FIAP GRADUAÇÃO

FIAP

# DATA SCIENCE

## BIG DATA ARCHITECTURING & DATA INTEGRATION

Prof. Dr. Renê de Ávila Mendes

# Objetivos da disciplina

**DISCIPLINA:** Big Data Architecturing & Data Integration

**OBJETIVOS:** Entenda as principais **arquiteturas** para ingestão, processamento e análise de grandes volumes de dados. Conheça as principais **ferramentas** open-source de Big Data como Hadoop, MapReduce, Spark, Sqoop, NiFi, Flume, Kafka, Zookeeper, HBase, Hive e as **integre** com as ferramentas de **extração, transformação e carga** de dados em modelos dimensionais. Entenda conceitos sobre **computação paralela e distribuída**, aplicação do Hadoop e bases Apache e arquiteturas *serverless* e desacopladas. Veja como **visualizar os dados** estruturados ou não estruturados com ferramentas de Self-Service Business Intelligence como PowerBI, utilizando as melhores práticas de **visualização de dados**.

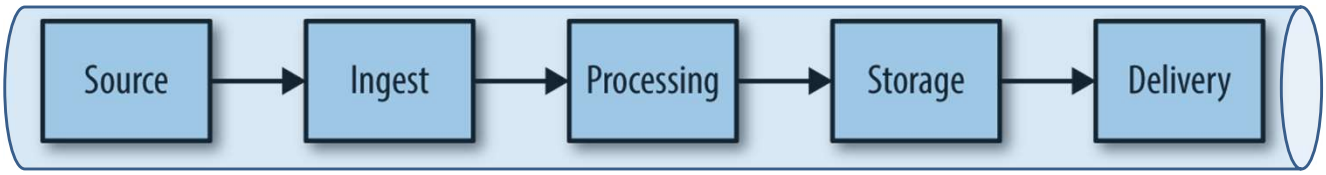
# Assuntos – 2º Semestre

- Arquiteturas para Big Data
- Data Pipelines
- Conceito de Data Lake
- PIG, HIVE, Spark, Data Streaming

# Objetivo dessa aula

**OBJETIVOS:** Entender o funcionamento da arquitetura de Big Data Lambda.

# O pipeline de dados analíticos

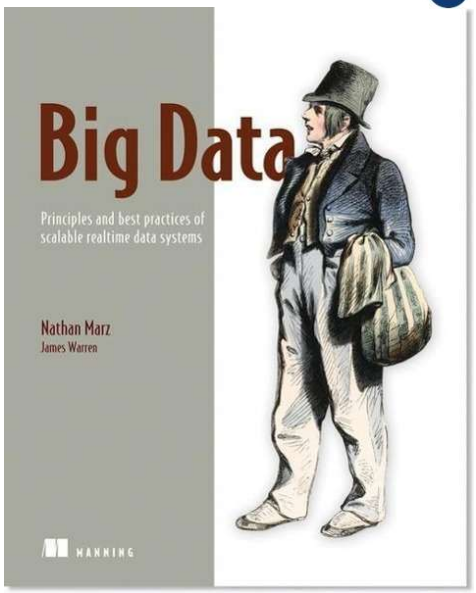



Os dados não se formatam por si mesmos para fins de análise. Eles passam por uma série de passos que envolvem a coleta a partir das origens, o tratamento para as formas necessárias à análise e finalmente sua disponibilização na forma de resultados para serem consumidos. Estes passos podem ser pensados como um “tubo”.
Este modelo ajuda a entender onde aplicar cada tecnologia.

TEJADA, Zoiner. **Mastering Azure Analytics: Architecting in the Cloud with Azure Data Lake, HDInsight, and Spark**." O'Reilly Media, Inc., 2017.

# Big Data – Arquitetura Lambda

- Marz e Warren (2013)
- Implementa funções arbitrárias em um conjunto arbitrário de dados
- Resposta em uma baixa latência pela combinação de diferentes ferramentas e técnicas
- Divide o sistema em três camadas dependentes e complementares




MARZ, N.; WARREN, J. Big Data: Principles and best practices of scalable realtime data systems. [S.l.]: Manning Publications Co., 2015.

## Lambda – Premissa

Se um sistema de dados qualquer responde perguntas consultando os dados, então o sistema mais genérico pode responder questões consultando todos os dados.

A arquitetura Lambda é a proposta de Marz e Warren (2015) para a implementação de funções arbitrárias em um conjunto arbitrário de dados (pesquisa = função(todos os dados)) obtendo uma resposta em uma baixa latência pela combinação de diferentes ferramentas e técnicas. A arquitetura foi proposta em 2013 por Nathan Marz sendo o primeiro padrão arquitetônico para processamento em lote e em tempo real.

MARZ, N.; WARREN, J. *Big Data: Principles and best practices of scalable realtime data systems*. [S.l.]: Manning Publications Co., 2015.

## Lambda – Premissa

Se um sistema de dados qualquer responde perguntas consultando os dados, então o sistema mais genérico pode responder questões consultando todos os dados.

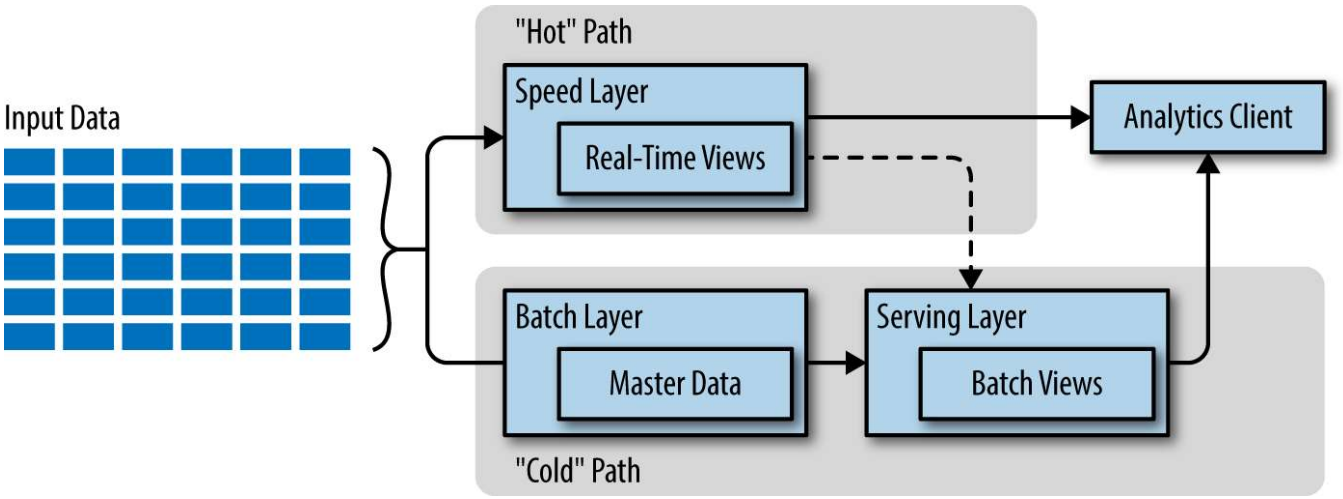
A definição genérica para um sistema de dados pode ser expressada como:

***consulta = função(todos os dados)***

O princípio de funcionamento da arquitetura Lambda é a divisão de um sistema Big Data em três camadas dependentes e complementares: camada batch, camada serving e camada speed. Cada camada atende a um conjunto de propriedades e é construída sobre as funcionalidades da camada mais próxima.

MARZ, N.; WARREN, J. *Big Data: Principles and best practices of scalable realtime data systems*. [S.l.]: Manning Publications Co., 2015.

# Lambda – Visão geral




Na arquitetura Lambda há dois caminhos para o dado fluir pelo “tubo” de dados analítico:

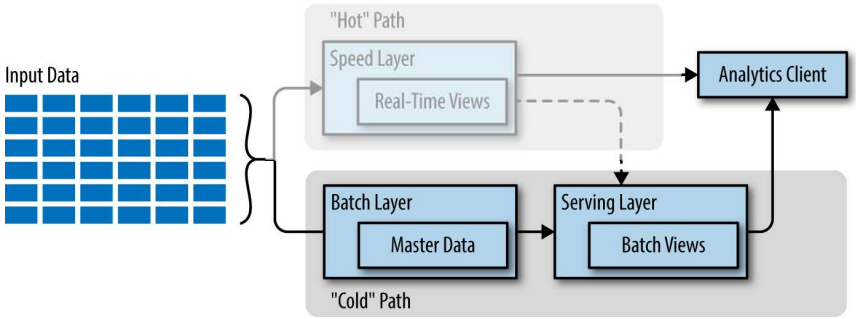
- Um caminho quente (hot path) no qual dados sensíveis a latência fluem para serem rapidamente consumidos por clientes analíticos;
- Um caminho frio (cold path) pelo qual todos os dados fluem e são processados em lotes que podem tolerar grandes latências até que o resultado seja produzido.

Em certas implementações a camada serving pode hospedar dados das visões real-time e das visões batch.

TEJADA, Zoiner. **Mastering Azure Analytics: Architecting in the Cloud with Azure Data Lake, HDInsight, and Spark**." O'Reilly Media, Inc., 2017.

# Lambda – Batch – Cold path

- Dados imutáveis (mudanças somente por novas entradas)
- Permite recomputação de qualquer ponto no tempo
- Resultados extremamente precisos
- Alta latência

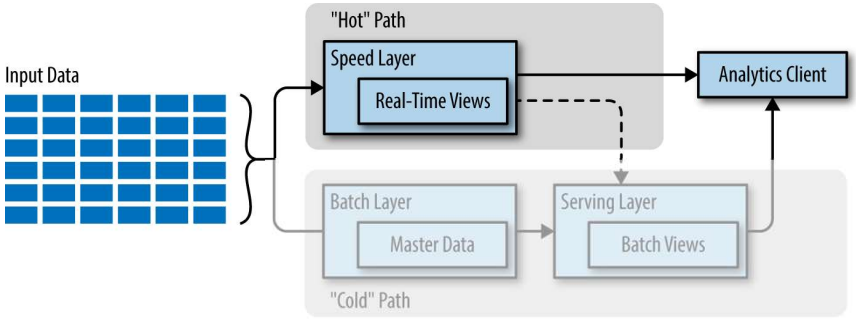



Quando o dado flui pelo caminho frio este dado é imutável. Quaisquer mudanças no valor dos dados são refletidas por novos dados, com um atributo temporal (timestamp), sendo armazenado no sistema junto com os valores antigos. Esta abordagem permite ao sistema recomputar o então atual valor para qualquer ponto no tempo dos dados coletados.  Como o caminho frio pode tolerar uma grande latência até que os resultados sejam produzidos, a computação pode ser feita em grandes conjuntos de dados, e o tipo de cálculo pode ser demorado. O objetivo do caminho frio pode ser sumarizado como: tome o tempo que precisar, mas devolva um resultado extremamente acurado.

TEJADA, Zoiner. **Mastering Azure Analytics: Architecting in the Cloud with Azure Data Lake, HDInsight, and Spark**." O'Reilly Media, Inc., 2017.

# Lambda – Stream – Hot path

- Dados mutáveis (podem ser corrigidos na hora)
- Menos acurácia em troca de respostas rápidas

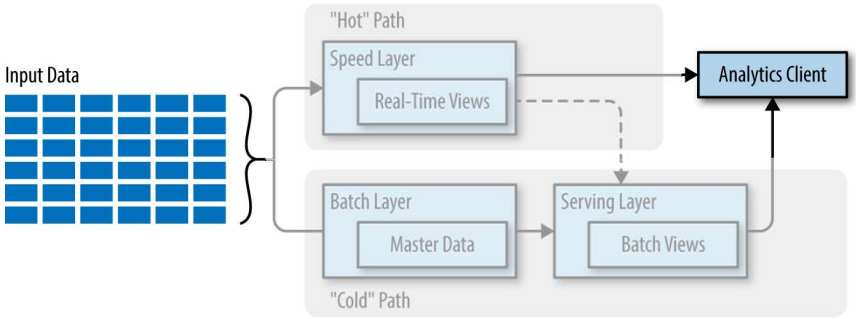



Quando o dado flui pelo caminho quente, estes dados são mutáveis e podem ser atualizados na hora. Neste caminho há uma restrição de latência para os dados, uma vez que os resultados são esperados em tempo quase real. O impacto desta restrição de latência é que os tipos de cálculos que podem ser executados são aqueles que podem ser executados rápido o suficiente. Isto pode implicar a troca de um algoritmo que ofereça acurácia perfeita por um que ofereça um resultado aproximado. O objetivo no caminho quente pode ser resumido assim: negociar um pouco de acurácia nos resultados para garantir que o resultado seja obtido rápido o suficiente.

TEJADA, Zoiner. **Mastering Azure Analytics: Architecting in the Cloud with Azure Data Lake, HDInsight, and Spark**." O'Reilly Media, Inc., 2017.

# Lambda – Cliente

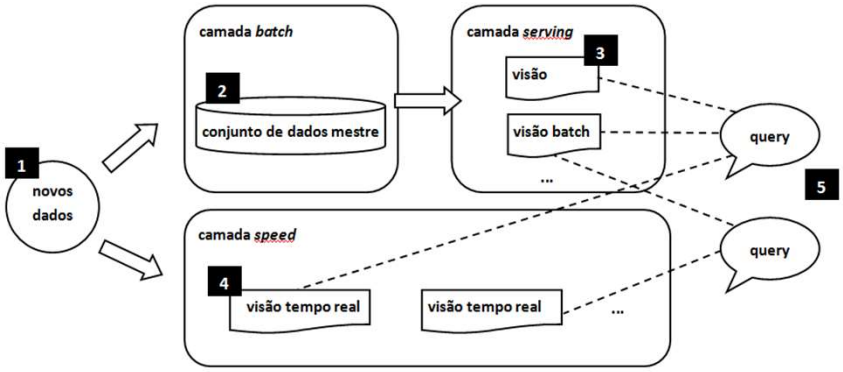
- Define o critério da obtenção do resultado
- Se tiver mais tempo, escolhe o resultado mais preciso
- Se não puder esperar, pesquisa por um resultado menos preciso e que represente um intervalo de tempo menor




Os caminhos frio e quente convergem finalmente para uma aplicação analítica cliente. O cliente precisa escolher o caminho a partir do qual ele obterá o resultado. Ele pode escolher obter dados menos precisos mas mais atualizados a partir do caminho quente ou ele pode usar um resultado mais antigo mas mais preciso a partir do caminho frio. Um importante componente da decisão relaciona-se à janela de tempo para a qual somente o caminho quente tem o resultado, uma vez que o caminho frio ainda não computou seu resultado. Olhando por este ponto de vista, o caminho quente tem resultados por somente uma pequena janela de tempo e seus resultados são finalmente atualizados pelos resultados mais precisos obtidos pelo caminho frio. Esta abordagem tem o efeito de minimizar o volume de dados com o qual o caminho quente precisa lidar.

TEJADA, Zoiner. **Mastering Azure Analytics: Architecting in the Cloud with Azure Data Lake, HDInsight, and Spark**." O'Reilly Media, Inc., 2017.

# Lambda - Resumo




*visão batch = função(todos os dados)*
*visão tempo real = função(visão tempo real, novos dados)*
*query = função(visão batch, visão tempo real)*

HAUSENBLAS, M.; BIJNENS, N. What is the Lambda Architecture? http://lambda-architecture.net

(1) todos os dados que entram no sistema vão para as camadas *batch* e *speed* para serem processados;
(2) a camada *batch* é responsável por gerenciar o conjunto de dados mestre (conjunto de dados imutáveis e crescentes) e por pré-processar as visões *batch*;
(3) a camada *serving* indexa o conteúdo das views *batch* para ser acessado pelas *queries*;
(4) a camada *speed* lida com dados recentes e complementa o resultado da camada *batch*;
(5) as consultas obtêm dados das visões *batch* e das visões de tempo real

## Como Lambda atende os requisitos sistêmicos

### 1. Robustez e tolerância a falhas

A camada *serving* usa a replicação para garantir sua disponibilidade, enquanto as camadas *batch* e *serving* são tolerantes a falhas humanas porque permitem a correção do algoritmo, a exclusão dos dados com erro e o reprocessamento das visões.

MARZ, N.; WARREN, J. *Big Data: Principles and best practices of scalable realtime data systems*. [S.l.]: Manning Publications Co., 2015.

## Como Lambda atende os requisitos sistêmicos

**2. Baixa latência de leitura e atualização**

A camada *speed* é escalável e permite leitura de dados em tempo real e quase real.

**3. Escalabilidade**

O escalamento das camadas *batch* e *serving* pode ser feito adicionando novos nós ao *cluster* de servidores.

MARZ, N.; WARREN, J. *Big Data: Principles and best practices of scalable realtime data systems*. [S.l.]: Manning Publications Co., 2015.

## Como Lambda atende os requisitos sistêmicos

### 4. Generalização

A arquitetura permite a computação de consultas arbitrárias em conjuntos arbitrários de dados.

### 5. Extensibilidade

O modelo de dados do conjunto de dados mestre permite a adição de novos tipos de dados, e a computação de novas visões é tão simples quanto a adição de novos tipos de dados.

MARZ, N.; WARREN, J. *Big Data: Principles and best practices of scalable realtime data systems*. [S.l.]: Manning Publications Co., 2015.

## Como Lambda atende os requisitos sistêmicos

### 6. Suporte a consultas *ad hoc*

São amplamente suportadas pela camada *batch* porque todos os dados são armazenados em um mesmo local.

### 7. Mínima manutenção

Os bancos de dados da camada *serving* não necessitam fazer escritas randômicas, diminuindo sua complexidade.

MARZ, N.; WARREN, J. *Big Data: Principles and best practices of scalable realtime data systems*. [S.l.]: Manning Publications Co., 2015.

# Como Lambda atende os requisitos sistêmicos

## 8. Capacidade de correção (“*debuggability*”)

Como as entradas e saídas de dados não se sobrepõem umas às outras, é possível identificar e corrigir falhas nesses pontos.

MARZ, N.; WARREN, J. *Big Data: Principles and best practices of scalable realtime data systems*. [S.l.]: Manning Publications Co., 2015.

# Lambda – Críticas à arquitetura

**Acurácia do processamento *stream (hot path)***

O processamento em tempo real não necessariamente será aproximado, menos poderoso e mais sujeito a perdas. Os *frameworks* de processamento em tempo real são menos maduros, mas oferecem a mesma garantia semântica que os *frameworks* de processamento em lote.

KREPS, J. *Questioning the Lambda Architecture: The Lambda Architecture has its merits, but alternatives are worth exploring.* 2014. Disponível em: <https:/-/www.oreilly.com/ideas/questioning-the-lambda-architecture>. Acesso em: 11 Junho 2016.

# Lambda – Críticas à arquitetura

**Complexidade de manutenção de código das camadas *batch* e *speed***

A lógica da transformação tem que ser implementada duas vezes, uma para cada camada. A separação do processamento em duas camadas implica o uso de diferentes *frameworks* distribuídos, tais como Hadoop e Storm, que possuem suas especificidades de codificação. Recomenda-se a adoção de um ou de outro processamento: em lote caso não haja sensibilidade à latência, ou em tempo real, caso a baixa latência seja um requisito imprescindível.

KREPS, J. *Questioning the Lambda Architecture: The Lambda Architecture has its merits, but alternatives are worth exploring.* 2014. Disponível em: <https:/-/www.oreilly.com/ideas/questioning-the-lambda-architecture>. Acesso em: 11 Junho 2016.

# Lambda – Críticas à arquitetura

## Integração de resultados

Como o resultado final é uma composição de resultados de visões *batch* e *speed*, a **sincronização** entre estas duas visões torna-se um fator chave para que se evitem resultados redundantes ou ausentes. O armazenamento dos resultados das camadas *batch* e *speed* em repositórios diferentes deixa o sistema em um estado conhecido como **persistência poliglota**, exigindo a agregação dos resultados das views para atendimento das consultas.

Na arquitetura Lambda, a camada *batch* é responsável por processar todo o conjunto de dados para gerar as visões *batch* enquanto que à camada *speed* cabe a responsabilidade de processar os novos dados que chegam ao sistema gerando a partir destes as visões *speed*. As pesquisas são feitas numa composição das visões *batch* e *speed*. Tão logo o dado é processado na camada *batch* o resultado é armazenado nas visões *batch* e retirado das visões *speed.*

Como solução para os problemas de sincronização e agregação, Vanhove propõe que os dados que chegam à camada speed sejam identificados com uma etiqueta (tag), chamada de Tn. O processamento desta camada gera como resultado (visão speed)Tn. Quando o processamento da camada batch é concluído, os dados resultantes são movidos para a visão batch e o sistema muda para uma nova etiqueta, Tn+1, para todos os novos dados. Os dados de (visão speed)Tn-1 são então apagados.

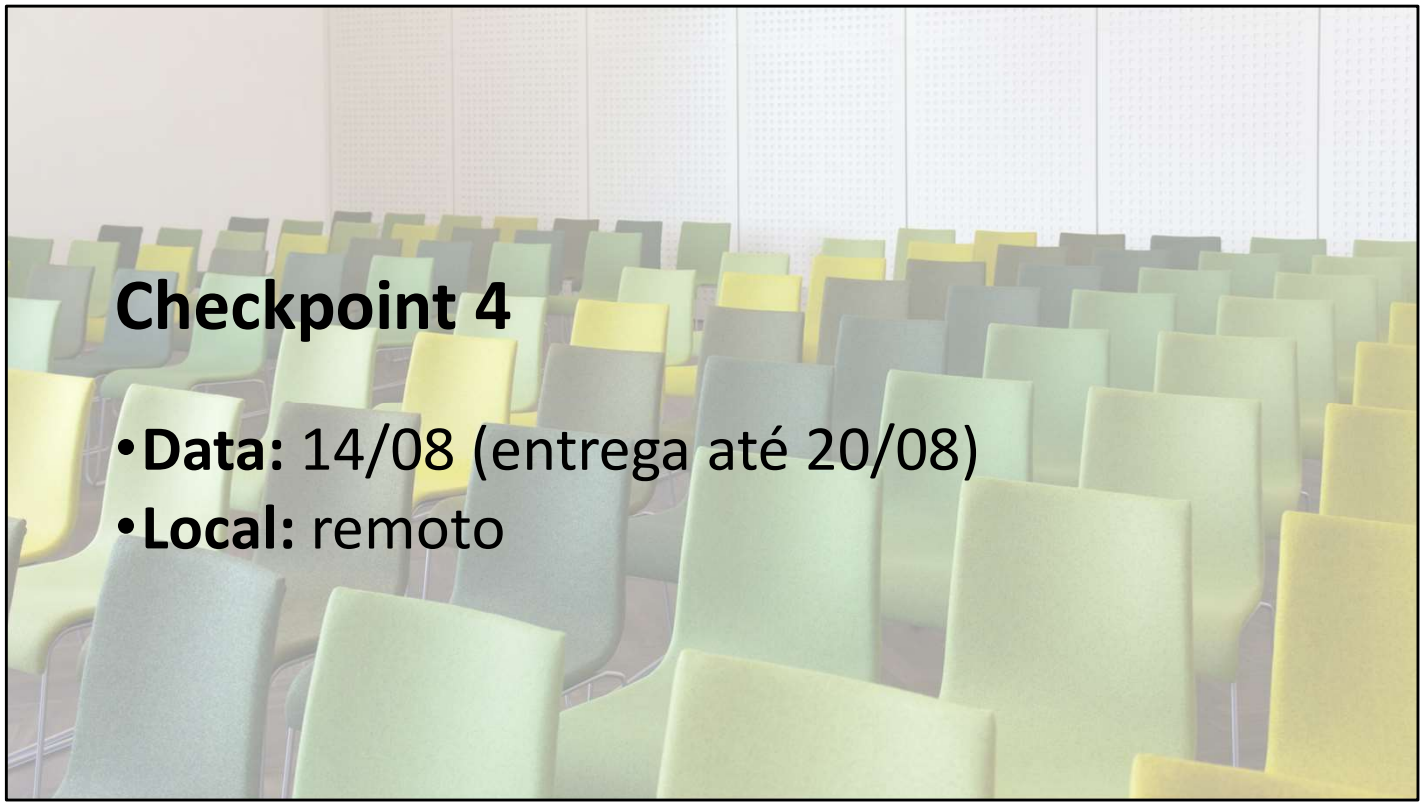
VANHOVE, T. et al. Managing the synchronization in the lambda architecture for optimized big data analysis. *IEICE Transactions on Communications*, The Institute of Electronics, Information and Communication Engineers, v. 99, n. 2, p. 297–306, 2016.

# Referências e leituras recomendadas

TEJADA, Zoiner. **Mastering Azure Analytics: Architecting in the Cloud with Azure Data Lake, HDInsight, and Spark**." O'Reilly Media, Inc., 2017.

https://learn.microsoft.com/en-us/azure/architecture/data-guide/big-data/


CHECKPOINT 4



## Checkpoint 4

- **Data:** 14/08 (entrega até 20/08)
- **Local:** remoto