

Techpuzzle 7 Solution
rloesch@

Enable the APIs required

```
gcloud services enable documentai.googleapis.com
gcloud services enable cloudfunctions.googleapis.com
...
```

Create Cloud Storage buckets

```
export PROJECT_ID=techpuzzle7
export BUCKET_LOCATION=us-central1
gsutil mb -c standard -l ${BUCKET_LOCATION} -b on \
  gs://${PROJECT_ID}-invoices
```

Create the Invoice Parser

The screenshot shows the Google Cloud Platform console for the project 'techpuzzle7'. On the left, the 'Document AI' menu is open, and the 'Processors' page is selected. The main area displays a grid of available processors. The 'Invoice Parser' is highlighted, showing its description: 'Extract text and values from invoices such as invoice number, supplier name, invoice amount, tax amount, invoice date, due date.' Below the grid, the 'Create processor' dialog is open. The 'Processor name' is set to 'techpuzzle7-invoice-parser'. The 'Region' is set to 'US (United States)'. The 'Encryption' section shows 'Google-managed encryption key' selected. The 'CREATE' button is visible at the bottom of the dialog.

Google Cloud Platform | techpuzzle7 | Search | Products, resources, docs (/)

Document AI | Create processor

Overview | Processors | Human-in-the-loop AI

Search processors

Processor	Description
1065 Parser	Extract from Form 1065, partnership name, address, etc.
1099-DIV Parser	Extract from Form 1099-DIV, including payer, recipient, etc.
1099-G Parser	Extract from Form 1099-G, including payer, recipient, etc.
1099-IN	Extract from Form 1099-IN, including payer, recipient, etc.
1099-R Parser	Extract from Form 1099-R, including payer, recipient, etc.
1120 Parser	Extract from Form 1120, partnership name, address, etc.
1120S Parser	Extract from Form 1120S, name, address, assets, etc.
Bank St	Extract from Form 1099-B, including payer, recipient, etc.
France Driver License Parser	Extract fields from France Driver License, including names and dates
France National ID Parser	Extract fields from France ID, including names, dates, etc.
France Passport Parser	Extract fields from France Passport, including names, dates
HOA St	Extract from Form 1099-B, including payer, recipient, etc.
Investment and Retirement Statement Parser	Extract fields from Form 1099-R, including payer, recipient, etc.
Invoice Parser	Extracts 30+ fields from Invoices: ID, amount, line items, etc.
Lending Doc Splitter/Classifier	Identify documents in a large file & classify known lending doc types
Mortgage	Extract from Form 1099-B, including payer, recipient, etc.

Create processor

Invoice Parser

Extract text and values from invoices such as invoice number, supplier name, invoice amount, tax amount, invoice date, due date. [Learn more](#)

Processor name *
techpuzzle7-invoice-parser

Must start with a letter. Can use letters, numbers, spaces, dashes, and underscores.

Region
US (United States)

This processor is encrypted with a Google-managed key by default. If you need to manage your encryption, you can use a customer-managed key instead.

Encryption

☒ Google-managed encryption key
No configuration required

☐ Customer-managed encryption key (CMEK)
Manage via Google Cloud Key Management Service

[SHOW LESS](#)

CREATE CANCEL

CLOUD SHELL

Terminal | techpuzzle7 | +

```
gs://$[PROJECT_ID]-input-invoices
creating gs://techpuzzle7-input-invoices/...
```

Test Invoice Parser

The screenshot displays the Google Cloud Platform (GCP) Document AI interface for an 'Invoice Parser analysis' document. The left sidebar shows the 'Processors' section with a list of fields and their corresponding values. The main area displays a preview of the invoice document, which includes a 'FROM' section for the Developer Company, a table of items with columns for Item, Hours, Rate, and Subtotal, and a 'Buyer Company' section. The bottom of the interface shows a terminal window with the command 'gs://[PROJECT_ID]-input-invoices' and the output 'Creating gs://techpuzzle7-input-invoices/...'.

Document AI Interface:

- Document:** Invoice Parser analysis
- Filter:** Type to filter
- Processors:**
- Human-in-the-loop AI:**

Fields and Values:

- supplier_name: Developer Company
- supplier_address: 123 Street, Beverly Hills
- invoice_id: 12-545678
- purchase_order: 12-545678
- receiver_tax_id: 12-545678
- invoice_date: 12/12/22
- supplier_phone: (123) 456-7890
- line_item/unit_price: 35.00
- line_item: 3 35.00 105.00 API Development
- line_item/descripti...: API Development
- line_item/quantity: 3
- line_item/amount: 105.00
- line_item: GUI Design 2 40.00 80.00
- line_item/descripti...: GUI Design
- line_item/amount: 80.00

Invoice Document Preview:

FROM

Developer Company
123 Street, Beverly Hills
developer@mail.com
(123) 456-7890

Invoice No.: 12-545678
Date: 12/12/22

Item	Hours	Rate	Subtotal
API Development	3	\$35.00	\$105.00
GUI Design	2	\$40.00	\$80.00
JavaScript Development	5	\$25.00	\$125.00
Total			\$310.00

Buyer Company
456 Street, Beverly Hills
California, 90210
buyer@mail.com
(123) 456-7890

Invoice Total
\$335.00

Review Completed Invoice Parser

The screenshot displays the Google Cloud Platform (GCP) Document AI interface. The top navigation bar shows 'Google Cloud Platform' and the project 'techpuzzle7'. The left sidebar contains a menu with 'Document AI', 'Overview', 'Processors', and 'Human-in-the-loop AI'. The main content area is titled 'techpuzzle7-invoice-parser' and includes a 'DISABLE PROCESSOR' button. Below the title are three tabs: 'OVERVIEW', 'MANAGE VERSIONS', and 'HUMAN-IN-THE-LOOP'. The 'OVERVIEW' tab is active, showing a table with processor details:

Name	techpuzzle7-invoice-parser
ID	9191950ebef6ca96
Status	Enabled
Processor Type	Invoice Parser
Encryption Type	Google-managed key

Below the table, the 'Prediction' section shows the 'Prediction endpoint' as <https://us-documentai.googleapis.com/v1/projects/872946784243/locations/us/processors/9191950ebef6ca96/process>. The 'Human Review' section indicates 'Not yet configured.' and features a blue button. The 'Test your processor' section lists supported formats (JPEG, JPG, PNG, WEBP, BMP, PDF, TIFF, TIF, GIF) and includes an 'UPLOAD TEST DOCUMENT' button. At the bottom, a 'CLOUD SHELL' terminal window is open, showing the command `gs://[PROJECT_ID]-input-invoices` and the output `Creating gs://techpuzzle7-input-invoices/...`.

Create Firestore and import the provided data

```
gcloud firestore import gs://techpuzzle7-firestore-import/2022-05-27T15:15:39_80829
```

←

→

↺

console.cloud.google.com/firestore/data/invoices/Invoice5?project=techpuzzle7

☆

Incognito

Google Cloud Platform

techpuzzle7

Search Products, resources, docs (/)

Firestore

Database

Indexes

Import/Export

Security Rules

Insights

Usage

Key Visualizer NEW

Data

Cloud Firestore in Native mode Database location: narn

/ > invoices > Invoice5

Root

+ START COLLECTION

invoices

invoices

+ ADD DOCUMENT

Invoice1

Invoice2

Invoice3

Invoice4

Invoice5

Invoice5

+ START COLLECTION

+ ADD FIELD

expected_amount: 335

invoice_id: "12-545678"

paid: false

Google Cloud Platform | techpuzzle7

Search Products, resources, docs (/)

Cloud Storage

Bucket details

techpuzzle7-firestore-import

Location: us-central1 (Iowa) | Storage class: Standard | Public access: Not public | Protection: None

OBJECTS | CONFIGURATION | PERMISSIONS | PROTECTION | LIFECYCLE

Buckets > techpuzzle7-firestore-import

UPLOAD FILES | UPLOAD FOLDER | CREATE FOLDER | MANAGE HOLDS | DOWNLOAD | DELETE

Filter by name prefix only | Filter | Filter objects and folders | Show deleted data

<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiration date
<input type="checkbox"/>	firestore_data.zip	2 KB	application/zip	Jun 10, 20...	Standard	Jun 10, 20...	Not public	—	Google-managed key	—

Cloud Shell | Terminal | (techpuzzle7) | Open Editor

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to techpuzzle7.

Write Cloud Function

```
package techpuzzle7

import (
    "context"
    "fmt"
    "io/ioutil"
    "math"
    "net/http"
    "reflect"
    "strconv"
    "strings"
    "time"

    documentai "cloud.google.com/go/documentai/apiv1"
    "cloud.google.com/go/firestore"
    "cloud.google.com/go/storage"
    "google.golang.org/api/iterator"
)
```

```

    documentaipb "google.golang.org/genproto/googleapis/cloud/documentai/v1"
)

var (
    storageClient *storage.Client
    firestoreClient *firestore.Client
)

const processor string =
"projects/872946784243/locations/us/processors/9191950ebef6ca96"
const projectID string = "techpuzzle7"

// GCSEvent is the payload of a GCS event.
type GCSEvent struct {
    Bucket      string    `json:"bucket"`
    Name        string    `json:"name"`
    Metageneration string  `json:"metageneration"`
    ResourceState string  `json:"resourceState"`
    TimeCreated time.Time `json:"timeCreated"`
    Updated     time.Time `json:"updated"`
}

type Invoice struct {
    Invoice_id string
    Amount     float64
    Paid       bool
}

// ProcessInvoice is executed when a file is uploaded to the Cloud Storage bucket you
// created for uploading invoices. It processes the invoice for text and updates the
// invoice data in Firebase.
func ProcessInvoice(ctx context.Context, event GCSEvent) error {

    // Read the file from GCS that triggered the event and get it as a byte array
    if event.Bucket == "" {
        return fmt.Errorf("empty file.Bucket")
    }
    if event.Name == "" {
        return fmt.Errorf("empty file.Name")
    }

    storageClient, err := storage.NewClient(ctx)

```

```

    if err != nil {
        return fmt.Errorf("failed to create client: %v", err)
    }
    bucket := storageClient.Bucket(event.Bucket)

    rc, err := bucket.Object(event.Name).NewReader(ctx)
    if err != nil {
        return fmt.Errorf("readFile: unable to open file from bucket %q, file %q: %v",
event.Bucket, event.Name, err)
    }
    defer rc.Close()
    data, err := ioutil.ReadAll(rc)
    if err != nil {
        return fmt.Errorf("readFile: unable to read data from bucket %q, file %q: %v",
event.Bucket, event.Name, err)
    }

    //Detect the mime type of the file
    mimeType := http.DetectContentType(data)

    //Invoke the document processor
    client, error := documentai.NewDocumentProcessorClient(ctx)
    if error != nil {
        return fmt.Errorf("failed to create new document processor client: %v", error)
    }
    defer client.Close()
    req := &documentaipb.ProcessRequest{
        Source: &documentaipb.ProcessRequest_RawDocument{
            RawDocument: &documentaipb.RawDocument{
                Content: data,
                MimeType: mimeType,
            },
        },
        Name: processor,
    }
    resp, err := client.ProcessDocument(ctx, req)
    if err != nil {
        return fmt.Errorf("failed to process response: %v", err)
    }

    // Parse through the detected fields on the invoice
    document := resp.GetDocument()

```

```

entities := document.GetEntities()
var inv Invoice
for _, entity := range entities {
    if entity.GetType() == "invoice_id" {
        inv.Invoice_id = entity.GetMentionText()
    }
    if entity.GetType() == "total_amount" {
        var strAmount string = entity.GetMentionText()
        strAmount = strings.Replace(strAmount, ",", "", -1)
        fAmount, err := getFloat(strAmount)
        if err == nil {
            inv.Amount = fAmount
        } else {
            return fmt.Errorf("failed to convert amount: %v", err)
        }
    }
}
fmt.Printf("Invoice ID: %s\n", inv.Invoice_id)
fmt.Printf("Total Amount: %.2f\n", inv.Amount)

// Get a Firestore client.
firestoreClient, err = firestore.NewClient(ctx, projectID)
if err != nil {
    return fmt.Errorf("failed to create firestore.NewClient: %v", err)
}
defer firestoreClient.Close()

//Look for the invoice in Firestore
iter := firestoreClient.Collection("invoices").Where("invoice_id", "=",
inv.Invoice_id).Documents(ctx)

for {
    dsnap, err := iter.Next()
    if err == iterator.Done {
        break
    }
    if err != nil {
        fmt.Printf("Invoice %s does not exist.\n", inv.Invoice_id)
        return fmt.Errorf("failed to convert amount: %v", err)
    }
}

```



```

    // Taking care of cases where the invoice is already paid or the amount doesn't
match the expected amount
    invoiceMap := dsnap.Data()
    var invData Invoice
    invData.Invoice_id = invoiceMap["invoice_id"].(string)
    iAmount := invoiceMap["expected_amount"]
    fAmount, err := getFloat(iAmount)
    if err == nil {
        inv.Amount = fAmount
    } else {
        return fmt.Errorf("failed to convert amount: %v", err)
    }

    invData.Amount = fAmount
    invData.Paid = invoiceMap["paid"].(bool)
    strPaid := "unpaid"
    if invData.Paid {
        strPaid = "paid"
        fmt.Printf("The invoice %s has already been paid!\n", invData.Invoice_id)
        // TODO: Implement duplicate invoice logic
    }

    fmt.Printf("Invoice %s has an expected amount of %.2f and is in status %s.\n",
invData.Invoice_id, invData.Amount, strPaid)

    if inv.Amount != invData.Amount {
        fmt.Printf("The paid amount %.2f does not match the expected amount
%.2f.\n", inv.Amount, invData.Amount)
        // TODO: Implement unexpected amount logic
    }

    // Update the Firestore record
    _, err = dsnap.Ref.Update(ctx, []firestore.Update{
        {
            Path: "paid",
            Value: true,
        }
    })

    if err != nil {
        return fmt.Errorf("failed to update invoice: %v", err)
    }
}

```

```

    return nil
}

// Helper function to convert amount to float where possible
var floatType = reflect.TypeOf(float64(0))
var stringType = reflect.TypeOf("")

func getFloat(unk interface{}) (float64, error) {
    switch i := unk.(type) {
    case float64:
        return i, nil
    case float32:
        return float64(i), nil
    case int64:
        return float64(i), nil
    case int32:
        return float64(i), nil
    case int:
        return float64(i), nil
    case uint64:
        return float64(i), nil
    case uint32:
        return float64(i), nil
    case uint:
        return float64(i), nil
    case string:
        return strconv.ParseFloat(i, 64)
    default:
        v := reflect.ValueOf(unk)
        v = reflect.Indirect(v)
        if v.Type().ConvertibleTo(floatType) {
            fv := v.Convert(floatType)
            return fv.Float(), nil
        } else if v.Type().ConvertibleTo(stringType) {
            sv := v.Convert(stringType)
            s := sv.String()
            return strconv.ParseFloat(s, 64)
        } else {
            return math.NaN(), fmt.Errorf("Can't convert %v to float64", v.Type())
        }
    }
}

```

```
}
```

Deploy Cloud Function

```
gcloud functions deploy process-invoice --runtime go116 --trigger-bucket techpuzzle7-invoices  
--entry-point ProcessInvoice
```

Google Cloud Platform

techpuzzle7

Search

Products, resources, docs (/)

50

Cloud Functions

Functions

CREATE FUNCTION

REFRESH

LEARN

RELEASE NOTES

Filter

Filter functions

	Environment	Name ↑	Region	Trigger	Runtime	Memory allocated	Executed function	Last deployed	Authentication	
	1st gen	process-invoice	us-central1	Bucket: techpuzzle7-invoices	Go 1.16	256 MB	ProcessInvoice	Jun 13, 2022, 2:51:58 PM		

Test File Upload and File Processing

```
vscode → /workspaces/Go Programming/techpuzzle7 $ ls -l ./samples/  
total 956
```

```
-rw-r--r-- 1 vscode vscode 27753 Jun 13 15:50 'Sample Invoice 1.pdf'  
-rw-r--r-- 1 vscode vscode 127861 Jun 13 15:51 'Sample Invoice 1.png'  
-rw-r--r-- 1 vscode vscode 45750 Jun 13 15:50 'Sample Invoice 2.pdf'  
-rw-r--r-- 1 vscode vscode 166775 Jun 13 15:51 'Sample Invoice 2.png'  
-rw-r--r-- 1 vscode vscode 25725 Jun 13 13:43 'Sample Invoice 3.pdf'  
-rw-r--r-- 1 vscode vscode 145461 Jun 13 15:51 'Sample Invoice 3.png'  
-rw-r--r-- 1 vscode vscode 118689 Jun 13 15:50 'Sample Invoice 4.pdf'  
-rw-r--r-- 1 vscode vscode 147516 Jun 13 15:51 'Sample Invoice 4.png'  
-rw-r--r-- 1 vscode vscode 17841 Jun 13 15:51 'Sample Invoice 5.pdf'  
-rw-r--r-- 1 vscode vscode 134351 Jun 13 15:51 'Sample Invoice 5.png'
```

```
vscode → /workspaces/Go Programming/techpuzzle7 $ gsutil -m cp ./samples/*.pdf  
gs://techpuzzle7-invoices
```

```
Copying file:///./samples/Sample Invoice 1.pdf [Content-Type=application/pdf]...  
Copying file:///./samples/Sample Invoice 2.pdf [Content-Type=application/pdf]...  
Copying file:///./samples/Sample Invoice 4.pdf [Content-Type=application/pdf]...  
Copying file:///./samples/Sample Invoice 3.pdf [Content-Type=application/pdf]...
```

Copying file:///samples/Sample Invoice 5.pdf [Content-Type=application/pdf]...
- [5/5 files][230.2 KiB/230.2 KiB] 100% Done
Operation completed over 5 objects/230.2 KiB.
vscode → /workspaces/Go Programming/techpuzzle7 \$

Google Cloud Platform techpuzzle7 Search Products, resources, docs (/)

Cloud Functions Function details EDIT DELETE COPY LEARN

process-invoice 1st gen Version 26, deployed at Jun 13, 2022, 2:51:58 P...

METRICS DETAILS SOURCE VARIABLES TRIGGER PERMISSIONS LOGS TESTING

Logs Showing 50 log entries Severity Default Filter Filter logs

2022-06-13T20:20:26.452155Z	process-invoice	451kvceb6pok	Invoice ID: 12-545678
2022-06-13T20:20:26.452180Z	process-invoice	451kvceb6pok	Total Amount: 335.00
2022-06-13T20:20:26.532593Z	process-invoice	451kvceb6pok	Invoice 12-545678 has an expected amount of 335.00 and is in status unpaid.
2022-06-13T20:20:26.542347Z	process-invoice	4quxotoniq78	Invoice 12-445678 has an expected amount of 266.09 and is in status unpaid.
2022-06-13T20:20:26.613296Z	process-invoice	451kvceb6pok	
2022-06-13T20:20:26.613317Z	process-invoice	451kvceb6pok	
2022-06-13T20:20:26.61422334Z	process-invoice	451kvceb6pok	Function execution took 4412 ms, finished with status: 'ok'
2022-06-13T20:20:26.625748Z	process-invoice	4quxotoniq78	
2022-06-13T20:20:26.625753Z	process-invoice	4quxotoniq78	
2022-06-13T20:20:26.62651709Z	process-invoice	4quxotoniq78	Function execution took 4181 ms, finished with status: 'ok'
2022-06-13T20:20:26.647951Z	process-invoice	2ag018b72ygm	Invoice ID: 12-145678
2022-06-13T20:20:26.647984Z	process-invoice	2ag018b72ygm	Total Amount: 443.48
2022-06-13T20:20:26.665826Z	process-invoice	smh6mh5t1mhu	Invoice ID: 12-345678
2022-06-13T20:20:26.665868Z	process-invoice	smh6mh5t1mhu	Total Amount: 2529.45
2022-06-13T20:20:26.759581Z	process-invoice	smh6mh5t1mhu	Invoice 12-345678 has an expected amount of 2529.45 and is in status unpaid.
2022-06-13T20:20:26.773701Z	process-invoice	2ag018b72ygm	Invoice 12-145678 has an expected amount of 443.48 and is in status unpaid.
2022-06-13T20:20:26.838847Z	process-invoice	2ag018b72ygm	
2022-06-13T20:20:26.839001Z	process-invoice	2ag018b72ygm	
2022-06-13T20:20:26.839692954Z	process-invoice	2ag018b72ygm	Function execution took 4431 ms, finished with status: 'ok'
2022-06-13T20:20:26.844848Z	process-invoice	smh6mh5t1mhu	
2022-06-13T20:20:26.844874Z	process-invoice	smh6mh5t1mhu	
2022-06-13T20:20:26.845614537Z	process-invoice	smh6mh5t1mhu	Function execution took 4270 ms, finished with status: 'ok'
2022-06-13T20:20:29.163208Z	process-invoice	4t91yodwa9jo	Invoice ID: 12-245678
2022-06-13T20:20:29.163238Z	process-invoice	4t91yodwa9jo	Total Amount: 231.74
2022-06-13T20:20:29.313490Z	process-invoice	4t91yodwa9jo	Invoice 12-245678 has an expected amount of 231.74 and is in status unpaid.
2022-06-13T20:20:29.413379Z	process-invoice	4t91yodwa9jo	
2022-06-13T20:20:29.413399Z	process-invoice	4t91yodwa9jo	
2022-06-13T20:20:29.413841777Z	process-invoice	4t91yodwa9jo	Function execution took 6337 ms, finished with status: 'ok'

Google Cloud Platform | techpuzzle7 | Search Products, resources, docs (/)

Firestore | **Data** | Cloud Firestore in Native mode | Database location: nam5

/ > invoices > Invoice5

Root	invoices	Invoice5
+ START COLLECTION	+ ADD DOCUMENT	+ START COLLECTION
<ul style="list-style-type: none"> invoices > 	<ul style="list-style-type: none"> Invoice1 Invoice2 Invoice3 Invoice4 Invoice5 > 	+ ADD FIELD
		<pre> expected_amount: 335 invoice_id: "12-545678" paid: true </pre>

Database: Data, Indexes, Import/Export, Security Rules

Insights: Usage, Key Visualizer **NEW**

Release Notes

Special Cases

Notes ...

Can you handle the case where a duplicate invoice is received?

What if the payment amount of the invoice doesn't match the expected amount?

What if we receive an invoice and don't know which order to apply it to?

```

for {
    dsnap, err := iter.Next()
    if err == iterator.Done {
        break
    }
}

```

```

    if err != nil {
        fmt.Printf("Invoice %s does not exist.\n", inv.Invoice_id)
        // TODO: Implement logic if invoice doesn't exist
        return fmt.Errorf("failed to convert amount: %v", err)
    }

    // Taking care of cases where the invoice is already paid or the amount doesn't
match the expected amount
    invoiceMap := dsnap.Data()
    var invData Invoice
    invData.Invoice_id = invoiceMap["invoice_id"].(string)
    iAmount := invoiceMap["expected_amount"]
    fAmount, err := getFloat(iAmount)
    if err == nil {
        inv.Amount = fAmount
    } else {
        return fmt.Errorf("failed to convert amount: %v", err)
    }

    invData.Amount = fAmount
    invData.Paid = invoiceMap["paid"].(bool)
    strPaid := "unpaid"
    if invData.Paid {
        strPaid = "paid"
        fmt.Printf("The invoice %s has already been paid!\n", invData.Invoice_id)
        // TODO: Implement duplicate invoice logic
    }

    fmt.Printf("Invoice %s has an expected amount of %.2f and is in status %s.\n",
invData.Invoice_id, invData.Amount, strPaid)

    if inv.Amount != invData.Amount {
        fmt.Printf("The paid amount %.2f does not match the expected amount
%.2f.\n", inv.Amount, invData.Amount)
        // TODO: Implement unexpected amount logic
    }

```

Useful Links

[Package cloud.google.com/go/documentai/apiv1 \(v1.4.0\)](https://pkg.go.dev/cloud.google.com/go/documentai/apiv1)

[Go-genproto documentai](#)

[Process documents by using client libraries](#)

[Invoice Parser](#)

[The Go Runtime](#)

[Golang-samples](#)

[Go googleapi](#)