

GCP Certification Series: 4.3

Gerenciando recursos do App Engine

Ajustando os parâmetros de divisão de tráfego do aplicativo.



Prashanta Paudel

9 de novembro de 2018 · 8 minutos de leitura

Quando você implanta mais de uma instância do aplicativo com várias versões ou apenas muitas instâncias com a mesma versão, é inútil até que você especifique o método de divisão de tráfego entre elas.

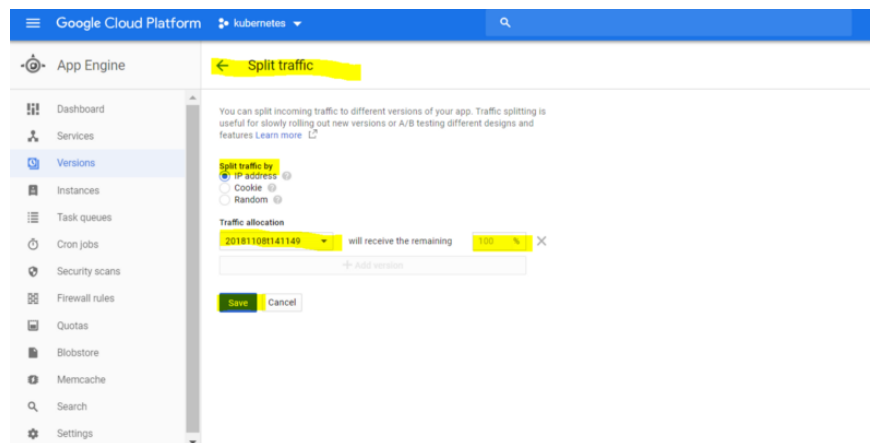
Podemos dividir o tráfego com base em

- Endereço de IP
- Biscoitos

A divisão do tráfego também ajuda a implantar a atualização contínua e a verificar o desempenho da nova versão do software.

Dividindo o tráfego entre as mesmas versões

No console goto versões dentro do motor de aplicativo e clique em tráfego dividido, então, você receberá um prompt onde você pode especificar o tipo de divisão e qual a percentagem de tráfego a ser servido pela versão específica do aplicativo.



tráfego dividido

Divisão de tráfego em várias versões

Quando tiver especificado duas ou mais versões para divisão, você deverá escolher se deseja dividir o tráfego usando um endereço IP ou um cookie HTTP. É mais fácil configurar uma divisão de endereço IP, mas uma divisão de cookie é mais precisa. Para mais informações, consulte [Divisão de endereço IP](#) e [Divisão de cookie](#).

Divisão de endereço IP

Se você optar por dividir o tráfego para o seu aplicativo por endereço IP, quando o aplicativo receber uma solicitação, ele colocará o endereço IP em um valor entre 0 e 999 e usará esse número para rotear a solicitação.

A divisão de endereços IP tem algumas limitações significativas:

- Os endereços IP são razoavelmente fixos, mas não são permanentes. Os usuários que se conectam a partir de telefones celulares podem ter um endereço IP de deslocamento ao longo de uma única sessão. Da mesma forma, um usuário em um laptop pode estar mudando de casa para um café para trabalhar e também mudará através de endereços IP. Como resultado, o usuário pode ter uma experiência inconsistente com seu aplicativo à medida que o endereço IP é alterado.
- Como os endereços IP são atribuídos independentemente a versões, a divisão de tráfego resultante será um pouco diferente do que você especificar. Embora, como seu aplicativo recebe mais

tráfego, quanto mais próximo a divisão real chega ao seu destino. Por exemplo, se você pedir que 5% do tráfego sejam entregues para uma versão alternativa, a porcentagem inicial de tráfego para a versão pode estar entre 3 e 7%, mas, em média, a média aproximará sua meta em 5%.

- Se você precisar enviar solicitações internas entre aplicativos, use a divisão de cookies. Os pedidos enviados entre aplicativos em execução na infraestrutura de nuvem do Google são originários de um pequeno número de endereços IP que provavelmente são todos atribuídos à mesma versão. Portanto, todas as solicitações internas podem se comportar de maneira semelhante às solicitações enviadas de um único endereço IP, o que significa que essas solicitações são todas roteadas para a mesma versão. Como resultado, as solicitações internas não respeitam muito as porcentagens que você definiu para as divisões de tráfego baseadas em IP. Por exemplo, se você definir uma versão para receber 1% de todo o tráfego do seu aplicativo e os endereços da infraestrutura da nuvem do Google forem atribuídos coincidentemente a essa versão, o resultado real poderá exceder 1%, pois todas as solicitações internas serão sempre encaminhadas para a versão atribuída.

Cookie splitting

Se você optar por dividir o tráfego para seu aplicativo por cookies, o aplicativo procurará no cabeçalho de solicitação HTTP por um cookie chamado `GOOGAPPUID`, que contém um valor entre 0 e 999:

- Se o cookie existir, o valor será usado para rotear a solicitação.
- Se não houver tal cookie, a solicitação será roteada aleatoriamente.

Se a resposta não contiver o `GOOGAPPUID` cookie, o aplicativo primeiro adicionará o `GOOGAPPUID` cookie com um valor aleatório entre 0 e 999 antes de ser enviado.

O uso de cookies para dividir o tráfego facilita a atribuição precisa de usuários a versões. A precisão do roteamento de tráfego pode chegar a 0,1% da divisão desejada. Embora a divisão de cookies tenha as seguintes limitações:

- Se você está escrevendo um aplicativo para dispositivos móveis ou executando um cliente de desktop, ele precisa gerenciar os `GOOGAPPID` cookies. Por exemplo, quando um `Set-Cookie` cabeçalho de resposta é usado, você deve armazenar o cookie e incluí-lo em cada solicitação subsequente. Os aplicativos baseados em navegador já gerenciam os cookies dessa maneira automaticamente.
- Dividir solicitações internas requer trabalho extra. Todas as solicitações de usuários enviadas de dentro da infraestrutura de nuvem do Google exigem que você encaminhe o cookie do usuário a cada solicitação. Por exemplo, você deve encaminhar o cookie do usuário em solicitações enviadas do seu aplicativo para outro aplicativo ou para ele mesmo. Observe que não é recomendado enviar solicitações internas se essas solicitações não forem originadas de um usuário.

Desativando a divisão de tráfego

Para desabilitar a divisão de tráfego, você migra todo o tráfego para uma única versão.

Divisão de tráfego por meio da CLI gcloud

Na `gcloud` CLI, você pode usar o seguinte subcomando:

```
gcloud app services set-traffic [MY_SERVICE] \  
  --splits [MY_VERSION1]=[VERSION1_WEIGHT],[MY_VERSION2]=[VERSION2_WEIGHT] \  
  --split-by [IP_OR_COOKIE]
```

Por exemplo:

```
gcloud app services set-traffic \  
  --splits 3=.1,2=.9 \  
  --split-by cookie
```

Definindo parâmetros de escalonamento automático com o API Explorer

Python 2.7 /3.7 | Java 8 | PHP 5.5 /7.2 | Go 1.9 /1.11 | Node.js

Se você usar o conjunto de ferramentas do Cloud SDK para implantar seu aplicativo, por exemplo `gcloud app deploy`, poderá definir os seguintes parâmetros de escalonamento automático no `app.yaml` arquivo de configuração:

- `min_instances`
- `max_instances`
- `target_throughput_utilization`
- `target_cpu_utilization`

No entanto, se você implantar usando a `appcfg` ferramenta do SDK do App Engine para Python, não poderá definir esses parâmetros de escalonamento automático no `app.yaml` arquivo de configuração. Em vez disso, você deve omitir esses parâmetros do arquivo de configuração e defini-los diretamente na interface do usuário do API explorer, após implantar seu aplicativo.

Para usar a interface do usuário do API explorer para definir um parâmetro de escalonamento automático:

1. Abra a página do explorador de APIs.
2. No painel direito, sob o rótulo *Try This API*, localize a caixa de texto de **nome** e insira a string do nome do aplicativo no seguinte formato:

```
apps/<YOUR-PROJECT-ID>/services/default/versions/<YOUR-VERSION-ID>
```

1. Substitua `YOUR-PROJECT-ID` pelo ID do projeto do aplicativo e, `<YOUR-VERSION-ID>` com a versão do aplicativo, você está enviando a solicitação para. Use o resto da string como mostrado.
2. Na caixa de texto **updateMask**, insira o `.json` nome completo do caminho do objeto que você está definindo, usando os

updateMask nomes da tabela abaixo:

3. updateMask

```
name automatic_scaling.standard_scheduler_settings.max_instances
automatic_scaling.standard_scheduler_settings.min_instances
automatic_scaling.standard_scheduler_settings.target_cpu_utilization
automatic_scaling.standard_scheduler_settings.target_throughput_utilization
```

- Se você estiver configurando mais de um parâmetro em uma solicitação, forneça o nome da máscara para cada parâmetro, separado por uma vírgula. Por exemplo, se você estiver definindo as instâncias mín e máx e a utilização da CPU, use a seguinte updateMask:

```
automatic_scaling.standard_scheduler_settings.max_instances,
automatic_scaling.standard_scheduler_settings.min_instances,
automatic_scaling.standard_scheduler_settings.target_cpu_utilization
```

- Na caixa **Solicitar corpo**, clique em **Adicionar parâmetros do corpo da solicitação**.
- Selecione o **dimensionamento automático**.
- Clique na bolha de dicas (o **+** ícone) e selecione **standardSchedulerSettings**.
- Clique na bolha da sugestão, selecione o parâmetro do planejador de escalonamento automático desejado e forneça o valor desejado.
- Para fornecer outro parâmetro do escalonador de escalonamento automático, clique no balão de dicas novamente, selecione o parâmetro e forneça seu valor.
- O exemplo a seguir mostra um corpo de solicitação de amostra preenchido:

```
{
  "automaticScaling": {
    "standardSchedulerSettings": {
      "maxInstances": 100,
      "minInstances": 1,
      "targetCpuUtilization": 0.75
```

```
}
}
}
```

1. Clique em **Executar** . Você pode ser solicitado a autorizar o API Explorer na primeira vez que executar isso. Se você for solicitado, autorize o API Explorer seguindo os prompts.
2. Confirme se as configurações corretas foram aplicadas abrindo a página de versões do App Engine para o seu projeto e clicando em **Visualizar** na coluna *Configuração* . Você deve ver os valores que acabou de definir.

Arquivos de configuração

Python 2.7 /3.7 | Java 8 | PHP 5.5 /7.2 | Go 1.9 /1.11 | Node.js 8

Nota : Os serviços eram anteriormente chamados de “módulos”. Cada versão de um serviço é definida em um `.yaml` arquivo, que fornece o nome do serviço e da versão. O arquivo YAML geralmente recebe o mesmo nome do serviço que ele define, mas isso não é necessário. Se você estiver implantando várias versões de um serviço, poderá criar vários arquivos yaml no mesmo diretório, um para cada versão.

Normalmente, você cria um diretório para cada serviço, que contém os arquivos YAML do serviço e o código-fonte associado. Arquivos de configuração de nível de aplicativo opcional (`dispatch.yaml` , `cron.yaml` , `index.yaml` , e `queue.yaml`) estão incluídos no diretório do aplicativo nível superior. O exemplo abaixo mostra três serviços. Em `service1` e `service2` , os arquivos de origem estão no mesmo nível do arquivo YAML. Em `service3` , existem arquivos YAML para duas versões.

Para projetos pequenos e simples, todos os arquivos do aplicativo podem residir em um diretório:

Todo arquivo YAML deve incluir um parâmetro de versão. Para definir o serviço padrão, você pode incluir explicitamente o parâmetro `service: default` ou deixar o parâmetro de serviço fora do arquivo.

O arquivo de configuração de cada serviço define o tipo de dimensionamento e a classe de instância para um serviço / versão

específico. Diferentes parâmetros de escala são usados dependendo do tipo de escala que você especificar. Se você não especificar o dimensionamento, o dimensionamento automático será o padrão. As configurações de escala e classe de instância são descritas na seção de [app.yaml referência](#).

Para cada serviço, você também pode especificar configurações que mapeiam solicitações de URL para scripts específicos e identificam arquivos estáticos para melhor eficiência do servidor. Essas configurações também estão incluídas no arquivo yaml e são descritas na seção de [app.yaml referência](#).

O serviço padrão

Cada aplicativo tem um único serviço padrão. Você pode definir o serviço padrão `app.yaml` com a configuração `service: default`, mas não é necessário fazer isso. Todos os parâmetros de configuração relevantes para serviços podem ser aplicados ao serviço padrão.

Arquivos de configuração opcionais

Esses arquivos de configuração controlam recursos opcionais que se aplicam a todos os serviços em um aplicativo:

- `dispatch.yaml`
- `queue.yaml`
- `index.yaml`
- `cron.yaml`
- `dos.yaml`

Para implantar atualizações desses arquivos de configuração no App Engine, execute o seguinte comando no diretório em que estão localizados:

```
gcloud app deploy [CONFIG_FILE]
```


Um exemplo

Aqui está um exemplo de como você configuraria arquivos YAML para um aplicativo que possui três serviços: um serviço padrão que lida com solicitações da web, além de mais dois serviços que manipulam solicitações móveis e processamento de back-end.

Comece definindo um arquivo de configuração nomeado `app.yaml` que manipulará todas as solicitações relacionadas à web:

```
runtime: python27
api_version: 1
threadsafe: true
```

Se o ID do projeto do console do GCP para este aplicativo for `simple-sample`, essa configuração criaria um serviço padrão com dimensionamento automático e um endereço público de <http://simple-sample.appspot.com>.

Em seguida, suponha que você deseja criar um serviço para lidar com solicitações da Web para dispositivos móveis. Para o bem dos usuários móveis (neste exemplo), a latência máxima pendente será apenas um segundo e sempre teremos pelo menos duas instâncias ociosas. Para configurar isso, você criaria um `mobile-frontend.yaml` arquivo de configuração. com o seguinte conteúdo:

```
service: mobile-frontend
runtime: python27
api_version: 1
threadsafe: true

automatic_scaling:
  min_idle_instances: 2
  max_pending_latency: 1s
```

O serviço criado por este arquivo pode ser acessado em <http://mobile-frontend.simple-sample.appspot.com>.

Por fim, adicione um serviço, chamado `my-service` por manipular o trabalho de back-end estático. Pode ser um trabalho contínuo que exporta dados do Datastore para o BigQuery. A quantidade de trabalho é relativamente fixa, portanto, você simplesmente precisa de um serviço residente a qualquer momento. Além disso, esses trabalhos precisarão lidar com uma grande quantidade de processamento na memória, portanto, você precisará de serviços com uma configuração de memória maior. Para configurar isso, você criaria um `my-service.yaml` arquivo de configuração com o seguinte conteúdo.

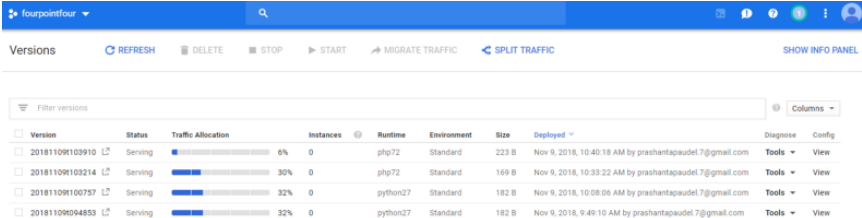
```
service: my-service
runtime: python27
api_version: 1
threadsafe: true

instance_class: B8
manual_scaling:
  instances: 1
```

O serviço criado por este arquivo pode ser acessado em <http://my-service.simple-sample.appspot.com>.

Observe a `manual_scaling`: configuração. O `instances`: parâmetro informa ao Google App Engine quantas instâncias serão criadas para esse serviço.

Você também pode querer baixar este [aplicativo de demonstração Python](#) e dar uma olhada.



Version	Status	Traffic Allocation	Instances	Runtime	Environment	Size	Deployed	Diagnose	Config
20181109103910	Serving	6%	0	php72	Standard	223 B	Nov 9, 2018, 10:40:18 AM by prashantapaudel.7@gmail.com	Tools	View
20181109103214	Serving	30%	0	php72	Standard	149 B	Nov 9, 2018, 10:33:22 AM by prashantapaudel.7@gmail.com	Tools	View
20181109100757	Serving	32%	0	python27	Standard	182 B	Nov 9, 2018, 10:08:06 AM by prashantapaudel.7@gmail.com	Tools	View
20181109094853	Serving	32%	0	python27	Standard	182 B	Nov 9, 2018, 9:49:10 AM by prashantapaudel.7@gmail.com	Tools	View

Tráfego dividido

