

SESIÓN 0: FOUNDATION SETUP

Slides de Teoría - 15 minutos (15 slides × 1 min c/u)

CONCEPTOS FUNDAMENTALES QUE DEBEN SABER

Pre-requisitos Básicos:

- ¿Qué es un servidor/computadora?
- ¿Qué es una aplicación/software?
- Diferencia entre almacenar vs procesar datos
- Concepto básico de "escalabilidad"

Conceptos Nuevos a Introducir:

- **Sistemas Distribuidos:** Múltiples computadoras trabajando juntas
 - **Containerización:** Docker como "caja" con todo incluido
 - **Cluster:** Grupo de computadoras coordinadas
 - **APIs:** Cómo se comunican las aplicaciones
 - **Tolerancia a Fallos:** Sistema sigue funcionando aunque algo falle
-



SLIDE 1: EL DESAFÍO - "El Muro Invisible de la Tecnología"

Imagen de Referencia: Persona frente a una pared enorme vs persona con escalera

TÍTULO: "El Desafío que Cambió el Mundo Tech"

EL DESAFÍO:  **MURO APARENTEMENTE IMPOSIBLE** Los datos crecían más rápido que la tecnología para procesarlos

- 2010: 1TB/día
- 2024: 10,000TB/día
- Una computadora → Ya no era suficiente"

PORQUÉ IMPORTA:  **IMPACTO REAL:** Netflix ahorra \$824 millones anuales  **BREAKTHROUGH:** 1000 computadoras pequeñas > 1 supercomputadora"

CÓMO NACIÓ LA SOLUCIÓN:  **CAMBIO DE MENTALIDAD:** En lugar de hacer 1 computadora más poderosa... ¿Y si hacemos que 1000 computadoras trabajen juntas?"

EVOLUCIÓN:  **DE ESTO NACIÓ:** Google, Facebook, Netflix, Uber - todo lo que usas diariamente"

 **MENSAJE DE ÁNIMO:** "¡Hoy VAS A DOMINAR la mentalidad que creó estas empresas gigantes!"

SLIDE 2: ¿QUÉ SON LOS SISTEMAS DISTRIBUIDOS?

Imagen de Referencia: Diagrama simple de red de computadoras conectadas

TÍTULO: "Sistemas Distribuidos: Trabajar en Equipo"

ANALOGÍA VISUAL:

- Un chef cocinando para 1000 personas (imposible)
- vs 10 chefs coordinados cocinando juntos (posible)

DEFINICIÓN SIMPLE: "Sistema Distribuido = Múltiples computadoras que trabajan juntas como si fueran una sola"

EJEMPLOS COTIDIANOS:

- Google Search: Miles de servidores respondiendo tu búsqueda
- WhatsApp: Mensajes viajando por múltiples países
- Netflix: Videos almacenados en cientos de ubicaciones

KEY INSIGHT: "Tu celular ya usa sistemas distribuidos todo el tiempo"

SLIDE 3: ¿POR QUÉ NECESITAMOS ESTO?

Imagen de Referencia: Gráfico de crecimiento exponencial de datos

TÍTULO: "El Problema: Los Datos Crecen Más Rápido que las Computadoras"

VISUAL: Gráfico mostrando:

- 2010: 1GB por día
- 2015: 1TB por día
- 2020: 1PB por día
- 2024: 10PB por día

REALIDADES:

- Facebook: 4 mil millones de likes por día
- YouTube: 720,000 horas de video subidas por día
- Amazon: 66,000 pedidos por hora

CONCLUSIÓN: "Una sola computadora ya no es suficiente"

SLIDE 4: CONCEPTOS CLAVE - ESCALABILIDAD

Imagen de Referencia: Diagrama de escalabilidad vertical vs horizontal

TÍTULO: "Dos Formas de Crecer: Vertical vs Horizontal"

ESCALABILIDAD VERTICAL (Scale Up):

- Imagen: Computadora con más RAM, CPU más potente
- "Hacer UNA computadora más poderosa"
- Problema: Tiene límites físicos y es muy caro

ESCALABILIDAD HORIZONTAL (Scale Out):

- Imagen: Muchas computadoras pequeñas conectadas
- "Agregar MÁS computadoras"
- Ventaja: Sin límites, más barato

ANALOGÍA: "Vertical = Un camión más grande Horizontal = Más camiones pequeños"

BIG DATA CHOICE: "Siempre elegimos horizontal"

SLIDE 5: ¿QUÉ ES DOCKER?

Imagen de Referencia: Contenedores de barco vs containers Docker

TÍTULO: "Docker: Como Contenedores de Barco, pero para Software"

ANALOGÍA VISUAL:

- Barco de carga con contenedores estandarizados
- Cada contenedor tiene todo lo necesario dentro
- Se pueden mover a cualquier barco/puerto

DOCKER = CONTENEDOR DE SOFTWARE:

- Contiene la aplicación + todo lo que necesita
- Funciona igual en cualquier computadora
- Se puede mover, copiar, escalar fácilmente

BENEFICIOS PRÁCTICOS:

- "Funciona en mi máquina" → "Funciona en cualquier máquina"
- Setup automático sin configuraciones manuales
- Cada servicio aislado (no se interfieren)

HOY USAREMOS: Docker para crear nuestro cluster Big Data

SLIDE 6: ¿QUÉ ES UN CLUSTER?

Imagen de Referencia: Diagrama de cluster con nodos coordinados

TÍTULO: "Cluster: Un Equipo de Computadoras Coordinadas"

VISUAL: Diagrama mostrando:

- Nodo Master (coordinador)
- Nodos Workers (ejecutores)
- Red que los conecta
- Trabajo distribuido entre todos

ANALOGÍA: "Como una orquesta"

- Director (Master): Coordina todo
- Músicos (Workers): Ejecutan la música
- Partitura (Algoritmo): Instrucciones para todos

COMPONENTS EN NUESTRO CLUSTER:

- HDFS: Para almacenar datos
- Spark: Para procesar datos
- Jupyter: Para escribir código
- Kafka: Para datos en tiempo real (después)

SLIDE 7: HITO 1 - MENTAL MODEL SHIFT

Imagen de Referencia: Cerebro con dos caminos: viejo vs nuevo pensamiento

TÍTULO: "🎯 HITO 1: Tu Primera Transformación Mental"

EL DESAFÍO: "😞 **PROBLEMA MENTAL:** Estamos programados para pensar 'una cosa a la vez'

- 1 persona cocina para 1000 → Imposible
- 1 computadora procesa petabytes → Imposible"

PORQUÉ CAMBIAR: "🧠 **BREAKTHROUGH MENTAL:** Los problemas grandes se pueden dividir

- 10 chefs cocinando juntos → Posible
- 1000 computadoras procesando juntas → ¡El futuro!"

CÓMO EVOLUCIONA TU MENTE:

ANTES: "¿Cómo hago que UNA computadora vaya más rápido?"

DESPUÉS: "¿Cómo divido el problema para MUCHAS computadoras?"

EVOLUCIÓN DEL PENSAMIENTO: "🌟 **NIVEL DESBLOQUEADO:** Ahora piensas como ingenieros de Google, Netflix, Amazon"

👉 **MENSAJE DE ÁNIMO:** "¡FELICITACIONES! Acabas de cambiar tu forma de pensar sobre tecnología. Este cambio mental vale más que cualquier título universitario. ¡Sigues creciendo hacia ingeniero de clase mundial! 🚀"

SLIDE 8: CAP THEOREM - LA REALIDAD DE LOS SISTEMAS DISTRIBUIDOS

Imagen de Referencia: Triángulo CAP con ejemplos reales

TÍTULO: "CAP Theorem: No Puedes Tenerlo Todo"

VISUAL: Triángulo con vértices:

- **Consistency** (Consistencia): Todos ven los mismos datos
- **Availability** (Disponibilidad): Sistema siempre funcionando
- **Partition Tolerance** (Tolerancia a fallos): Funciona aunque fallen conexiones

LA REGLA: "Solo puedes elegir 2 de 3"

EJEMPLOS REALES:

- **Google Search** (A + P): Siempre disponible, resultados pueden variar
- **Bancos** (C + P): Datos exactos, puede estar temporalmente no disponible
- **Facebook** (A + C): Disponible + consistente en una región

TU DECISIÓN: "Como ingeniero, TÚ decides qué sacrificar según el caso"

SLIDE 9: DATA LOCALITY - EL SECRETO DE LA PERFORMANCE

Imagen de Referencia: Diagrama mostrando movimiento de datos vs código

TÍTULO: "Data Locality: Mueve el Código, No los Datos"

ANTI-PATTERN (LO QUE NO HACER):

- Imagen: Datos en servidor A, procesamiento en servidor B
- Flecha gigante: 10GB de datos viajando por la red
- Resultado: CUELLO DE BOTELLA

PRO PATTERN (LO CORRECTO):

- Imagen: Código pequeño (KB) viajando a cada servidor
- Procesamiento local en cada servidor con sus datos
- Solo resultados pequeños (MB) se combinan

IMPACTO REAL:

- Netflix: Reduce tráfico de red en 1000x
- Google: Puede procesar todo internet por este principio
- Amazon: Millones de transacciones/segundo

APLICACIÓN: "Todo lo que aprenderás (HDFS, Spark, Kafka) usa este principio"

SLIDE 10: IDEMPOTENCY - SISTEMAS CONFIABLES

Imagen de Referencia: Botón que se puede presionar múltiples veces sin efecto negativo

TÍTULO: "Idempotencia: Operaciones Seguras de Repetir"

DEFINICIÓN SIMPLE: "Ejecutar la misma operación múltiples veces = mismo resultado que ejecutarla una vez"

EJEMPLO VISUAL: Interruptor de luz

- Presionar "ON" cuando ya está prendida = sigue prendida
- Presionar "OFF" cuando ya está apagada = sigue apagada

EJEMPLO MALO (NO idempotente):

Sumar \$100 a cuenta bancaria
Si se ejecuta 2 veces por error = +\$200 (PROBLEMA!)

EJEMPLO BUENO (Idempotente):

Establecer saldo en \$1000
Se puede ejecutar 1000 veces = siempre \$1000 (SEGURO!)

APLICACIONES REALES:

- PayPal: Transacciones seguras
 - Kubernetes: Deployments repetibles
 - Terraform: Infraestructura como código
-

SLIDE 11: HITO 2 - FAULT TOLERANCE MASTERY

Imagen de Referencia: Sistema funcionando perfectamente mientras partes fallan

TÍTULO: "🎯 HITO 2: Dominas la Mentalidad Anti-Fallo"

EL DESAFÍO: "💣 **REALIDAD BRUTAL:** En sistemas distribuidos, algo SIEMPRE está fallando

- Discos se rompen cada día
- Cables se desconectan
- Software tiene bugs
- Servidores se reinician"

PORQUÉ ES REVOLUCIONARIO: "🛡️ **MINDSET BREAKTHROUGH:** No evitar fallos, ASUMIR que van a pasar Diseñar sistemas que funcionan AUNQUE cosas fallen"

CÓMO NACIÓ: "📖 **EVOLUCIÓN:** De 'hacer sistemas perfectos' a 'hacer sistemas anti-frágiles'"

TÉCNICAS QUE DOMINAS:

- Replicación automática
- Recovery automático
- Degradación elegante

💪 **MENSAJE DE ÁNIMO:** "¡INCREÍBLE! Ahora tienes la mentalidad de sistemas confiables. Ya no le temes a los fallos - los anticipas y los superas. ¡Eres más resistente que el 90% de desarrolladores! 💪"

SLIDE 12: APIS - CÓMO SE COMUNICAN LAS APLICACIONES

Imagen de Referencia: Dos aplicaciones comunicándose por API

TÍTULO: "APIs: El Idioma Común de las Aplicaciones"

ANALOGÍA: Como un mesero en restaurante

- Tú (cliente) pides comida
- Mesero (API) lleva orden a cocina
- Cocina (aplicación) prepara comida
- Mesero trae la comida de vuelta

EN NUESTRO SISTEMA:

- Jupyter habla con Spark via API
- Spark habla con HDFS via API
- Todo coordinado sin que tú veas la complejidad

TIPOS QUE USAREMOS:

- **REST APIs:** Para interfaces web
- **gRPC:** Para comunicación rápida entre servicios
- **SQL:** Para queries a bases de datos

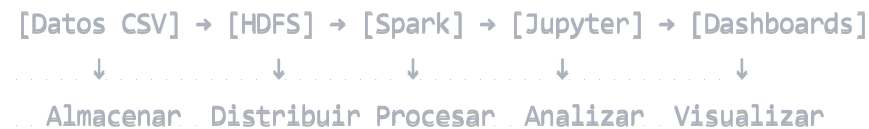
BENEFIT: "Cada herramienta se especializa, pero todas trabajan juntas"

SLIDE 13: ARQUITECTURA QUE CONSTRUIREMOS HOY

Imagen de Referencia: Diagrama completo del stack que implementarán

TÍTULO: "Nuestro Stack Big Data Real"

VISUAL FLOW:



COMPONENTS:

- **Docker:** Contenedores para todo
- **HDFS:** Sistema de archivos distribuido
- **Spark:** Motor de procesamiento
- **Jupyter:** Ambiente de desarrollo
- **Dataset Real:** Transporte público de Lima

INTERFACES QUE USAREMOS:

- <http://localhost:8888> - Jupyter Lab
 - <http://localhost:9870> - HDFS Web UI
 - <http://localhost:8080> - Spark Master UI
-




SLIDE 14: PRIMER WIN - LO QUE LOGRAREMOS HOY

Imagen de Referencia: Dashboard con datos reales funcionando

TÍTULO: "Tu Primer Éxito: Cluster Big Data Funcionando en 30 Minutos"

WINS CONCRETOS:

1. ☒ Cluster distribuido real en tu laptop
2. ☒ Datos reales de transporte cargados en HDFS

3.  Primer análisis con Spark funcionando
4.  Notebook de validación 100% green
5.  Interfaces web monitoreando en tiempo real

PROOF POINTS:

- Comando HDFS exitoso: `hdfs dfs -ls /`
- Job Spark completado: Análisis de rutas de transporte
- Dashboard web funcionando en browser

CONFIDENCE BUILDER: "Si logras esto hoy, puedes lograr cualquier cosa en Big Data"

CAREER IMPACT: "Este setup es más sofisticado que el 80% de empresas"

SLIDE 15: HITO 3 - PRIMER CLUSTER FUNCIONANDO

Imagen de Referencia: Celebración con cluster real funcionando en pantalla





TÍTULO: "🎯 HITO 3: ¡Tu Primer Cluster Big Data REAL!"

EL DESAFÍO QUE VAS A SUPERAR: "🏔️ **DESAFÍO TÉCNICO:** Configurar un cluster distribuido que funcione (Esto intimida al 80% de desarrolladores)"

PORQUÉ ES UN LOGRO ÉPICO: "⚡ **BREAKTHROUGH TÉCNICO:** En 30 minutos tendrás:

- Sistema más sofisticado que en muchas empresas
- Herramientas que usan en Google, Netflix, Amazon
- Confianza técnica para cualquier desafío"

CÓMO EVOLUCIONAS: "🚀 **TRANSFORMACIÓN:** De 'no sé configurar esto' a 'puedo armar cualquier sistema'"

LO QUE LOGRARÁS HOY:  Cluster distribuido real funcionando  Datos reales procesándose
 Dashboard profesional activo  Confianza técnica desbloqueada

💪 **MENSAJE DE ÁNIMO FINAL:** "¡ESTÁS A 30 MINUTOS DE SER INGENIERO BIG DATA! Si logras esto hoy, puedes lograr CUALQUIER COSA en tecnología. No hay límites para lo que puedes construir. ¡Vamos a hacer historia tecnológica JUNTOS! 🌟🚀"

CALL TO ACTION: "¡MANOS A LA OBRA! → <http://localhost:8888>"

NOTAS PARA EL INSTRUCTOR

Timing Sugerido:

- Slides 1-3: 3 minutos (setup mental)
- Slides 4-6: 3 minutos (conceptos técnicos básicos)
- Slides 7-11: 6 minutos (core big data concepts)
- Slides 12-13: 2 minutos (arquitectura específica)
- Slides 14-15: 1 minuto (transition)

Puntos de Interacción:

- Slide 2: "¿Quién ha usado Google hoy?" (manos arriba)
- Slide 4: "¿Preferirían un camión grande o 10 camiones pequeños para mudanza?"
- Slide 8: "¿Qué es más importante: banco siempre disponible o saldo exacto?"

Transición a Práctica:

- Después del slide 15: INMEDIATAMENTE a terminals
- Instructor hace setup en pantalla mientras estudiantes siguen
- Primeros 10 minutos: Setup validation together
- Siguientes 20 minutos: Primer notebook guided

Conceptos Críticos para Retener:

1. **Distributed thinking** sobre monolithic thinking
2. **Data locality** principle
3. **Fault tolerance** mindset
4. **Trade-offs** awareness (CAP theorem)
5. **Idempotency** para reliability

Meta: Estudiantes entienden EL PORQUÉ antes del CÓMO