**Engenheiro de Dados GCP^1 v20190802**

**1. Visão Geral do Exame**

O exame consiste de 50 perguntas que devem ser respondidas em 2 horas.

O conteúdo inclui:

* Armazenamento (20% das perguntas),
* Processamento de Big Data (35%),
* Aprendizado de Máquina (18%),
* estudos de caso (15%) e
* outros (Hadoop e segurança cerca de 12%).

Todas as perguntas são cenários simulados onde você tem que escolher qual opção seria a melhor maneira de lidar com a situação.

**2. Ecossistema de Big Data^2**

Particularmente focando em Apache Pig, Hive, Spark e Beam.

* **Hadoop**
  + framework open source MapReduce
  + a tecnologia subjacente para Dataproc
* **HDFS**
  + Sistema de Arquivos Hadoop
* **Pig**
  + linguagem de script que compila em trabalhos MapReduce
  + Procedural Data Flow Language: Pig Latin
  + desenvolvimento eficiente e com menos código
  + eficiência
  + não tem noção de partições
  + suporta Avro
* **Hive**
  + sistema de armazenamento de dados e linguagem de consulta do tipo SQL
* **Spark**
  + rápido, interativo, framework de propósito geral para SQL, streaming, aprendizado de máquina, etc.
  + resolve problemas similares aos do Hadoop MapReduce mas com uma abordagem rápida in-memory
* **Sqoop**
  + transfere dados entre Hadoop e estruturas de dados relacionais
  + Sqoop importa dados de um sistema de banco de dados relacional para o HDFS.
  + Rodando Sqoop em um cluster Dataproc Hadoop dá acesso ao armazenamento incorporado no Google Cloud Storage Connector
  + Os dois pontos anteriores significam que você pode usar Sqoop para importar dados diretamente no Cloud Storage
* **Oozie**
  + sistema de agendamento de fluxo de trabalho para gerenciar trabalhos Hadoop
  + Trabalhos de fluxo de trabalho Oozie são Direcionados, Gráficos Acíclicos de Ações (DAGs)
* **Cassandra**
* Wide-column store baseada nas ideias de BigTable e DynamoDB(Datastore)
* Wide column store
* solução para problemas onde um dos seus requisitos é ter um sistema de escrita muito reativo e você quer ter um sistema de relatório responsivo em cima dos dados armazenados
* não fornece ACID e propriedades relacionais de dados
* disponível, sistema tolerante a partições que suporta consistência eventual
* **MongoDB**
  + é adequado para casos de uso onde seu sistema demanda um documento menos rígido
* **HBase**
  + pode ser adequado para motores de busca, análise de log, ou qualquer local onde a varredura de dados bidimensionais sem junções é um requerimento
* **Redis**
  + é construído para prover busca In-Memory para variedades de estruturas de dados como árvores, filas, listas vinculadas, etc, e pode ser usado para sistemas de ranking em tempo real como leaderboards, sistemas de publicação e assinatura.
* **MYSQL**
  + Você pode facilmente adicionar nós ao cluster MySQL e construí-lo para ser usado para OLAP (resposta para uma questão simulada)

**3. Armazenamento**

**Cloud Storage/Cloud SQL/Datastore/BigTable/BigQuery.**

**^1 por github.com/xg1990 modificado baseado nos 'GCP Dotpoints' do James com várias fontes online que são reconhecidas no final, por favor, compartilhe sob a licença Creative Commons Attribution 3.0 Austrália. Por favor, sinta-se livre para enviar qualquer problema em** [**https://github.com/xg1990/GCP-Data-Engineer-Study-Guide/**](https://github.com/xg1990/GCP-Data-Engineer-Study-Guide/)

**^2 referir a https://hadoopecosystemtable.github.io**

**3.1. Armazenamento na Nuvem (GCS)**

* Armazenamento tipo Blob. Faça upload de qualquer quantidade de bytes para um local. O conteúdo não é indexado, apenas armazenado (semelhante ao Amazon S3).
* Armazenamento ilimitado oferecido através do Nearline e Coldline:
  + Nearline para acesso aproximadamente mensal,
  + Coldline para acesso menos frequente.
* Buckets para separar itens de armazenamento geograficamente:
  + persistentes, duráveis, replicados,
  + espalhados por áreas de dados para minimizar impacto de interrupções de serviço,
  + espalhados por áreas de dados para fornecer acesso global aos dados.
* Ideal para armazenar, mas não para alto volume de leitura/escrita (por exemplo, dados de sensores).
* Uma maneira de armazenar os dados que podem ser comumente usados por Dataproc e BigQuery.

**Criptografia:**

* A Google Cloud Platform criptografa dados do cliente armazenados em repouso por padrão:
  + Sem opções: criptografia padrão do lado do servidor,
  + Criptografia suprida pelo cliente: você pode criar e gerenciar suas próprias chaves de criptografia para criptografia do lado do servidor,
  + Criptografia gerenciada pelo cliente: você pode gerar e gerenciar suas chaves de criptografia usando o Cloud Key Management Service,
  + Criptografia do lado do cliente: criptografia que ocorre antes que os dados sejam enviados para o Cloud Storage.

**3.2. Cloud SQL e Spanner**

* Banco de dados relacional gerenciado/Sem operações (MySQL e PostgreSQL) semelhante ao Amazon RDS.
* Adequado para gigabytes de dados com natureza transacional:
  + Baixa latência,
  + Escala bem além de gigabytes de dados,
  + Infraestrutura e camadas subjacentes são distribuídas e escaláveis.
* Spanner é uma solução de RDBMS distribuída e escalável, porém mais cara.
* Gerenciamento:
  + Backups gerenciados e replicação automática,
  + Conexão rápida com GCE/GAE,
  + Usa segurança do Google,
  + Preços flexíveis, pague pelo que usar.

**3.3. BigTable**

* Características:
  + Armazenado no armazenamento interno do Google Colossus,
  + Sem suporte transacional (portanto, pode lidar com petabytes de dados),
  + Sem relações (sem SQL ou junções), ACID apenas no nível de linha.
* Evita designs de esquema que requerem atomicidade entre linhas.
* Alto rendimento: O rendimento tem crescimento linear com a contagem de nós se equilibrado corretamente.
* Funciona com a API do HBase,
* Sem operações, auto-equilibrado, replicado, compactado.

**Consulta:**

* Busca por chave única. Sem busca por propriedade.
* Chaves armazenadas lexicograficamente em formato big endian para que as chaves possam ser qualquer coisa.
* Consulta rápida por intervalo.

**Desempenho:**

* Escala rápida até petabytes, não é uma boa solução para armazenar menos de 1 TB de dados.
* Acesso de leitura/escrita de baixa latência.
* Alto throughput de acessos.
* Suporte nativo para séries temporais.
* Para grandes cargas de trabalho analíticas e operacionais.
* Projetado para tabelas esparsas.

**Design Chave:**

* Projete suas chaves como pretende consultar.
* Se a consulta mais comum é pelos dados mais recentes, use uma chave de ordenação reversa no final da chave.
* Certifique-se de que suas chaves estejam distribuídas uniformemente para evitar pontos quentes. Isso é por que carimbos de data/hora como uma chave são uma má ideia, já que todas as gravações dos dados mais recentes estão sendo escritas ao mesmo tempo.
  + Para análise de dados históricos, os pontos quentes de leitura não são um grande problema.
* Para dados de séries temporais, use tabelas altas/estreitas. Designe preferencialmente múltiplas tabelas altas e estreitas.
* Evite hotspots
* Promoção de campo (preferencial): Mova campos do meio da coluna para o rodeio para tornar as escritas não contíguas.
* Salting: (onde a promoção de campo não resolve) Adicione um elemento calculado artificialmente à chave de linha para artificialmente tornar as escritas não contíguas.

**Teste de Desempenho:**

* Aprende sobre seus padrões de acesso e irá ajustar os metadados armazenados nos nós para balancear suas cargas de trabalho.
* Isso leva de minutos a horas e requer o uso de pelo menos 300GB de dados.
* Use uma instância de produção.
* Mantenha-se abaixo da utilização de armazenamento recomendada por nó.
* Antes de testar, execute um teste prévio intenso por vários minutos
* Executar teste por pelo menos 10 minutos

**Atualização de Dados**

* Ao consultar o BigTable, seleciona-se o valor mais recente que corresponde à chave.
  + Isso significa que ao deletar/atualizar realmente escrevemos uma nova linha com o dado deletado removido. Isso pode temporariamente aumentar o uso de disco.

**Colunas de Grupo**

* Agrupe colunas de dados que você provavelmente irá consultar juntas (por exemplo, endereços, primeiro/último nome e detalhes de contato).
* Use nomes curtos de colunas, organize em famílias de colunas (por exemplo, MD:symbol, MD:price).

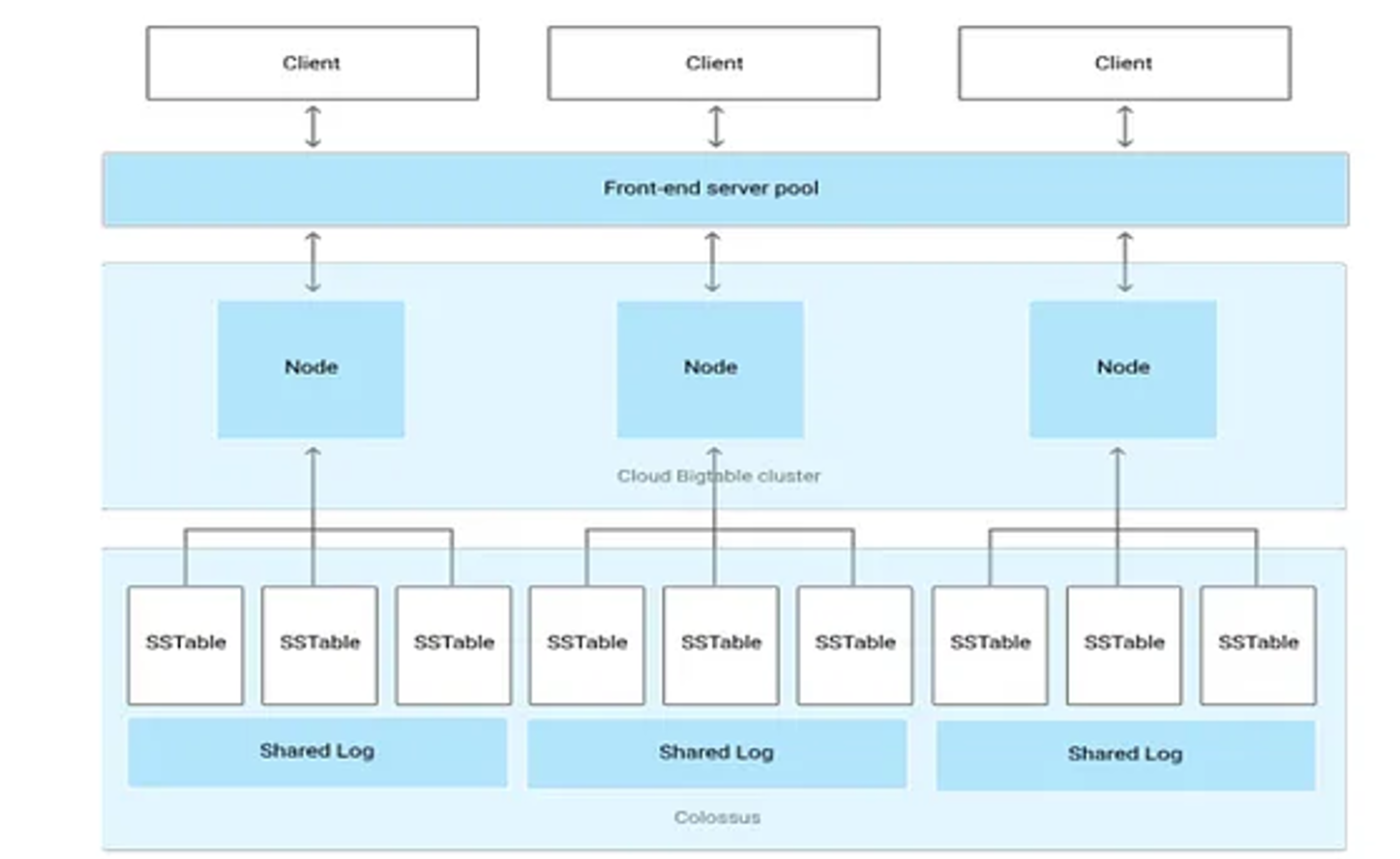
**Compactações Periódicas**

* O BigTable compacta periodicamente os dados para você. Isso significa reorganizar e remover registros deletados uma vez que não são mais necessários.

**Tabelas de Strings Ordenadas**

* O BigTable depende de Tabelas de Strings Ordenadas para organizar os dados.
* São pares imutáveis pré-ordenados de chave/valor de strings ou protobufs.

**Arquitetura**



**Limites de Tamanho**

* Uma chave de linha única: 4 KB (limite suave?)
* Um único valor em uma célula de tabela: 100 MB
* Todos os valores em uma única linha: 256 MB

**Controle de Acesso**

* Você pode configurar controle de acesso no nível do projeto e no nível da instância (recurso IAM mais baixo)
  + Uma instância Bigtable é na maioria das vezes um ponto de controle para seus usuários e nós, que fazem todo o trabalho real.
  + Tabelas pertencem a instâncias, não a clusters ou nós. Portanto, se você tem instâncias com até 2 clusters, você não pode atribuir tabelas a clusters individuais.

**Produção e Desenvolvimento**

* Produção: Uma instância padrão com 1 ou 2 clusters, bem como 3 ou mais nós em cada cluster.
  + use replicação para prover alta disponibilidade
* Desenvolvimento: Uma instância de baixo custo para desenvolvimento e teste, com desempenho limitado ao equivalente de um nó-1 de cluster.
  + É recomendado criar sua instância Compute Engine no mesmo zone que sua instância Cloud Bigtable para o melhor desempenho possível.

**Ferramentas**

* **cbt** é uma ferramenta para interações básicas com o Bigtable.
* **HBase shell** é uma ferramenta de linha de comando que executa tarefas administrativas, como criar e deletar tabelas.
  + você pode atualizar qualquer uma das seguintes configurações sem tempo de inatividade:
    - número de clusters / replicação
    - atualizar uma instância de desenvolvimento para produção (permanente)
* Impossível alternar entre SSD e HDD
  + exportar dados da instância existente e importar para uma nova instância.
  + OU escrever um trabalho Cloud Dataflow ou Hadoop MapReduce que copia os dados de uma instância para outra.

**3.4 Datastore**

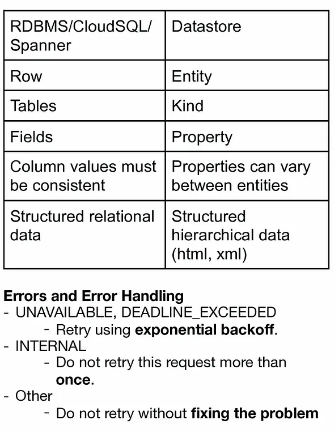
* Construído sobre o BigTable.
* Não consistente para cada linha
* Banco de dados documental adequado para dados não relacionais
* Adequado para:
  + Transações atômicas: executa um conjunto de operações onde todas são bem-sucedidas ou nenhuma.
  + Suporta transações ACID, consultas SQL-like.
  + Para dados estruturados.
  + Para armazenamento de documentos hierárquicos como HTML.

Consulta:

* + pode ser pesquisado por chaves ou propriedades (se indexado)
  + Buscas de chave um tanto quanto como o Amazon DynamoDB
  + Permite consultas como SQL até o nível de propriedade
  + não suporta junções complexas com múltiplos filtros de desigualdade

Performance:

* Rápido para terabytes de escala, baixa latência
* Leitura rápida, escrita lenta (pois depende de indexar cada propriedade por padrão) e deve atualizar índices conforme atualizações/escritas ocorrem



* Linha: **Entidade**
* Tabelas: **Kind**
* Campos: **Propriedade**
* Valores de coluna devem ser consistentes: Propriedades podem variar entre entidades
* Dados relacionais estruturados: Dados hierárquicos estruturados (html, xml)

Erros e Tratamento de Erros

* UNAVAILABLE, DEADLINE\_EXCEEDED
  + Tente novamente com backoff exponencial.
* INTERNAL
  + Não tente esta solicitação mais de uma vez.
* Outro
  + Não tente novamente sem corrigir o problema.

**3.5.BigQuery Característica**

* Armazenamento de dados completamente gerenciável
* Possui conectores para BigTable, GCS, Google Drive e pode importar de backups do Datastore, CSV, JSON e AVRO
  + para análises. Alternativa para Hadoop com Hive
* Performance
  + Escala petabyte
  + Mais usado para análise do que para baixa latência como RDBMS tipo CloudSQL ou Spanner

Query

* SQL padrão (preferencial) ou SQL legado (antigo)
* Não pode usar ambos SQL legado e SQL2011 na mesma consulta
* Particionamento de tabela
* Escrita distribuída para arquivo de saída. Ex.: file-00001-of-00002
* Funções definidas pelo usuário em JS (UDFJS)
* Consultas assíncronas são executadas para carregar, exportar, consultar ou copiar dados.
* Se você usar a cláusula LIMIT, o BigQuery ainda processará a tabela inteira.
* Evite SELECT \*, selecione apenas as colunas necessárias (SELECT EXCEPT)
  + benefícios de usar dados desnormalizados
    - aumenta a velocidade da consulta
    - simplifica consultas
    - MAS: a normalização é a maneira mais organizada mas com menos performance otimizada

tipos de consultas:

* Interativa: consulta é executada imediatamente, conta em direção à cota diária/concorrente (padrão)
* Batch: lotes de consultas são enfileirados e a consulta começa quando recursos ociosos estão disponíveis, conta apenas para a cota diária e troca para interativa se ociosa por 24 horas

Data Import

* em lote (grátis)
* console web (arquivos locais), GCS, GDS
* stream (caro)
  + dados com CDF, registro de Cloud ou POST de arquivos
* Fontes de dados brutas:
  + fonte de dados federada, CSV/JSON/Avro em GCS, planilhas do Google
  + Google Drive
    - Carregar dados no BigQuery do Google Drive não é atualmente suportado, mas é possível carregar dados no Google Drive usando uma extensão beta.
* Por padrão, o serviço BigQuery espera todos os dados de entrada para serem UTF-8 codificados
  + os arquivos JSON devem sempre ser codificados em UTF-8
* para suportar mudança de esquema ocasional, você pode usar 'Detectar automaticamente' para esquemas de mudança. Automaticamente detectar não é o padrão sugerido.
* Você não pode usar a Web UI para:
  + Fazer upload de um arquivo maior que 10 MB de tamanho
  + Fazer upload de múltiplos arquivos ao mesmo tempo
  + Fazer upload de um arquivo no formato SQL

Partições

* o qual melhora a performance da consulta e reduz custos
* Você não pode alterar uma tabela existente em uma tabela particionada. Você deve criar uma tabela particionada do zero.
* Dois tipos de tabelas particionadas:
  + Tabelas de ingestão de tempo: Tabelas particionadas baseadas na data de ingestão (data de carregamento) dos dados. Cada partição (lote) terá um pseudocampo \_PARTITIONTIME ou terá dados carregados no lote.

**Tabelas Curinga**

* Usadas se você quer unir todas as tabelas semelhantes que possuem nomes semelhantes de tabela. (por exemplo, **project.dataset.table\***)
* Tabelas Particionadas incluem uma pseudocoluna chamada **\_PARTITIONTIME** que contém um carimbo de data/hora baseado em data para carregar os dados na tabela
  + Pode ser usado para especificar partições específicas na cláusula WHERE

**Janelamento (Windowing)**

* Funções de janelamento aumentam a eficiência e reduzem a complexidade de consultas que analisam partições (windows) de dados ao evitar a divisão de operações complexas sem a necessidade de muitas tabelas intermediárias para armazenar dados temporários.

**Bucketing**

* Como o particionamento, mas cada divisão/partição deve ter o mesmo tamanho e é baseada no hash de uma coluna. Cada bucket é um arquivo separado, o que torna a amostragem e a junção de dados mais eficientes.

**legacy vs. standard SQL^3**

* **project.dataset.table\***
* É definido cada vez que você executa uma consulta
* A linguagem de consulta padrão é
  + SQL padrão para a interface do usuário Beta
  + SQL legado para a interface do usuário clássica

**Antipadrões**

* Evite self-joins
* Particionamento/Skew: evite partições desigualmente dimensionadas ou quando um valor ocorre mais frequentemente que qualquer outro valor

**Cross-Join**

* Evite joins que geram mais saídas do que entradas
* Update/Insert Single Row/Column: evite ponto-específicos DML, em vez disso faça atualizações em lotes e inserções em anexo
* Antipadrões e design de schema: <https://cloud.google.com/bigtable/docs/schema-design>

**Controle de Acesso**

* A segurança pode ser aplicada no nível do projeto e no nível do conjunto de dados, mas não no nível da tabela ou visualização
* Organize níveis de recursos no BigQuery com visões autorizadas, projetos e conjuntos de dados
* Visões autorizadas permitem que você compartilhe os resultados da consulta com usuários/grupos sem lhes dar acesso aos dados subjacentes
  + Pode ser usado para restringir o acesso a colunas ou linhas particulares
  + Crie um conjunto de dados separado para armazenar a visão

**Faturamento**

* Baseado no armazenamento (quantidade de dados armazenados), consultas (quantidade de dados/número de bytes processados pela consulta) e inserções em fluxo contínuo
* Opções de armazenamento são ativas e longo-prazo (modificadas ou não nos últimos 90 dias).
* Opções de consulta são sob demanda e tarifa fixa.

**Tipos de Tabela**

* Tabelas Nativas: tabelas respaldadas pelo armazenamento nativo do BigQuery.
* Tabelas Externas: tabelas respaldadas pelo armazenamento externo ao BigQuery (também conhecido como fonte de dados federados). Para mais informações, veja Consultando Dados Externos.

^3 <http://cloud.google.com/bigquery/docs/reference/standard-sql/migrating-from-legacy-sql>

**4. Processamento de Big Data** Cobrindo conhecimentos sobre BigQuery, Cloud Dataflow, Cloud Dataproc, Cloud Datalab e Cloud Pub/Sub.

**4.1. App Engine**

* Executar código em instâncias gerenciadas de máquinas com dimensionamento automático e implantação
* Lidar com aumentos súbitos e extremos de tráfego que exigem resposta imediata

**4.2. GCP Compute**

* VM
* Instâncias pré-emptivas (até 80% de desconto mas podem ser retiradas)
* Alocadas sob demanda e só paga pelo tempo que estão ativas

**4.3. Dataflow** Características

* Executa pipelines do Apache Beam (sem operações, usa Spark, Flink)
* Pode ser usado para processamento de dados em lote ou em fluxo contínuo
* Escalável, tolerante a falhas, processamento de dados multi-etapa
* Comumente usado para preparação/ETL de conjuntos de dados
* Filtrar, agrupar, transformar
* Pipelines e como eles funcionam para ETL

**DataSource**

* O conector Cloud Dataflow para o Cloud Bigtable permite usar o Cloud Bigtable em um pipeline Dataflow.
* Pode ler dados de múltiplas fontes e pode iniciar múltiplas execuções de jobs em paralelo escrevendo para múltiplos sinks de maneira distribuída

**Janelamento**

* Pode aplicar janelamento a fluxos para média móvel, máximo em uma janela, etc.
* tipos de janelas
  + Janelas de Tempo Fixo
  + Janelas de Tempo Deslizante (sobrepostas)
  + Janelas de Sessão
  + Janela Global Única
* Comportamento de janelamento padrão é atribuir todos os elementos de uma PCollection para uma única janela global, mesmo para PCollections ilimitadas

**Gatilhos**

* Determina quando o conteúdo de uma janela deve ser emitido com base em certos critérios sendo atendidos
  + Permite especificar um critério para controlar quando (no tempo de processamento) resultados para janelas em processamento devem ser emitidos.
  + Se não especificado, o comportamento padrão é disparar primeiro quando o marcador de água passar o fim da janela e, em seguida, disparar novamente sempre que houver dados tardios.

**Gatilhos Baseados em Tempo**

* Gatilhos de Tempo do Evento. Estes gatilhos operam no tempo do evento, conforme indicado pelo carimbo de tempo em cada elemento de dados.
* Gatilhos de Tempo de Processamento. Estes gatilhos operam no tempo de processamento - o tempo em que o elemento de dado está sendo processado em qualquer etapa no pipeline.

**Gatilhos Baseados em Dados**

* Estes gatilhos operam examinando os dados conforme eles chegam em cada janela e disparando quando esses dados atendem a uma certa propriedade.

**Tecnologia:**

* **PCollections**: abstração que representa um conjunto de dados potencialmente distribuído e multi-elementos que atua como a pipeline de dados.
* **Transform**: é uma operação de processamento de dados, ou uma etapa, na sua pipeline. Uma transformação toma uma ou mais PCollections como entrada, realiza um processamento que você fornece nos elementos dessa PCollection, e produz uma saída PCollection.
* **DirectPipelineRunner** permite que você execute operações na pipeline diretamente e localmente
* Crie um trabalho cron com o Google App Engine Cron Service para executar o trabalho do Cloud Dataflow

**IAM:**

* **dataflow.developer** permite que o desenvolvedor interaja com o trabalho do Cloud Dataflow
* **dataflow.worker** fornece as permissões necessárias para uma Conta de Serviço Compute Engine executar unidades de trabalho para uma pipeline do Dataflow

**Atualização de Pipeline:**

* Com Update, para substituir uma pipeline no lugar com o novo código e preservar garantias de processamento exatamente uma vez do Dataflow quando aplicável.
* O comando DRAIN é usado em vez de CANCEL para manter os dados em trânsito (in-flight) durante a atualização.
  + A funcionalidade de DRAIN é suportada apenas para pipelines de streaming.
* pipelines não podem compartilhar dados ou transformações

**Exportação:**

* Os dados só podem ser exportados em JSON / CSV / Avro
  + A única opção de compressão disponível é GZIP.
    - A compressão GZIP não é suportada para exportações Avro.
  + Para exportar mais de 1 GB de dados, você precisa colocar um curinga no nome do arquivo de destino (até 1 TB de dados em uma única fila).

Para mais informações, veja: <https://cloud.google.com/bigquery/docs/> <https://github.com/jorwalk/data-engineering-gcp/blob/master/know/bigquery.md>

**Coisas chave para focar são:**

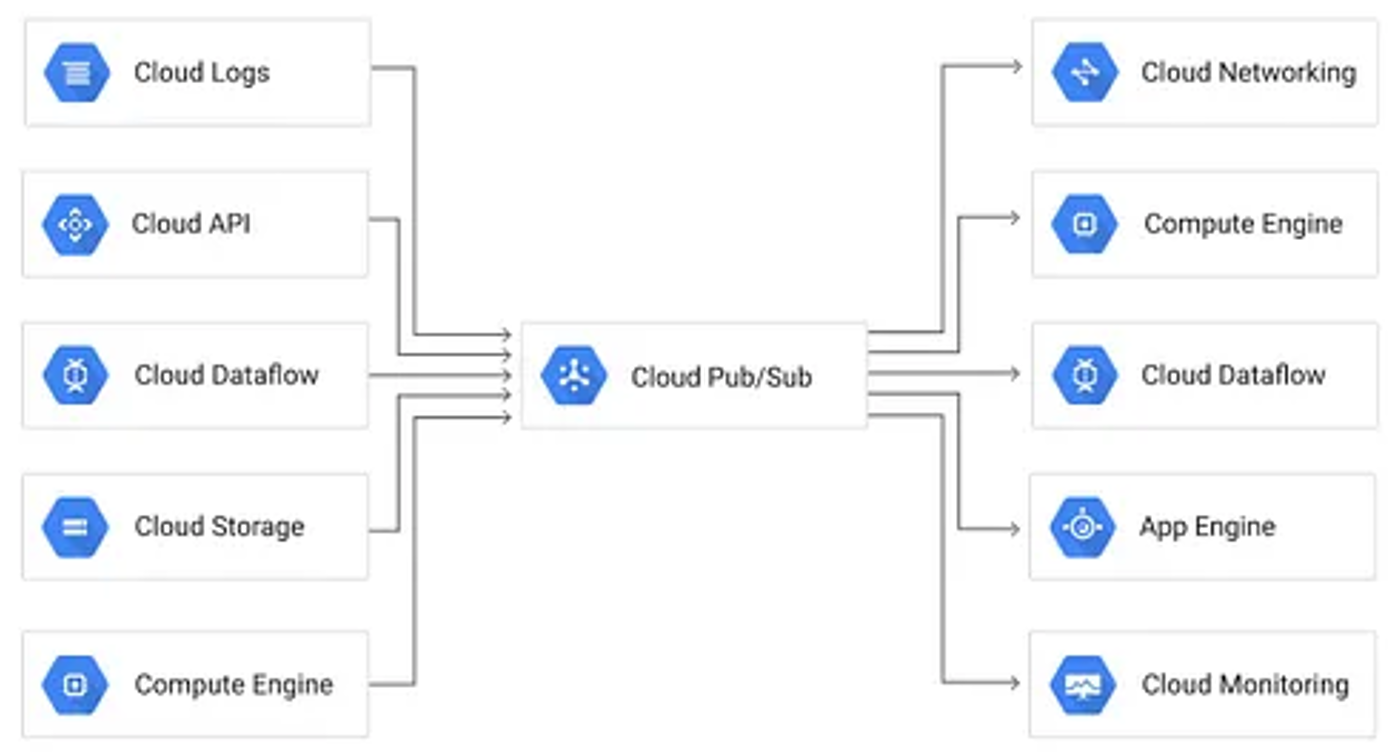
* Evento vs. Tempo de processamento
* Configurando pipelines ETL

Como integrar com BigQuery e as restrições que você pode ter. Por que você usaria JSON ou Java relacionado a Pipelines.

**4.4. Cloud Pub/Sub**

Características

* Mensageria sem servidor
* Desacopla produtores e consumidores de dados em organizações grandes / complexas
* O adesivo que conecta todos os componentes.
* Ordem não garantida



Processamento Assíncrono

* Disponibilidade (buffer durante quedas)
* Mudança de gerenciamento
* Throughput (balanceamento de carga entre os trabalhadores)
* União (gerenciamento de requisições na borda da rede)
* Consistência

Conceitos básicos

* tópicos
* inscrições
* push e pull
  + pull é um mecanismo mais eficiente de entrega/recebimento de mensagens

Fluxo de Mensagens

* Publicador cria um tópico no Cloud Pub/Sub e envia mensagens para o tópico.
* Mensagens são persistidas em um armazém de mensagens até que sejam entregues e reconhecidas pelos assinantes.
* O Pub/Sub encaminha mensagens de um tópico para todas as suas inscrições, individualmente. Cada inscrição recebe mensagens seja por push/pulling.
* O assinante recebe mensagens pendentes de sua inscrição e reconhece a mensagem.
* Quando uma mensagem é reconhecida pelo assinante, ela é removida da fila de mensagens da inscrição.

Deduplicação

* Manter um banco de dados para armazenar o hash de valor e outros metadados de cada entrada.
* Cloud Pub/Sub atribui um **message\_id** único para cada mensagem, que pode ser usado para detectar mensagens duplicadas recebidas pelo assinante.
* Muitas mensagens duplicadas podem ocorrer quando: o ponto final não está reconhecendo mensagens dentro do prazo de reconhecimento.

**4.5. Dataproc** Características

* Dataproc é um cluster Hadoop gerenciado (No Ops) no GCP (por exemplo, Hadoop gerenciado, Pig, Hive, programas Spark)
* Gerenciamento automático de cluster, redimensionamento
* Apenas código/Query
* Gerenciamento de trabalho somente pela tela do console.
* Pense em termos de 'recurso específico para o trabalho', para cada trabalho, crie um cluster e depois exclua.

Usado se **migrando infraestrutura Hadoop ou Spark existente on-premise para o Google Cloud Platform sem esforço de redeseenvolvimento.**

Armazenamento

* Pode armazenar dados no HDFS ou pode usar GCS
* GCS permite o uso de máquinas preemptíveis que podem reduzir significativamente os custos
  + armazene no HDFS (dividido no cluster, mas requer cluster ativo) ou no GCS (separado do cluster e armazenamento)

Personalize o software

* Defina ações de inicialização
* Modifique configurações do cluster usando propriedades
* Faça login no nó mestre e faça alterações a partir daí
* NÃO use o Gerenciador de Implantação

Tecnologia:

* criar um novo cluster Dataproc com a operação projects.regions.clusters.create, quatro valores são necessários: projeto, região, nome e zonas.
* você pode acessar o YARN via interface web configurando um proxy SOCKS
* você pode SSH diretamente para os nós do cluster
* você pode usar um proxy SOCKS para conectar seu navegador através de um túnel SSH.
* YARN ResourceManager e as interfaces HDFS NameNode estão disponíveis no nó mestre

Faturamento:

* é cobrado pelo segundo. Todos os clusters Dataproc são faturados em incrementos de relógio de um segundo, sujeito a um mínimo de faturamento de um minuto.

**IAM**

* Contas de serviço usadas com o Cloud Dataproc devem ter a função Dataproc Worker (ou ter todas as permissões concedidas pelo papel de Worker do Dataproc).
  + precisam de permissões para ler e escrever no Google Cloud Storage e para o Google Cloud Logging.

**4.6. Dataprep**

* Trifacta gerenciado para preparar e analisar a qualidade e transformar os dados de entrada
* serviço para visualmente explorar, limpar e preparar dados para análise, pode transformar dados de qualquer tamanho armazenados em CSV, JSON ou formatos relacionais-tabelares

**4.7. Cloud Functions**

* Funções NodeJS como um serviço
* Sem operações, o servidor apenas codifica o ponto de entrada e a resposta, auto-escalado pelo GCP
* Pode ser disparado por dataflow, GCS dispara eventos de bucket, mensagens Pub/Sub e chamadas HTTP

**5. Aprendizado de Máquina^4** Cobrindo conhecimento sobre GCP API (Vision API, Speech API, Natural Language API e Translate API) e TensorFlow.

* Embeddings
* Deploying Models

^4 Esta parte ainda não está bem organizada

**Folha de Dicas e Terminologia do TensorFlow** Entenda os diferentes serviços de ML disponíveis: ML Engine, ML APIs e TensorFlow, bem como a relevância do Cloud Datalab. Principalmente domínio de ML (e não específico do TensorFlow) cobrindo basicamente sobre treinamento. Nada sobre o serviço Cloud ML.

**5.1. Termos de ML**

* Rótulo: A classificação/valor correto
* Entrada: Variáveis preditoras
* Exemplo: Entrada + Rótulo para treinar o modelo
* Modelo: Matemática simplificada, algum trabalho é feito nas entradas para produzir uma saída
* Treinamento: Ajustando os pesos em um modelo para minimizar erro
* Predição: Usando o modelo para adivinhar o rótulo para uma entrada
* Aprendizado Supervisionado: Treinar o modelo usando dados etiquetados para prever dados futuros
* Aprendizado Não Supervisionado: Treinar é feito sem rótulos para padrões ou clusters
* Neurônio: Uma maneira de combinar entradas e ponderá-las para tomar uma decisão (é uma unidade de combinação)
* Descida de Gradiente: O processo de testar erro para minimizar seu valor iterativamente diminuindo em direção a um mínimo
* Taxa de Aprendizado: passo crítico para garantir que o processo de descida de gradiente chegue a um mínimo global ou local
* Camada Escondida: Um conjunto de neurônios que atuam nos mesmos dados de entrada

**Recursos:**

* Os valores/campos de dados que você escolhe para modelar, podem ser transformados (x^2, y^2, etc)
* Feature Engineering: O processo de construir um conjunto de combinações de recursos para agir sobre as entradas
* Precisão: O quão preditivo é um modelo em quantas vezes ele corretamente previu uma classificação (gato)
* Recall: A taxa verdadeira positiva, quantas vezes um modelo pensa que é da classe (o número real de gatos)
  + Apenas 1/10 reconhecido corretamente, mas estava certo. 100% precisão, 10% recall

**Motor de Recomendação:**

* Cluster similaridade entre Usuário A e Usuário B ambos classificam a Casa Z como 4
* Cluster itens similares (produtos, casas, etc): A maioria dos usuários classifica a Casa Y como 2
* Combina esses dois para produzir uma classificação

**5.1. Básicos de ML**

* Há duas principais etapas de ML, Treinamento e Inferência. Inferência é frequentemente preditiva na natureza
* Conversão de entradas em vetores para análise bem como problemas documentados que podem ser melhorados com o uso do ML em si. Alguns públicos incluem <https://en.wikipedia.org/wiki/Word2vec>
* Inicialização do peso é importante para acertar, intuição humana pode ajudar com esse início
* Pesos são ajustados iterativamente. Pesos iniciais -> Calcular Erro -> ajustar peso ->recalcular erro -> repetir até o erro ser minimizado
* Na imagem ML, cada pixel é representado por um número. Ver texto em imagens é substancialmente mais fácil do que ver quando você tem que reconhecer que Homem é para Mulher assim como Menino é para Menina; então eles precisam ter diferenças de magnitude semelhantes.

**Rede Neural**

* O objetivo é minimizar o custo
* O custo depende do problema - geralmente a soma dos erros quadráticos para problemas de regressão ou classificação
* Engenharia de Recursos menos necessária do que modelos lineares, mas ainda útil
* Entenda como reduzir o ruído

**Modelo de Aprendizado Profundo Wide & Deep**

* O modelo wide é usado para memorização, enquanto o modelo deep é utilizado para generalização
* Usado para sistemas de recomendação, busca e problemas de classificação

**Treinamento Online e/ou Aprendizado Contínuo**

* Construa uma pipeline para continuar treinando seu modelo com base em dados novos e antigos

**ML Efetivo:**

* Coleta de Dados
* Organização de Dados
* Modelagem usando intuição humana e conhecimento de domínio
* Use hackathons para desenvolver o modelo a partir dos dados de entrada
* Seu conjunto de dados deve cobrir todos os casos, tanto positivos quanto negativos, e deve ter pelo menos 5 casos de cada onde o modelo consiga classificar corretamente aquele caso. Erros são quase sempre importantes para investigação. Explore seus dados, encontre as causas dos problemas e tente corrigi-los. Se não puder ser corrigido, tente encontrar mais 'casos ruins' para tentar treinar seu modelo com eles ou, caso contrário, remova esses dados do conjunto.

**Machine Learning no GCP**

* TensorFlow - para pesquisador de ML, use SDK
* CloudML - para Cientista de Dados, use modelo customizado, escalável, sem operações (onde houver dados suficientes para treinar um modelo)
* APIs de ML - para Desenvolvedor de Aplicações, use modelos pré-construídos, por exemplo, visão, fala, linguagem (onde precisaria de mais dados para treinar um modelo)

**5.3. ML Engine**

* Treine e preveja modelos de ML
* Pode usar múltiplas plataformas como TensorFlow, scikit-learn e XGBoost

**Cluster de Treinamento:**

* O serviço de treinamento aloca os recursos para as máquinas do tipo que você especificar
* Réplica: Seu trabalho em execução em um nó é chamado de réplica.
  + Cada réplica no cluster de treinamento é atribuída um papel ou tarefa específica no treinamento distribuído:
    - mestre: Exatamente uma réplica é designada como mestre. Essa tarefa gerencia as demais e reporta o status do trabalho como um todo.
    - workers: Uma ou mais réplicas podem ser designadas como trabalhadoras. Essas réplicas executam sua porção do trabalho conforme você designar em sua configuração de trabalho.

**Previsão Online versus em Lote**

* Online
  + Otimizado para minimizar a latência das previsões de servir.
  + Previsões retornadas na resposta da mensagem.
  + Retorna o mais rápido possível.
* Em Lote
  + Otimizado para lidar com um alto volume de instâncias em um trabalho e para executar modelos mais complexos.
  + Previsões escritas em arquivos de saída em um local de armazenamento do Cloud Storage que você especificar.
  + Requisição assíncrona.

**Exceção**

* O serviço de treinamento executa seu trabalho até que as threads encontrem um erro irrecuperável.
* No caso distribuído, é o status do mestre no Cloud ML que sinaliza o status geral do trabalho.
* Executar treinamento do ML Engine localmente (gcloud ml-engine local train) é especialmente útil no caso de testar modelos distribuídos.

**5.4. TensorFlow**

* Plataforma de Aprendizado de Máquina/Aprendizado Profundo de SO
* Avaliação preguiçosa durante a construção, execução completa
* suas mudanças de dados frequentemente e você quer priorizar a frescura dos dados sobre o desempenho.
* você pode usar um cache de dados que incorre em custos (por exemplo, BigQuery) e quer minimizar esses custos.

**Tensorflow**

* + Biblioteca de aprendizado de máquina
  + Usado em muitos produtos do Google
  + Engine em C++ e API (pode rodar em GPUs)
  + API em Python (para facilitar a escrita de código)
  + Para usar o Tensorflow:
  + Colete preditores e dados-alvo
  + Descarte as informações que identificam uma linha (precisa ter pelo menos 5-10 exemplos de um valor particular - para evitar overfitting)
  + Dados preditores devem ser numéricos (não categóricos / colunas)
  + Engenharia de Características
  + Quantos nós e camadas precisamos?
  + Treinar o modelo baseado em dados de entrada
  + O modelo de regressão prevê um número
  + O modelo de classificação prevê uma categoria
  + Use o modelo em novos dados

**Engenharia de Características**

* + Um vetor esparso com muitos zeros, contém apenas um único 1
  + Se você não souber o conjunto de valores possíveis antecipadamente, você pode usar column\_with\_hash\_bucket em vez disso.
  + Um embedding é um mapeamento de objetos discretos, como palavras, para vetores de números reais.
* **5.5. Datalab**
* Datalab: notebooks Jupyter gerenciados para uso com um cluster dataproc para escrever trabalhos pyspark
* Como executar o Datalab
* notebook open source construído em Jupyter
* usar pacotes Python existentes

**6. Gestão**

**6.1. IAM & Faturamento**

* 3 tipos de membros: Conta de Serviço, conta google e grupo google.
  + Contas de Serviço são para usuários não humanos, como aplicações
  + Contas Google são para usuários únicos
  + Grupos Google são para múltiplos usuários
* Taxas de transferência de projeto estão associadas com o iniciador
* O acesso ao faturamento pode ser fornecido a um projeto ou conjunto de projetos sem conceder acesso ao conteúdo. Isso é útil para separação de funções entre finanças/desenvolvimento etc.
* Como o faturamento funciona entre projetos

**6.2. Stackdriver**

* Para armazenar, pesquisar, analisar, monitorar e alertar sobre dados e eventos.
* Certifique-se de conhecer os subprodutos do Stackdriver (Debugger, Error Reporting, Alerting, Trace, Logging), o que eles fazem e quando devem ser usados.
* Serviço híbrido de monitoramento.
* Como você pode usar o Stackdriver para monitorar e registrar o uso.
* Como usar o Stackdriver para ajudar a depurar o código
* Auditoria de Logs para revisar o acesso aos dados (por exemplo, BigQuery)
* Monitoramento do Stackdriver
  + pode ver o uso dos slots de consulta do BigQuery
* Stackdriver Trace
  + é um sistema de rastreamento distribuído para a plataforma Google Cloud que coleta dados de latência do App Engine, balanceadores de carga HTTPS do Google e aplicações instrumentadas com as SDKs do Stackdriver Trace, e mostra em tempo real na consola Google Cloud

**6.3. Data Studio 360**

* Painel de Dados
  + pode usar os dados existentes do YouTube como fonte
* O cache de pré-busca só está ativo para fontes de dados que usam as credenciais do proprietário para acessar os dados subjacentes.
* Desativar o cache de consulta pode resultar em maiores custos de uso de dados para fontes pagas, como BigQuery
* Você pode desativar o cache de pré-busca para um relatório específico. Você pode querer fazer isso se:

**6.4. Cloudshell**

* VM temporária
* reciclada a cada 60 minutos (aproximadamente)

**6.5. Cloud Deployment Manager**

* Permite especificar todos os recursos necessários para sua aplicação em um formato declarativo usando yaml
* Processo de Implantação Repetível

**6.6. Data Transfer**

* Transferência de Dados de Armazenamento
  + importar serviço online para o Cloud Storage.
  + agenda de repetição
  + transferência de dados dentro do Cloud Storage, de um bucket para outro.
  + Fonte: GCS, S3, URL
* Appliance de Transferência
  + uso único
  + Rack, capture e então envie seus dados offline para o Google Cloud
* Se você tem um grande número de arquivos para transferir, pode querer usar a opção gsutil -m para realizar uma cópia paralela (multi-threaded/multi-processamento)

**Comprimindo e combinando arquivos menores em poucos arquivos maiores também é uma melhor prática para aumentar a velocidade de transferência**

**Formato de Dados Avro^5**

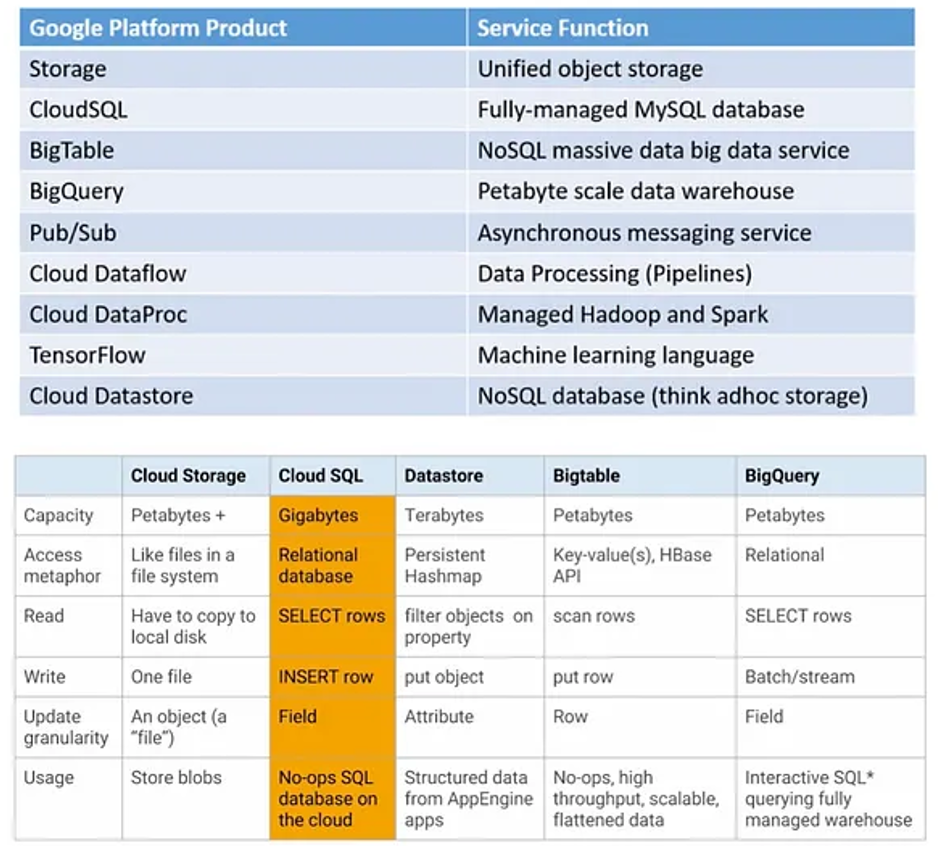
* É mais rápido para carregar. Os dados podem ser lidos em paralelo, mesmo se os blocos de dados estiverem comprimidos.
* Não requer tipagem ou serialização.
* É mais fácil de analisar porque não há questões de codificação como ASCII.
* Arquivos Avro comprimidos não são suportados, mas os blocos de dados são. BigQuery suporta os codecs DEFLATE e Snappy.

**7. Seleção de Produto**

* se há um requisito para pesquisar terabytes - petabytes de dados relativamente rápido, faz mais sentido simplesmente armazenar no BigQuery (comparável ao AWS Redshift).
* Para o DataStore, há uma possibilidade de que isso possa funcionar como uma substituição para Cassandra.
* É mais provável que BigTable seja uma solução melhor
  + se o conjunto de dados for relativamente pequeno <10TB, então o DataStore será preferido.
  + se o conjunto de dados for >10TB e/ou não houver requisito para múltiplos índices, então o BigTable será melhor.
* Esteja ciente de quaisquer limitações sobre índices e particionamento,

**Buscando por objetos pelo valor do atributo - Datastore (apenas escrita por chave de linha única)**

* Alta produtividade de escrita de dados de valor único - Bigtable
* Armazenamento de dados estruturados - BigQuery



**8. Estudo de Caso**

Existem 2 estudos de caso que são iguais aos do site do GCP: um estudo de caso logístico da Flowlogistic e um estudo de caso de comunicações de hardware da MJTelco. Cada estudo de caso inclui questões que pedem que se pense em formas como cada empresa poderia usar as atuais tecnologias do GCP para alcançar os melhores resultados. Esses estudos de caso estão disponíveis na LinuxAcademy.

A resposta correta não é necessariamente a melhor solução técnica, mas sim a que proporciona o melhor resultado para a empresa dadas suas limitações atuais.^7

Flowlogistic: <https://cloud.google.com/certification/guides/data-engineer/casestudy-flowlogistic>

* para encontrar uma maneira de armazenar dados que podem ser comumente usados pelo DataProc e BigQuery
* Segundo com DataProc e BigQuery interagindo com o Cloud Storage

MJTelco: <https://cloud.google.com/certification/guides/data-engineer/casestudy-mjtelco> Desconstruindo um Estudo de Caso: Data Engineer Exam: <https://www.youtube.com/watch?v=_yrYDysfB-k>

Outro caso útil: Spotify's Event Delivery – A Estrada para a Nuvem: <https://labs.spotify.com/2016/02/25/spotifys-event-delivery-the-road-to-the-cloud-part-i/>

**9. Recursos e Referências**

Recursos:

* Folha de Dicas de Engenharia de Dados do Google: <https://github.com/ml874/Data-Engineering-on-GCP-Cheatsheet>
* Roadmap de Engenharia de Dados: <https://github.com/hasbrain/data-engineer-roadmap>
* Exame de Simulação de Whizlab: [https://www.whizlabs.com](https://www.whizlabs.com/)
* Deixe-me saber se alguma fonte estiver faltando

Youtube Videos para Assistir: Auto-awesome: ciência de dados avançada na Google Cloud Platform (Google Cloud Next '17): <https://www.youtube.com/watch?v=Jp-gJFJ9fWw&list=PLIivdWyY5sqLq-eM4W2bIgbrpAs5aLZt>

Introdução ao Google Cloud Machine Learning (Google Cloud Next '17): <https://www.youtube.com/watch?v=COSqX5HKaO4>

Introdução às ferramentas do Google Cloud Next '17 que proporcionam insights profundos (Google Cloud Next '17): <https://www.youtube.com/watch?v=ld2rPHJMsZg>

Webinar: Construindo uma pipeline de análise em tempo real com o BigQuery e o Cloud Dataflow (EMEA): <https://www.youtube.com/watch?v=kdmAiQeYGAe>

^7 <https://www.linkedin.com/pulse/google-cloud-certified-professional-data-engineer-writeup-rix/>