

[Terminar o laboratório](#) 00:57:26

Cuidado: Quando estiver no console, siga as instruções do laboratório. Caso contrário, sua conta poderá ser bloqueada. [Saiba mais](#).

[Open Google Console](#)

Username 1
student-01-6985c845d8b1

Username 2
student-01-410c2d3b4c01

Password
u0hS7ERJrKG0

GCP Project ID
qwiklabs-gcp-03-d84fad

Crie um sistema resiliente e assíncrono com o Cloud Run e o Pub/Sub

1 hora Gratuito

- GSP650
- Visão geral
- Configuração e requisitos
- Arquitetura
- Criar o serviço de resultado do laboratório
- O serviço de e-mail
- O serviço de SMS
- Testar a resiliência do sistema
- Parabéns!
- [Terminar o laboratório](#)

0/100

GSP650



Visão geral

Nos laboratórios do workshop sem servidor do GCP: Quest da Pet Theory, você terá acesso a um cenário de negócios fictício e ajudará os personagens com o plano deles de migração sem servidor.

Há 12 anos, Lilian fundou a rede de clínicas veterinárias Pet Theory. Ao longo dos anos, o número de clínicas cresceu, levando a um aumento nas necessidades de automação. A maneira como a Pet Theory lida com os resultados dos exames médicos quando eles voltam do laboratório é lenta demais e propensa a erros, e Lilian quer melhorar esse processo.

Atualmente, Pedro, administrador de TI da Pet Theory, processa os resultados dos exames manualmente. Sempre que recebe um resultado de exame, ele escreve e envia um e-mail ao cliente cujo animal de estimação fez o exame, em seguida, digita uma mensagem de texto no celular e manda os resultados para o cliente em formato de texto.

Pedro está trabalhando com Raquel, consultora de software, para elaborar um sistema com maior escalabilidade. Eles querem criar uma solução que não exija muita manutenção contínua. Pedro e Raquel optaram pela tecnologia sem servidor.

Pré-requisitos

Para fazer este laboratório, é preciso saber usar o Console do GCP e ambientes shell. Este laboratório faz parte da série [Cloud](#). Consulte as [Instruções de laboratório](#) para obter mais informações.

laboratório faz parte de uma série. Concluir os laboratórios anteriores será útil, mas não é obrigatório:

- [Como migrar dados para um banco de dados do Firestore](#)
- [Crie um app da Web sem servidor com o Firebase e o Firestore](#)
- [Crie um app sem servidor que gera arquivos PDF com o Cloud Run](#)

Você também precisa estar familiarizado com a edição de arquivos. Use o editor de texto da sua preferência (como `nano`, `vi` etc.) ou inicie o editor de código do Cloud Shell, disponível na barra superior:



Configuração e requisitos

Antes de clicar no botão Start Lab

Leia estas instruções. Os laboratórios são cronometrados e não podem ser pausados. O timer é iniciado quando você clica em **Começar o laboratório** e mostra por quanto tempo os recursos do Google Cloud ficarão disponíveis.

Este laboratório prático do Qwiklabs permite que você realize as atividades em um ambiente real de nuvem, não em uma simulação ou demonstração. Você receberá novas credenciais temporárias para fazer login e acessar o Google Cloud durante o laboratório.

O que é necessário

Para fazer este laboratório, você precisa ter:

- acesso a um navegador de Internet padrão (recomendamos o Chrome);
- tempo para concluir as atividades.

Observação: não use seu projeto ou sua conta do Google Cloud neste laboratório.

Observação: se estiver usando um dispositivo Chrome OS, abra uma janela anônima para executar o laboratório.

Como iniciar seu laboratório e fazer login no console do Google Cloud

1. Clique no botão **Começar o laboratório**. Se for preciso pagar, você verá um pop-up para selecionar a forma de pagamento. No painel **Detalhes do laboratório** à esquerda, você verá o seguinte:

- O botão **Abrir Console do Cloud**
- Tempo restante
- As credenciais temporárias que você vai usar neste laboratório
- Outras informações se forem necessárias

2. Clique em **Abrir Console do Google**. O laboratório ativa recursos e depois abre outra guia com a página **Fazer login**.

Dica: coloque as guias em janelas separadas lado a lado.

Observação: se aparecer a caixa de diálogo **Escolher uma conta**, clique em **Usar outra conta**.

3. Caso seja preciso, copie o **Nome de usuário** no painel **Detalhes do laboratório** e cole esse nome na caixa de diálogo **Fazer login**. Clique em **Avançar**.

4. Copie a **Senha** no painel **Detalhes do laboratório** e cole na caixa de diálogo **Olá**. Clique em **Avançar**.

Importante: você precisa usar as credenciais do painel à esquerda. Não use suas credenciais do Google Cloud Ensina.

Observação: se você usar sua própria conta do Google Cloud neste laboratório, é possível que receba cobranças adicionais.

5. Acesse as próximas páginas:

- Aceite os Termos e Condições.
- Não adicione opções de recuperação nem autenticação de dois fatores (porque essa é uma conta temporária).
- Não se inscreva em testes gratuitos.

Depois de alguns instantes, o console do GCP vai ser aberto nesta guia.

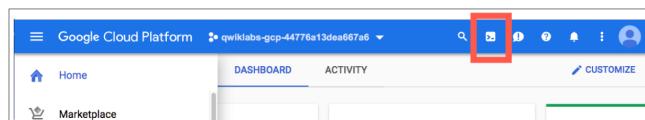
Observação: para ver uma lista dos produtos e serviços do Google Cloud, clique no **Menu de navegação** no canto superior esquerdo.



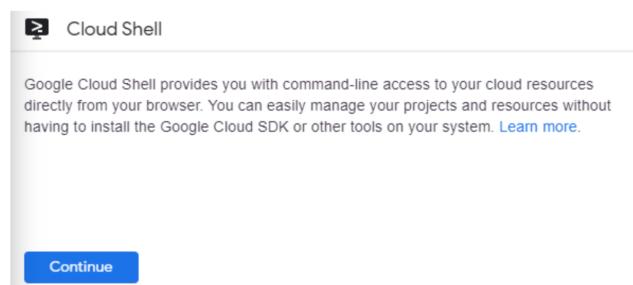
Ative o Google Cloud Shell

O Google Cloud Shell é uma máquina virtual com ferramentas de desenvolvimento. Ele conta com um diretório principal permanente de 5 GB e é executado no Google Cloud. O Google Cloud Shell permite acesso de linha de comando aos seus recursos do GCP.

1. No Console do GCP na barra de ferramentas superior direita, clique no botão **Abrir o Cloud Shell**.



2. Clique em **Continue** (continuar):



Demora alguns minutos para provisionar e conectar-se ao ambiente. Quando você está conectado, você já está autenticado e o projeto é definido como seu **PROJECT_ID**. Por exemplo:



gcloud é a ferramenta de linha de comando do Google Cloud Platform. Ele vem pré-instalado no Cloud Shell e aceita preenchimento com tabulação.

É possível listar o nome da conta ativa com este comando:

```
gcloud auth list
```

Saída:

```
ACTIVE: *
ACCOUNT: student-01-xxxxxxxxxx@qwiklabs.net
To set the active account, run:
$ gcloud config set account `ACCOUNT`
```

É possível listar o ID de projeto com este comando:

```
gcloud config list project
```

Saída:

```
[core]
project = <project_ID>
```

Exemplo de saída:

```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

A documentação completa do **gcloud** está disponível na página [Visão geral do gcloud](#) do Google Cloud.

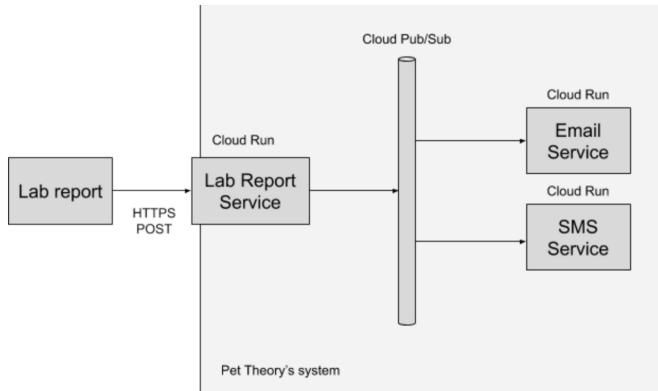
Os responsáveis pela Pet Theory querem automatizar o envio dos resultados dos exames aos clientes. Eles passaram por maus bocados para acompanhar o aumento no volume de consultas, o que levou Lilian a pedir ajuda a Raquel...

 <i>Lilian, fundadora da Pet Theory</i>	Oi, Raquel Obrigada por organizar o portal de seguros. Será que podemos fazer algo em relação aos resultados dos exames? Precisamos de uma maneira mais eficiente de enviar os resultados para os clientes. Lilian
 <i>Raquel, consultora de software</i>	Olá, Lilian. Claro, verei o que é possível fazer. Tenho algumas ideias para melhorar a situação. Raquel

Arquitetura

Na Pet Theory, os exames médicos são realizados por um laboratório externo. Depois de concluir o exame, o laboratório envia os resultados de volta para a Pet Theory.

Para enviar os resultados, o laboratório usa um POST de HTTP(s) para o endpoint da Web da Pet Theory. A ilustração abaixo descreve a arquitetura geral.



Depois de analisar o processo geral, Raquel acredita que seja possível projetar um sistema em que a Pet Theory possa:

1. receber a solicitação do POST de HTTP e enviar ao laboratório uma confirmação de recebimento;
2. enviar um e-mail ao cliente com o resultado;
3. enviar uma mensagem de texto (SMS) e um e-mail ao cliente com o resultado.

O projeto de Raquel isola cada uma das atividades acima e exige:

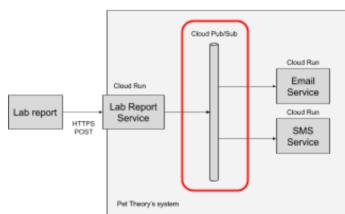
- um serviço para enviar as solicitações e as respostas relacionadas aos resultados;
- um serviço para enviar por e-mail os resultados dos exames para o cliente;
- um serviço para enviar uma mensagem de texto (SMS) ao cliente;
- o uso do Pub/Sub na comunicação entre serviços;
- o uso de uma infraestrutura sem servidor na arquitetura do aplicativo.

Para desenvolver um código mais fácil de escrever e com menos bugs, Raquel lança mão de funções de uso único.

 <i>Raquel, consultora de software</i>	<p>Oi, Pedro.</p> <p>Lilian quer que eu crie um protótipo para ajudar no processamento dos prontuários.</p> <p>Para começar, será que você pode configurar um tópico do Pub/Sub chamado <code>new-lab-report</code>?</p> <p>Raquel</p>
 <i>Pedro, administrador de TI</i>	<p>Olá, Raquel.</p> <p>Parece ser um projeto muito interessante. Consigo fazer isso para você ainda esta manhã. A configuração das duas atividades no GCP é muito rápida.</p> <p>Pedro</p>

Crie um tópico do Pub/Sub

Ajude Pedro a criar um tópico do Pub/Sub chamado `new-lab-report`.



Quando um serviço publica uma mensagem do Pub/Sub, ela deve ser marcada com um tópico. O serviço que será criado publica uma mensagem para cada resultado de exame encontrado.

Primeiramente, você precisa criar um tópico para essa tarefa.

Execute o seguinte comando para criar um tópico do Pub/Sub:

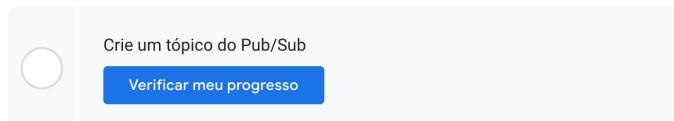
```
gcloud pubsub topics create new-lab-report
```

Qualquer serviço inscrito no tópico "new-lab-report" conseguirá processar a mensagem publicada pelo serviço de resultado. No diagrama acima, você verá dois desses processadores, o serviço de e-mail e o serviço de SMS.

Em seguida, ative o Cloud Run, que executará seu código na nuvem:

```
gcloud services enable run.googleapis.com
```

Clique em **Verificar meu progresso** para conferir o objetivo.

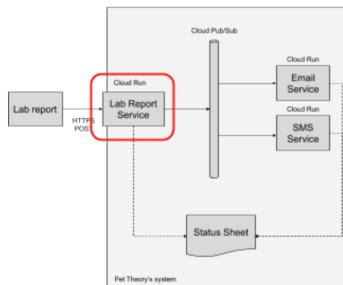


Não se esqueça de dizer a Raquel que o tópico do Pub/Sub está pronto para ela usar.

 <i>Pedro, administrador de TI</i>	<p>Olá, Raquel.</p> <p>Tudo pronto.</p> <p>Se você tiver tempo, eu gostaria de ver como este protótipo é preparado. Podemos trabalhar nisso juntos?</p> <p>Pedro</p>
 <i>Raquel, consultora de software</i>	<p>Oi, Pedro.</p> <p>Ótimo. Obrigada por fazer isso tão rápido. Vou marcar um horário e começaremos a trabalhar nisso.</p> <p>Raquel</p>

Criar o serviço de resultado do laboratório

Ajude Raquel a configurar um novo serviço de resultados.



Este serviço será usado para prototipagem; dessa forma, ele fará somente duas coisas:

1. Receber o POST de HTTPS do serviço contendo os dados do resultado.
2. Publicar uma mensagem no Pub/Sub.

Adicionar código para o Lab Report Service

No Cloud Shell, clone o repositório necessário para este laboratório:

```
git clone https://github.com/rosera/pet-theory.git
```

Acesse o diretório `lab-service`:

```
cd pet-theory/lab05/lab-service
```

Instale os pacotes a seguir, que serão necessários para receber as solicitações de HTTPS de entrada e publicar no Pub/Sub:

```
npm install express
npm install body-parser
npm install @google-cloud/pubsub
```

Esses comandos atualizam o `package.json` do arquivo para indicar as dependências necessárias para o serviço.

Agora você editará o arquivo `package.json` de modo que o Cloud Run saiba como iniciar seu código.

Abra o arquivo `package.json`.

Na seção "scripts", adicione a linha `"start": "node index.js"` mostrada abaixo e salve o arquivo.

```
"scripts": {
  "start": "node index.js",
  "test": "echo \\"$Error: no test specified\\" && exit 1"
},
```

Crie um arquivo com o nome `index.js` e adicione este código a ele:

```
const {PubSub} = require('@google-cloud/pubsub');
const pubsub = new PubSub();
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
app.use(bodyParser.json());
const port = process.env.PORT || 8080;
app.listen(port, () => {
  console.log('Listening on port', port);
});
app.post('/', async (req, res) => {
  try {
    const labReport = req.body;
    await publishPubSubMessage(labReport);
    res.status(204).send();
  }
  catch (ex) {
    console.log(ex);
    res.status(500).send(ex);
  }
})
async function publishPubSubMessage(labReport) {
  const buffer = Buffer.from(JSON.stringify(labReport));
  await pubsub.topic('new-lab-report').publish(buffer);
}
```

A parte mais importante do código está nesta seção:

```
const labReport = req.body;
await publishPubSubMessage(labReport);
```

Essas duas linhas fazem o principal trabalho do serviço:

1. Extrair o resultado do laboratório da solicitação de POST.
2. Publicar uma mensagem do Pub/Sub contendo o resultado recém-publicado.

Agora crie um arquivo com o nome `Dockerfile` e adicione o código abaixo a ele:

```
FROM node:10
```

```
WORKDIR /usr/src/app
COPY package.json package*.json .
RUN npm install --only=production
COPY . .
CMD [ "npm", "start" ]
```

Esse arquivo define como criar um pacote do serviço Cloud Run em um contêiner.

Implante o lab-report-service

Crie um script com o nome `deploy.sh` e cole estes comandos nele:

```
gcloud builds submit \
--tag gcr.io/$GOOGLE_CLOUD_PROJECT/lab-report-service
gcloud run deploy lab-report-service \
--image gcr.io/$GOOGLE_CLOUD_PROJECT/lab-report-service \
--platform managed \
--region us-east1 \
--allow-unauthenticated \
--max-instances=1
```

Execute o seguinte comando para tornar o arquivo executável:

```
chmod u+x deploy.sh
```

Chegou a hora de implantar o serviço de resultados do laboratório. Execute o script de implantação:

```
./deploy.sh
```

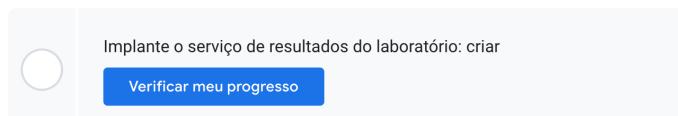
Devido a problemas de sincronização, talvez apareça uma mensagem de erro na primeira vez que você executar este comando. Se acontecer isso, basta executar novamente `deploy.sh`.

Depois que a implantação for concluída, você verá uma mensagem semelhante a esta:

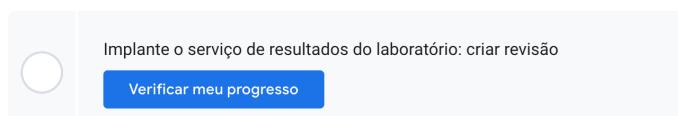
```
Service [lab-report-service] revision [lab-report-service-00001] has been
deployed and is serving traffic at https://lab-report-service-
[hash].a.run.app
```

Bom trabalho. O serviço de resultados do laboratório foi implantado e processará os resultados por HTTP. Agora você pode testar se o novo serviço está instalado e funcionando.

Clique em **Verificar meu progresso** para conferir o objetivo.



Clique em **Verificar meu progresso** para conferir o objetivo.



Teste o serviço de resultados do laboratório

Para validar o serviço de resultados, simule três POSTs de HTTPS criados pelo laboratório, cada um contendo um resultado. Para fins de teste, os resultados criados conterão somente um ID.

Primeiramente, para facilitar o trabalho, coloque o URL do resultado em uma variável de ambiente.

```
export LAB_REPORT_SERVICE_URL=$(gcloud run services describe lab-report-
service --platform managed --region us-east1 --
```

```
format="value(status.address.url)")
```

Confirme que LAB_REPORT_SERVICE_URL foi capturada:

```
echo $LAB_REPORT_SERVICE_URL
```

Crie um novo arquivo com o nome post-reports.sh e adicione o código abaixo a ele:

```
curl -X POST \
-H "Content-Type: application/json" \
-d "{\"id\": 12}" \
$LAB_REPORT_SERVICE_URL &
curl -X POST \
-H "Content-Type: application/json" \
-d "{\"id\": 34}" \
$LAB_REPORT_SERVICE_URL &
curl -X POST \
-H "Content-Type: application/json" \
-d "{\"id\": 56}" \
$LAB_REPORT_SERVICE_URL &
```

O script acima usará o comando curl para postar três IDs distintos para o URL do serviço do laboratório. Cada comando será executado individualmente em segundo plano.

Torne o script post-reports.sh executável:

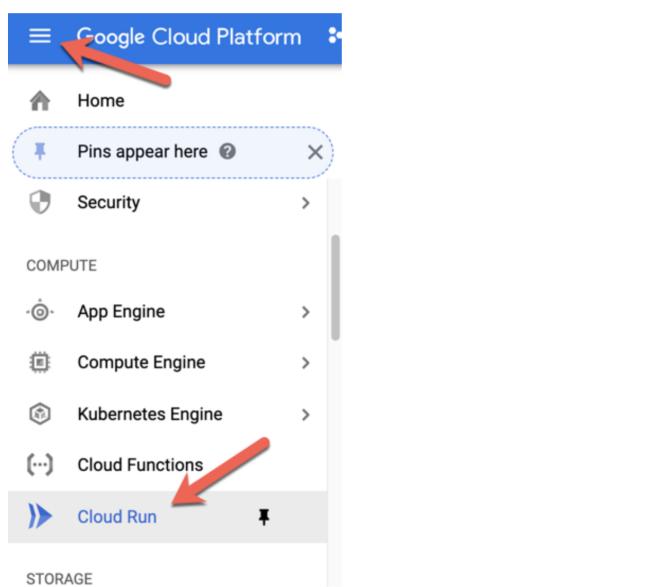
```
chmod u+x post-reports.sh
```

Para testar o endpoint do serviço, você deve postar nele três resultados usando o script descrito acima:

```
./post-reports.sh
```

Esse script postou três resultados no serviço. Verifique os registros para ver os resultados.

No **Menu de navegação**, clique em **Cloud Run**.



Agora você verá o **lab-report-service** recém implantado. Clique nele.

The screenshot shows the 'Cloud Run' service list. It displays a single service named 'lab-report-service'. The table includes columns for Name, Region, Authentication, Connectivity, Last deployed, and Deployed by. A red arrow points to the 'Name' column of the 'lab-report-service' row.

Name	Region	Authentication	Connectivity	Last deployed	Deployed by
lab-report-service	us-central1	Allow	External	Mar 1, 2020, 12:22:31 PM	student-01-5e2d0e1b85e@qwiklabs.net

A página seguinte mostra os detalhes do lab-report-service. Clique na guia **Registros**.

The screenshot shows the detailed view of the 'lab-report-service'. The top navigation bar has tabs for 'Registers', 'Services', 'CREATE SERVICE', 'DELETE', and 'SHOW INFO PANEL'. The 'Registers' tab is highlighted with a red arrow. Below the tabs, there's a table with columns for 'Key/Sec' and 'Value'.

The screenshot shows the 'Service details' page for a service named 'lab-report-service'. The 'Logs' tab is highlighted with a red arrow. Below the tabs, there are time filters: '1 hour', '6 hours', '1 day' (which is selected), '7 days', and '30 days'.

Na página Registros, estão os resultados dos três resultados de teste que você acabou de postar com o script. Esperamos que os códigos de retorno do HTTP sejam 204, o que significa OK - sem conteúdo, mostrado abaixo. Se você não vir nenhuma entrada, tente rolar para cima e para baixo usando a barra de rolagem à direita. Essa ação faz com que o registro seja recarregado.

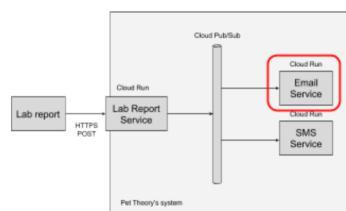
The screenshot shows the 'Logs' section of the Cloud Run service details. It displays 15 messages, all of which are POST requests to the service. The third message is circled in red, and a red arrow points to the 'Try scrolling' button at the bottom right of the log viewer.

Time	Method	Duration	URL
2020-03-01T20:28:12.199582Z	POST	284 ms	curl/7.52.1 https://lab-report-service-gcp0h24qja-uc.a.run.app/
2020-03-01T20:28:12.315562Z	POST	284 ms	curl/7.52.1 https://lab-report-service-gcp0h24qja-uc.a.run.app/
2020-03-01T20:28:12.439682Z	POST	284 ms	curl/7.52.1 https://lab-report-service-gcp0h24qja-uc.a.run.app/

A próxima tarefa é escrever os serviços de SMS e de e-mail. Esses serviços serão acionados quando o serviço de resultados publicar uma mensagem do Pub/Sub no tópico "new-lab-report".

O serviço de e-mail

Ajude Raquel a configurar o novo serviço de e-mail.



Adicione o código do serviço de e-mail

Acesse o diretório do serviço de e-mail:

```
cd ~/pet-theory/lab05/email-service
```

Instale estes pacotes para que o código processe as solicitações de HTTPS de entrada.

```
npm install express
npm install body-parser
```

O comando acima atualizará o arquivo `package.json`, que descreve o app e as dependências dele. Adicione a instrução `start` para que o Cloud Run saiba como executar o código.

Abra o arquivo `package.json`.

Na seção "scripts", adicione a linha `"start": "node index.js"` mostrada abaixo e salve o arquivo.

```
"scripts": {
  "start": "node index.js",
  "test": "echo \"Error: no test specified\" && exit 1"
},
```

Crie um arquivo chamado `index.js` e adicione o seguinte a ele:

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
app.use(bodyParser.json());
const port = process.env.PORT || 8080;
app.listen(port, () => {
  console.log('Listening on port', port);
});
app.post('/', async (req, res) => {
  const labReport = decodeBase64Json(req.body.message.data);
  try {
    console.log(`Email Service: Report ${labReport.id} trying...`);
    sendEmail();
    console.log(`Email Service: Report ${labReport.id} success :-)`);
    res.status(204).send();
  }
  catch (ex) {
    console.log(`Email Service: Report ${labReport.id} failure: ${ex}`);
    res.status(500).send();
  }
})
function decodeBase64Json(data) {
  return JSON.parse(Buffer.from(data, 'base64').toString());
}
function sendEmail() {
  console.log('Sending email');
}
```

Esse código será executado quando o Pub/Sub postar uma mensagem no serviço. Veja o que ele faz:

- Ele decodifica a mensagem do Pub/Sub e tenta chamar a função `sendEmail()`.
- Se isso funcionar e nenhuma exceção for gerada, ele retornará o código de status 204 para que o Pub/Sub saiba que a mensagem foi processada.
- Se houver uma exceção, o serviço retornará o código de status 500 para que o Pub/Sub saiba que a mensagem não foi processada e que, posteriormente, ela será postada novamente para o serviço.

Quando a comunicação entre os serviços estiver funcionando, Raquel adicionará o código à função `sendEmail()` para enviar o e-mail.

Agora crie um arquivo com o nome Dockerfile e adicione o código abaixo a ele:

```
FROM node:10
WORKDIR /usr/src/app
COPY package.json package*.json .
RUN npm install --only=production
COPY . .
CMD [ "npm", "start" ]
```

Esse arquivo define como criar um pacote do serviço Cloud Run em um contêiner.

Implante o serviço de e-mail

Crie um arquivo chamado `deploy.sh` e adicione o seguinte a ele:

```
gcloud builds submit \
  --tag gcr.io/$GOOGLE_CLOUD_PROJECT/email-service
gcloud run deploy email-service \
  --image gcr.io/$GOOGLE_CLOUD_PROJECT/email-service \
  --platform managed \
  --region us-central1 \
  --no-allow-unauthenticated \
  --max-instances=1
```

Torne `deploy.sh` executável:

```
chmod u+x deploy.sh
```

Implante o serviço de e-mail

```
./deploy.sh
```

Quando a implantação estiver concluída, você verá uma mensagem parecida com esta:

```
Service [email-service] revision [email-service-00001] has been deployed  
and is serving traffic at https://email-service-[hash].a.run.app
```

O serviço foi implantado. Agora você precisa garantir que o serviço de e-mail seja acionado quando houver uma mensagem do Pub/Sub disponível.

Clique em **Verificar meu progresso** para conferir o objetivo.

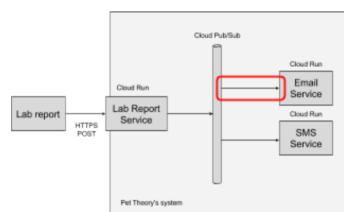


Clique em **Verificar meu progresso** para conferir o objetivo.



Configure o Pub/Sub para acionar o serviço de e-mail

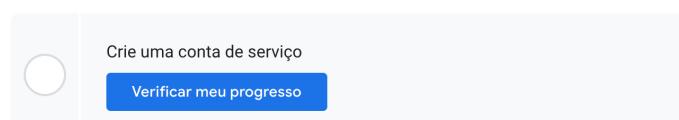
Sempre que uma nova mensagem do Pub/Sub for publicada com o tópico "new-lab-report", ela deve acionar o serviço de e-mail. Para realizar essa tarefa, configure uma conta de serviço para tratar automaticamente das solicitações associadas ao serviço.



Crie uma nova conta de serviço que será usada para acionar os serviços que respondem a mensagens do Pub/Sub:

```
gcloud iam service-accounts create pubsub-cloud-run-invoker --display-name "PubSub Cloud Run Invoker"
```

Clique em **Verificar meu progresso** para conferir o objetivo.



Dê à nova conta de serviço permissão para invocar o serviço de e-mail:

```
gcloud run services add-iam-policy-binding email-service --member=serviceAccount:pubsub-cloud-run-invoker@google.cLOUD_PROJECT.iam.gserviceaccount.com --role=roles/run.invoker --region us-central1 --platform managed
```

A seguir, faça com que o Pub/Sub invoque o serviço de SMS quando for publicada uma mensagem do "new-lab-report".

Coloque o número do projeto em uma variável de ambiente para facilitar o acesso:

```
PROJECT_NUMBER=$(gcloud projects list --filter="qwiklabs-gcp" --format='value(PROJECT_NUMBER)')
```

A seguir, ative o projeto para criar tokens de autenticação do Pub/Sub.

Execute o código abaixo:

```
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT --member=serviceAccount:pubsub-cloud-run-invoker@google.cLOUD_PROJECT.iam.gserviceaccount.com --role=roles/run.invoker --region us-central1 --platform managed
```

```
member=serviceAccount:service-$PROJECT_NUMBER@gcp-sa-
pubsub.iam.gserviceaccount.com --
role=roles/iam.serviceAccountTokenCreator
```

Coloque o URL do serviço de e-mail em outra variável de ambiente:

```
EMAIL_SERVICE_URL=$(gcloud run services describe email-service --platform
managed --region us-east1 --format="value(status.address.url)")
```

Confirme que EMAIL_SERVICE_URL foi capturada:

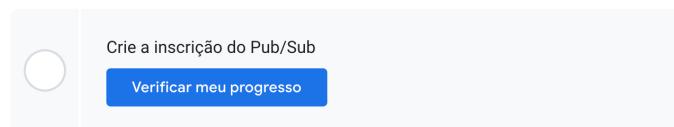
```
echo $EMAIL_SERVICE_URL
```

Crie uma inscrição do Pub/Sub para o serviço de e-mail.

```
gcloud pubsub subscriptions create email-service-sub --topic new-lab-
report --push-endpoint=$EMAIL_SERVICE_URL --push-auth-service-
account=pubsub-cloud-run-
invoker@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com
```

Bom trabalho. O serviço agora está configurado para responder a mensagens do Cloud Pub/Sub. A etapa seguinte é validar o código para confirmar que ele atende aos requisitos.

Clique em **Verificar meu progresso** para conferir o objetivo.



Teste o serviço de resultados do laboratório e o serviço de e-mail juntos

Usando o script criado anteriormente, poste os resultados do laboratório novamente:

```
~/pet-theory/lab05/lab-service/post-reports.sh
```

Em seguida, abra o registro (**Menu de navegação > Cloud Run**). Você verá os dois serviços do Cloud Run na sua conta.

Name	Region	Authentication	Connectivity	Last deployed	Deployed by
email-service	us-central1	External	External	Mar 1, 2020, 12:48:16 PM	student01@se27se1t45e@qwiklabs.net
lab-report-service	us-central1	Allow unauthenticated	External	Mar 1, 2020, 12:22:31 PM	student01@se27se1t45e@qwiklabs.net

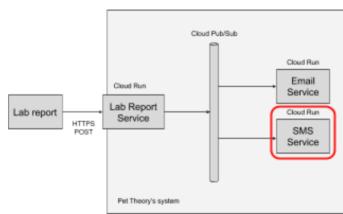
Clique em **email-service** e depois clique em **Registros**. Você verá o resultado do serviço ser acionado pelo Pub/Sub. Se você não vir as mensagens que espera, role para cima e para baixo com a barra de rolagem para atualizar o registro.

Logs	Showing 15 messages	Any log level	Filter
2020-01-01T00:00:00Z Email Service: Report 56 success ->			
2020-01-01T00:00:00Z Email Service: Report 12 success ->			
2020-01-01T00:00:00Z Email Service: Report 12 success ->			

Bom trabalho! Agora o serviço de e-mail é capaz de gravar as informações no registro sempre que uma mensagem é processada na fila do tópico do Cloud Pub/Sub. A última tarefa é criar o serviço de SMS.

O serviço de SMS

Ajude Raquel a configurar o novo serviço de SMS.



Adicione o código do serviço de SMS

Crie um diretório para o serviço de SMS:

```
cd ~/pet-theory/lab05/sms-service
```

Instale os pacotes necessários para receber as solicitações de HTTPS de entrada:

```
npm install express
npm install body-parser
```

Abra o arquivo `package.json`.

Na seção "scripts", adicione a linha `"start": "node index.js"` mostrada abaixo e salve o arquivo.

```
...
"scripts": {
  "start": "node index.js",
  "test": "echo \"Error: no test specified\" && exit 1"
},
...
```

Crie um arquivo chamado `index.js` e adicione o seguinte a ele:

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
app.use(bodyParser.json());
const port = process.env.PORT || 8080;
app.listen(port, () => {
  console.log('Listening on port', port);
});
app.post('/', async (req, res) => {
  const labReport = decodeBase64Json(req.body.message.data);
  try {
    console.log(`SMS Service: Report ${labReport.id} trying...`);
    sendSms();
    console.log(`SMS Service: Report ${labReport.id} success :-)`);
    res.status(204).send();
  }
  catch (ex) {
    console.log(`SMS Service: Report ${labReport.id} failure: ${ex}`);
    res.status(500).send();
  }
})
function decodeBase64Json(data) {
  return JSON.parse(Buffer.from(data, 'base64').toString());
}
function sendSms() {
  console.log('Sending SMS');
}
```

Agora crie um arquivo com o nome `Dockerfile` e adicione o código abaixo a ele:

```
FROM node:10
WORKDIR /usr/src/app
COPY package.json package*.json .
RUN npm install --only=production
COPY
```

```
COPY . .
CMD [ "npm", "start" ]
```

Esse arquivo define como criar um pacote do serviço Cloud Run em um contêiner. Agora que o código foi criado, o próximo passo é implantar o serviço.

Implante o serviço de SMS

Crie um arquivo com o nome `deploy.sh` e adicione este código a ele:

```
gcloud builds submit \
  --tag gcr.io/$GOOGLE_CLOUD_PROJECT/sms-service
gcloud run deploy sms-service \
  --image gcr.io/$GOOGLE_CLOUD_PROJECT/sms-service \
  --platform managed \
  --region us-east1 \
  --no-allow-unauthenticated \
  --max-instances=1
```

Torne `deploy.sh` executável:

```
chmod u+x deploy.sh
```

Implante o serviço de SMS:

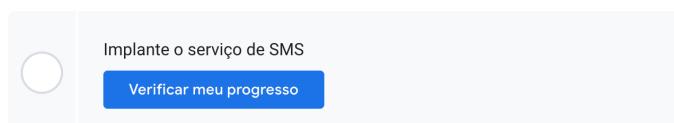
```
./deploy.sh
```

Quando a implantação estiver concluída, uma mensagem semelhante a esta será exibida:

```
Service [sms-service] revision [sms-service-0001] has been deployed and
is serving traffic at https://sms-service-[hash].a.run.app
```

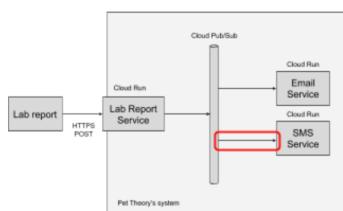
O serviço SMS está implantado, mas não está vinculado ao serviço do Cloud Pub/Sub. Corria isso na seção seguinte.

Clique em **Verificar meu progresso** para conferir o objetivo.



Configure o Cloud Pub/Sub para acionar o serviço de SMS

Como acontece no serviço de e-mail, é preciso configurar o link entre o Cloud Pub/Sub e o serviço de SMS para processar as mensagens.



Configure as permissões para fazer com que o Pub/Sub acione o serviço de SMS:

```
gcloud run services add-iam-policy-binding sms-service --
member=serviceAccount:pubsub-cloud-run-
invoker@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com --
role=roles/run.invoker --region us-central1 --platform managed
```

A seguir, faça com que o Pub/Sub invoque o serviço de SMS quando for publicada uma mensagem do "new-lab-report".

A primeira etapa é colocar o endereço do URL do serviço de SMS em uma variável de ambiente:

```
SMS_SERVICE_URL=$(gcloud run services describe sms-service --platform managed --region us-east1 --format="value(status.address.url)")
```

Confirme que SMS_SERVICE_URL foi capturada:

```
echo $SMS_SERVICE_URL
```

Em seguida, crie a inscrição do Pub/Sub:

```
gcloud pubsub subscriptions create sms-service-sub --topic new-lab-report --push-endpoint=$SMS_SERVICE_URL --push-auth-service-account=pubsub-cloud-run-invoker@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com
```

Execute novamente o script de teste para postar três resultados no serviço de resultados do laboratório:

```
~/pet-theory/lab05/lab-service/post-reports.sh
```

Em seguida, abra o registro (**Menu de navegação > Cloud Run**). Você verá os três serviços do Cloud Run na sua conta.

Name	Req/sec	Region	Authentication	Connectivity	Last deployed	Deployed by
email-service	0	us-central1		External	Mar 1, 2020, 3:17:41 PM	student-01-0ef80babe773@qwiklabs.net
lab-report-service	0	us-central1	Allow unauthenticated	External	Mar 1, 2020, 3:10:31 PM	student-01-0ef80babe773@qwiklabs.net
sms-service	0	us-central1		External	Mar 1, 2020, 3:27:15 PM	student-01-0ef80babe773@qwiklabs.net

Clique em **sms-service**, em seguida, clique em **Registros**. Você verá o resultado do serviço ser acionado pelo Pub/Sub.

O sistema de protótipo foi criado e testado com sucesso. Entretanto, Pedro está receoso porque não houve teste de resiliência no processo de validação inicial.

Testar a resiliência do sistema

O que acontecerá se um dos serviços cair? Pedro já se deparou com isso antes, já que é uma situação comum.

Ajude Raquel a preparar o sistema para lidar com essa situação. Ela quer testar o que acontece quando ocorre um erro em um serviço devido à implantação de uma versão incorreta do serviço de e-mail.

Volte para o diretório `email-service`:

```
cd ~/pet-theory/lab05/email-service
```

Adicione um texto inválido ao aplicativo de serviço de e-mail para provocar um erro.

Edita `index.js` e adicione a linha `throw` à função `sendEmail()`, como mostrado abaixo. Isso vai gerar uma exceção, como se o servidor de e-mail estivesse inativo:

```
...
function sendEmail() {
  throw 'Email server is down';
  console.log('Sending email');
}
...
```

A adição desse código provocará um erro no serviço quando ele for invocado.

Implemente esta versão incorreta do serviço de e-mail.

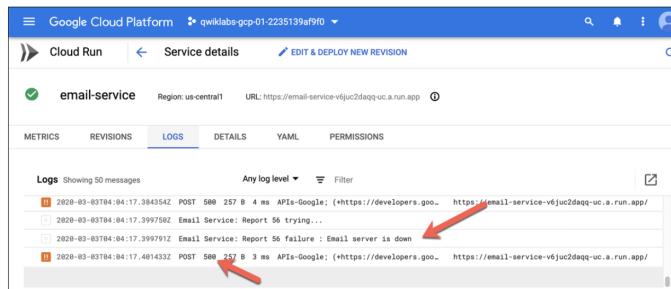
```
./deploy.sh
```

Quando a implantação do serviço de e-mail for concluída, poste novamente os dados dos resultados do laboratório e verifique atentamente o status do registro de **email-service**:

```
~/pet-theory/lab05/lab-service/post-reports.sh
```

Abra o registro do serviço de e-mail incorreto: **Menu de navegação > Cloud Run**. Quando você vir os três serviços do Cloud Run na sua conta, clique em **email-service**.

O serviço de e-mail está sendo invocado, mas continuará dando erro. Se você rolar de volta nos registros, encontrará a causa-raiz: "Email server is down". Você também verá que o serviço retorna o código de status 500 e que o Pub/Sub continua tentando chamar o serviço.



Se você observar os registros do serviço de SMS, verá que ele está funcionando bem.

Agora corrija o erro no serviço de e-mail e restaure o aplicativo.

Abra o arquivo `index.js`, remova a linha `throw` inserida anteriormente e salve o arquivo.

A função `sendEmail` de `index.js` agora se parece com o seguinte:

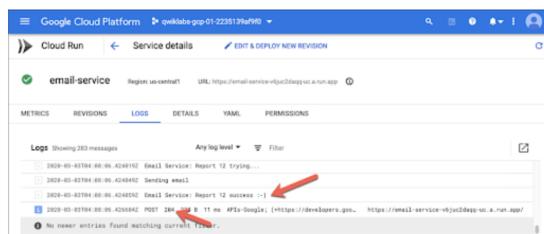
```
function sendEmail() {
  console.log('Sending email');
}
```

Implante a versão corrigida do serviço de e-mail:

```
./deploy.sh
```

Depois de terminada a implantação, clique no ícone **atualizar** no canto superior direito.

Você verá que, quando os e-mails dos resultados 12, 34 e 56 foram finalmente enviados, o serviço de e-mail retornou o código de status 204, e o Pub/Sub parou de invocar o serviço. Não houve perda de dados; o Pub/Sub continuou tentando até que finalmente conseguiu. Essas são as bases de um sistema robusto.



Principais pontos

1. Se os serviços se comunicarem de maneira assíncrona pelo Pub/Sub em vez de chamarem uns aos outros diretamente, o sistema poderá ser mais resiliente.
2. Graças ao uso do Pub/Sub, o acionamento do serviço de resultados do laboratório é independente de outros serviços. Por exemplo, se os clientes também quiserem receber os resultados do laboratório por algum outro serviço de mensagens, será possível adicioná-lo sem precisar atualizar o serviço de resultados.

- O Cloud Pub/Sub lidou com as tentativas sucessivas. Os serviços não precisaram fazer isso. Os serviços precisam apenas retornar o código de status: sucesso ou erro.
- Graças às sucessivas tentativas do Pub/Sub, se há a queda de um serviço, o sistema consegue "se corrigir" sozinho quando o serviço fica on-line novamente.

Parabéns!

Com sua ajuda, Raquel criou um sistema de protótipo resiliente. O serviço é capaz de enviar automaticamente a todos os clientes um e-mail e uma mensagem de SMS. No caso da inatividade temporária de algum serviço, o sistema implementará um mecanismo de tentativas sucessivas de modo a não haver perda de dados. Raquel recebe elogios bem merecidos.

 <i>Lilian, fundadora da Pet Theory</i>	<p>Olá, Raquel.</p> <p>Muito obrigada por todo seu trabalho árduo e liderança.</p> <p>Em um curto espaço de tempo, nossos principais sistemas foram totalmente reformulados.</p> <p>Na sexta-feira, faremos uma pequena reunião para celebrar, e seria ótimo se você pudesse participar como nossa convidada de honra.</p> <p>Lilian</p>
 <i>Melody, diretora administrativa</i>	<p>Raquel,</p> <p>Recebi da Pet Theory elogios pelo seu trabalho. Você é muito importante para nossa equipe.</p> <p>Agora que você terminou essa tarefa, eu gostaria de conversar com você sobre uma função mais sênior em um novo projeto.</p> <p>Melody</p> <p>Diretora administrativa</p> <p>Computer Consulting Inc.</p>



Termine a Quest

Este laboratório autoguiado faz parte da Quest [Google Cloud Run Serverless Workshop](#) do Qwiklabs. Uma Quest é uma série de laboratórios relacionados que formam o programa de aprendizado. Concluir esta Quest dá a você o selo acima como reconhecimento pela sua conquista. Você pode publicar os selos e incluir um link para eles no seu currículo online ou nas redes sociais. Caso você já tenha feito este laboratório, inscreva-se nesta Quest para ganhar os créditos de conclusão imediatamente. Veja outras [Quests do Qwiklabs](#).

Faça o seu próximo laboratório

Continue sua quest com o próximo laboratório da série, [Developing a REST API with Go and Cloud Run](#).

Próximas etapas / Saiba mais

Artigo do Medium: [Cloud Run as an internal async worker](#)

Terminar o laboratório

Após terminar seu laboratório, clique em **End Lab**. O Qwiklabs removerá os recursos usados e limpará a conta para você.

Você poderá classificar sua experiência neste laboratório. Basta selecionar o número de estrelas, digitar um comentário e clicar em **Submit**.

O número de estrelas indica o seguinte:

- 1 estrela = muito insatisfeito
- 2 estrelas = insatisfeito
- 3 estrelas = neutro
- 4 estrelas = satisfeito
- 5 estrelas = muito satisfeito

Feche a caixa de diálogo se não quiser enviar feedback.

Para enviar seu feedback, fazer sugestões ou correções, use a guia **Support**.

Treinamento e certificação do Google Cloud

...ajuda você a aproveitar as tecnologias do Google Cloud ao máximo. [Nossas aulas](#) incluem habilidades técnicas e práticas recomendadas para ajudar você a alcançar rapidamente o nível esperado e continuar sua jornada de aprendizado. Oferecemos treinamentos que vão do nível básico ao avançado, com opções de aulas virtuais, sob demanda e por meio de transmissões ao vivo para que você possa encaixá-las na correria do seu dia a dia. As [certificações](#) ajudam você a validar e comprovar suas habilidades e conhecimentos das tecnologias do Google Cloud.

Manual atualizado em 26 de outubro de 2021

Laboratório testado em 26 de outubro de 2021

Copyright 2020 Google LLC. Todos os direitos reservados. Google e o logotipo do Google são marcas registradas da Google LLC. Todos os outros nomes de produtos e empresas podem ser marcas registradas das respectivas empresas a que estão associados.

Continuar a Quest

Laboratório

Crie um sistema resiliente e assíncrono com o Cloud Run e o Pub/Sub

Iniciar