Go in the Cloud

This lesson explains how Go and Google App Engine collaborate to create a flexible environment for cloud computing.

WE'LL COVER THE FOLLOWING

Google cloud computing

Google App Engine (from now on shortened to GAE) is the Google way of cloud computing, which includes executing your web applications and storing your data on the vast Google infrastructure without having to worry about the servers, the network, the operating system, the data store, and so on. This collection of resources is commonly referred to as the cloud, and its maintenance is the sole responsibility of Google itself. For you, as a developer, only your application counts, and the services it can deliver to its users, who can work with and run your application on any device which can connect to the internet.

You only pay for the resources (CPU processing time, network bandwidth, disk storage, memory, and so on) your software really needs. When there are peak moments, the cloud platform automatically increases resources for your application and decreases them when they are no longer needed. This is also known as scalability. Scalability is one of the biggest advantages of cloud

computing.

Collaborative applications (where groups of people work together on, share data, communicate, and so on), applications that deliver services and applications that perform large computations, are excellent candidates for cloud computing. GAE has had Go support since 2011. You can visit its start page here.

Go applications running in the cloud are a kind of web applications, so the typical user interface for a cloud application is a browser environment. To run Go in GAE, you have to choose its runtime environment. This can either be a *standard* and *flexible* environment:

Environment provided by GAE

Standard environment

Here, your application runs within its own secure and reliable environment that is independent of the hardware, operating system, or physical location of the server. The focus is on executing reliably under a heavy load and with large amounts of data.

Flexible environment

Based on Google Compute Engine, this environment focuses on balancing the load, and optimizing for adaptive scaling.

For both environments, you first need to install the Cloud SDK. This installs the `gcloud` command-line tool; for more info, click here. Your cloud app is configured through **.yaml** files (see yaml), of which the **app.yaml** is the most important.

Here is a simple example of a cloud Go app:

Environment Variables

Key:	Value:
GOROOT	/usr/local/go
GOPATH	//root/usr/local/go/src
PATH	//root/usr/local/go/src/bin:/usr/local/go...

```
package main
import (
"fmt"

"log"
"net/http"
"os"
)

func main() {
  http.HandleFunc("/", indexHandler)
  port := os.Getenv("PORT")
  if port == "" {
    port = "3000"
    log.Printf("Defaulting to port %s", port)
  }
  log.Printf("Listening on port %s", port)
  log.Fatal(http.ListenAndServe(fmt.Sprintf(":%s", port), nil))
}

// indexHandler responds to requests with our greeting.
func indexHandler(w http.ResponseWriter, r *http.Request) {
  if r.URL.Path != "/" {
    http.NotFound(w, r)
    return
  }
  fmt.Fprint(w, "Hello, World!")
}
```

Click **RUN** and wait for the terminal to start. Once it's started, click the URL next to the **Your app can be found at** and see the result.

> **Remark:** To run it locally, change **line 13** to `port = 8080`. To test this program open browser at http://localhost:8080.

You see that looking at the code, it is just a normal web application. You can monitor your app and its resource usage through a sophisticated web dashboard.

That's it about networking and applications that communicate over the Internet. We have covered almost all advanced programming concepts in Go. In the next chapter, the common programming mistakes are mentioned so you can avoid making them. Furthermore, some common patterns are revised with a purpose to gain proficiency in Go programming.