

Sending Mails with SMTP

This lesson explains how Go code can be used to send an email using an SMTP server.

WE'LL COVER THE FOLLOWING ^

- Overview of SMTP in Go
- Explanation

Overview of SMTP in Go

The package `net/smtp` implements the *Simple Mail Transfer Protocol* for sending mail. It contains a `Client` type that represents a client connection to an SMTP server:



- `Dial` returns a new `Client` connected to an SMTP server.
- Set `Mail` (=from) and `Rcpt` (= to)
- `Data` returns a writer that can be used to write the data, here with `buf.WriteTo(wc)`.

Explanation

```
package main
import (
    "bytes"
    "log"
    "net/smtp"
)

func main() {
    // Connect to the remote SMTP server.
    client, err := smtp.Dial("smtp.gmail.com:587")
    if err != nil {
        log.Fatal(err)
    }
    // Set the sender and recipient
```

```

// Set the sender and recipient.
client.Mail("sender email address")
client.Rcpt("receiver email address")
// Send the email body.
wc, err := client.Data()
if err != nil {
    log.Fatal(err)
}
defer wc.Close()
buf := bytes.NewBufferString("This is the email body.")
if _, err = buf.WriteTo(wc); err != nil {
    log.Fatal(err)
}
}
}

```

SMTP Server

In the code above, we need the package `net/smtp`, which is imported at **line 5**. First, we need to connect to an active remote SMTP server, which is done with the `Dial` method at **line 10**, creating a client instance. Error-handling (from **line 11** to **line 13**) exits the program on error.

We set up the *sender* and *receiver* email address at **line 15** and **line 16**, respectively. **Line 18** constructs the `Data` method on `client`, which makes a client writer `wc` with similar error-handling from **line 19** to **line 21**. At **line 22**, we make sure that the writer `wc` will be closed. Then, at **line 23**, we make a buffered string and write it to `wc` at **line 24**. This if-statement is combined with error-handling, logging an eventual error and exiting the program at **line 25**.

The function `SendMail` can be used if authentication is needed and when you have a number of recipients. It connects to the server at `addr`, switches to TLS (Transport Layer Security encryption and authentication protocol), authenticates with the mechanism if possible, and then sends an email from *address* `from` to *addresses* `to`, with the message `msg`:

```

func SendMail(addr string, a Auth, from string, to []string, msg []byte) error

```

Go through the following illustrations that explain how to set a Gmail account for sending an email before running a program.

Google Account Search Google Account

Home

Personal info

Data & personalization

Security Click on **Security**

People & sharing

Payments & subscriptions

Help

Send feedback

Manage your info, privacy, and security to make Google work better for you

Privacy & personalization
See the data in your Google Account and choose what activity is saved to personalize your Google experience
[Manage your data & personalization](#)

Security issues found
Protect your account now by resolving these issues
[Secure account](#)

Account storage
Your account storage is shared across Google services, like Gmail and Photos
3% used – 0.53 GB of 15 GB
[Manage storage](#)

Take the Privacy Checklist
This step-by-step guide helps you choose the privacy settings that are right for you
[Get started](#)

Only you can see your settings. You might also want to review your settings for Maps, Search, or whichever Google services you use most. Google keeps your data private, safe, and secure. [Learn more](#)

Step 1: Open your Account Settings

1 of 7

Google Account Search Google Account

Home

Personal info

Data & personalization

Security

People & sharing

Payments & subscriptions

Help

Send feedback

Security issues found
Protect your account now by resolving these issues
[Secure account](#)

Signing in to Google
Password Last changed Aug 10, 2015
Use your phone to sign in Off
2-Step Verification Off

Ways we can verify it's you
These can be used to make sure it's really you signing in or to reach you if there's suspicious activity in your account

Recovery phone Add a mobile phone number >

Recovery email Add an email address >

Recent security events

Turn 2-Step Verification on.
If already on move to the next step

Step 2: Enable 2-Step Verification

2 of 7

Google Account Search Google Account

Home
Personal info
Data & personalization
Security
People & sharing
Payments & subscriptions
Help
Send feedback

Security

Settings and recommendations to help you keep your account secure

We keep your account protected

The Security Checkup gives you personalized recommendations to secure your account

[Get started](#)

Signing in to Google

| | | |
|----------------------|---------------------------|---|
| Password | Last changed Aug 10, 2015 | > |
| 2-Step Verification | On | > |
| App passwords | None | > |

Ways we can verify it's you

These can be used to make sure it's really you signing in or to reach you if there's suspicious activity in your account

Recovery phone Add a mobile phone number >

Click **App passwords** to generate a password
If already generated then use the generated password.
If you don't remember it make a new one.

Step 3: Generate an App Password

3 of 7

After the verification, the following page will open

Google Account

App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

You don't have any app passwords.

Select the app and device you want to generate the app password for.

Select app

Select device

[GENERATE](#)

Privacy Policy · Terms of Service · Help

Step3: Generate App Password

4 of 7

← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

You don't have any app passwords.

Select the app and device you want to generate the app password for.

Select app

Select device

GENERATE

Mail

Calendar

Contacts

YouTube

Other (Custom name)

Choose **Other** option and give any name by your own choice

[Privacy Policy](#) · [Terms of Service](#) · [Help](#)

Step3: Generate App Password

5 of 7

← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

You don't have any app passwords.

Select the app and device you want to generate the app password for.

Select app

Select device

GENERATE

iPhone

iPad

BlackBerry

Mac

Windows Phone

Windows Computer

Other (Custom name)

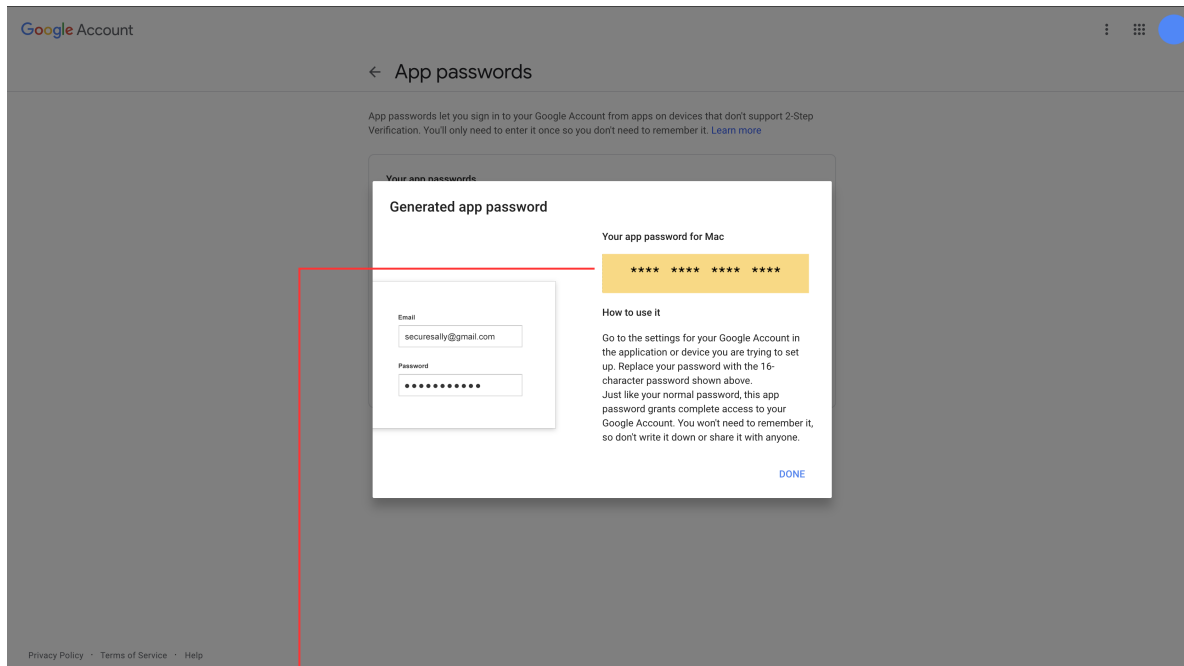
After selecting **Device** and **App** click **Generate**

Choose your device from the pop-up
If your device isn't listed then select **Other**
and name you device

[Privacy Policy](#) · [Terms of Service](#) · [Help](#)

Step3: Generate App Password

6 of 7



→ Don't forget to copy this 16 digit password.
You can use this password in the program
to send email rather than using original password.

Step 4: Copy the Password

7 of 7



Look at the following program to see how it works:

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"
    "net/smtp"
)
// smtpServer data to smtp server
type smtpServer struct {
    host string
    port string
}
// serverName URI to smtp server
func (s *smtpServer) serverName() string {
    return s.host + ":" + s.port
}
```

```

}

func main() {
    // Sender data.
    from := "sender email address"
    password := "password" // you can enter original password or password generated with App
    // Receiver email address.
    to := []string{
        "first receiver email address",
        //"second receiver email address",
    }
    // smtp server configuration.
    smtpServer := smtpServer{host: "smtp.gmail.com", port: "587"}
    // Message.
    message := []byte("Enter the message you want to send.")
    // Authentication.
    auth := smtp.PlainAuth("", from, password, smtpServer.host)
    // Sending email.
    err := smtp.SendMail("smtp.gmail.com:587", auth, from, to, message)
    if err != nil {
        fmt.Println(err)
        return
    }
    fmt.Println("Email Sent!")
}

```

Click the **RUN** button and wait for the terminal to start. Type `go run main.go` in the terminal and press **ENTER**.

Hurrah! You just sent your first email with Go! Cloud computing is so popular nowadays, and Go provides support for it. See the next lesson to see how Golang has made this possible.