instance, you might use this to send out a report email on a daily basis, to update some cached data every 10 minutes, or to update some summary information once an hour.

A cron job will invoke a URL, using an HTTP GET request, at a given time of day. An HTTP request invoked by cron can run for up to 10 minutes, but is subject to the same limits as other HTTP requests.

Free applications can have up to 20 scheduled tasks. Paid applications can have up to 100 scheduled tasks.

Available in Python | Java | PHP | Go

Computation

Images

The Images API provides the ability to describe and manipulate image data. The API provides information about an image, such as its format, width, height, and a histogram of color values. It can resize, rotate, flip, and crop images. It also has the ability to composite multiple images into one image, enhance images automatically, and convert images between several formats

The Images service can accept image data directly from the app, or it can use data retreived from Blobstore or Google Cloud Storage.

Available in Python | Java | Go

MapReduce **Open Source**

App Engine MapReduce is an open source library that is built on top of App Engine services, including Datastore and Task Queues. You must include the library with your application, it provides:

- A programming model for large-scale distributed data processing
- Automatic parallelization and distribution within your existing codebase
- Access to Google-scale data storage
- I/O scheduling
- Fault-tolerance, handling of exceptions
- User tunable settings to optimize for speed/cost
- Tools for monitoring status

Available on Github for Java and Python

App configuration and management

App Identity

The Application Identity service provides the ability for an application to identify itself, by retrieving its App ID or the hostname part of its URL. The app's identity may be used to generate a URL or email address, or to make some run-time decision.

Many Google APIs support OAuth assertions to identify the source of a request. The App Identity API can create tokens that can be used to assert that the source of a request is the application itself. This token can be included in the HTTP headers of a call to identify the calling application. The OAuth token only works against Google systems. However you can use the underlying signing technology in the API to assert the identity of your application to other systems.

Available in Python | Java | PHP | Go

Appstats Analytics **Trusted Tester**

Appstats Analytics provides data visualization and analysis pertaining to the

utilization of your application. This is a trusted tested feature; you must apply for permission to use it.

Available in Python | Java | Go

OAuth Beta

As users put more of their information online, it becomes increasingly useful for networked applications to access that data, even when the data is not stored in the application itself. If access to that data is restricted by a username and password, the data-consuming application (the consumer) needs a way to tell the service holding the data (the service provider) that it is acting on behalf of the user. One way to do this is for the user to give her username and password to the consumer—but with most services, this gives the consumer too much control over the user's account with the service provider, and risks exposing the user's password if the consumer has a security breach.

The OAuth protocol provides a way for a consumer to authenticate with a service provider and act on behalf of a user without the user having to give the consumer her username and password. For example, a travel application may have a feature to add an itinerary to a user's Google Calendar automatically. The travel application communicates with Google Calendar, and directs the user's browser to a Google Accounts authorization screen. The user signs in using her Google account, and tells Google Calendar that the travel application has permission to access her calendar data. The travel application can now access the user's calendar and add travel itineraries. It can do so until the user revokes this permission from her Google Accounts settings.

Using Google Accounts and the OAuth API, any App Engine application can be an OAuth service provider. Google Accounts handles the details of the OAuth protocol. The app accesses the consumer authentication information using the OAuth API during a service request, just as it would with the Users API if the user were accessing the app directly.

Available in Python | Java | Go

OpenID Beta

OpenID is an open technology used for authenticating users across various web services. A user creates an account on a service that acts as an OpenID *provider*. When logging in to a provider, the user receives an OpenID *identifier*, which can be used to sign in to other services, called OpenID *relying parties*.

An App Engine app can be configured as an OpenID relying party, and use OpenID for

sign in. App Engine apps cannot be Open ID providers.

Available in Python | Java | PHP | Go

PageSpeed **Beta**

The PageSpeed service automatically optimizes the way your application serves web content. It compresses and caches your HTML. It can also inline and combine resources. Fast pages create a better user experience and lead to increased traffic and reduced bounce rates.

Available in Python | Java | PHP | Go

Capabilities

You can use the Capabilities API to detect the availability of services (like the Images service) or specific capabilities of a service (such as datastore reads and writes). You can reduce downtime in your application by testing for the availability of a capability before trying to use it. The services that support this API vary depending which runtime you are using.

Available in Python | Java | Go

Modules

Modules are used to factor large applications into logical components that can share stateful services and communicate in a secure fashion. An app that handles customer requests might include separate modules to handle other tasks:

- API requests from mobile devices
- Internal, admin-like requests
- Backend processing such as billing pipelines and data analysis

Modules can have different versions, performance levels, and authorization. While running, a particular module will have one or more instances, which may be managed statically or dynamically. Incoming requests are routed to an existing or new instance of the appropriate module.

Scaling types control the creation of instances. There are three scaling types. Each type offers a variety of instance classes, with different amounts of CPU and Memory:

- A module with manual scaling runs continuously, allowing you to perform complex initialization and rely on the state of its memory over time.

- A module with basic scaling will create an instance when the application receives a request. The instance will be turned down when the app becomes idle. Basic scaling is ideal for work that is intermittent or driven by user activity.

- Automatic scaling is the scaling policy that App Engine has used since its inception. It is based on request rate, response latencies, and other application metrics. Previously users could use the Admin Console to configure the automatic scaling parameters (instance class, idle instances and pending latency) for an application's frontend versions only. These settings now apply to every version of every module that has automatic scaling.

Available in Python | Java | PHP | Go

Multitenancy

Multitenancy is a software architecture in which one instance of an application, running on a remote server, serves many client organizations, or *tenants*.

Google App Engine provides the Namespaces API, which includes a namespace manager. The Namespaces API is also incorporated in other namespace-enabled APIs. When you set the namespace in the namespace manager, all the namespace-enabled APIs use that namespace by default. This lets you partition data across tenants by specifying a unique namespace for each tenant. For instance, using namespaces with the Datastore API, all users can share the same data schema, but different users see different content.

The Namespaces API is integrated with Google Apps, allowing you to use your Google Apps domain as the current namespace. Because Google Apps lets you deploy your app to any domain that you own, you can easily set unique namespaces for all domains linked to your Google Apps account.

Available in Python | Java | Go

Remote

Lets external applications transparently access App Engine services. For example, you can use Remote API to access a production datastore from an app running on your local machine.

Available in Python | Java | Go

SSL for Custom Domains

Allows applications to be served via both HTTPS and HTTP via a custom domain instead of an`appspot.com` address.

Available in Python | Java | PHP | Go

Traffic Splitting

Allows you to roll out features for your app slowly over a period of time and do A/B testing. Traffic Splitting works by splitting incoming requests to different versions of your app.

Available in Python | Java | PHP | Go

Users

App Engine applications can authenticate users using any one of three methods: Google Accounts, accounts on your own Google Apps domains, or OpenID identifiers. (Note that the support for OpenID is in Beta.) An application can detect whether the current user has signed in, and can redirect the user to the appropriate sign-in page to sign in or, if your app uses Google Accounts authentication, create a new account. While a user is signed in to the application, the app can access the user's email address (or OpenID identifier if your app is using OpenID). The app can also detect whether the current user is an administrator, making it easy to implement admin-only areas of the app.

Available in Python | Java | PHP | Go