

When Streams Fail Kafka Off the Shore What If?

© 2017 Goldman Sachs. This presentation should not be relied upon or considered investment advice. Goldman Sachs does not warrant or guarantee to anyone the accuracy, completeness or efficacy of this presentation, and recipients should not rely on it except at their own risk. This presentation may not be forwarded or disclosed except with this disclaimer intact.

These materials ("Materials") are confidential and for discussion purposes only. The Materials are based on information that we consider reliable, but Goldman Sachs does not represent that it is accurate, complete and/or up to date, and it should not be relied on as such. The Materials do not constitute advice nor is Goldman Sachs recommending any action based upon them. Opinions expressed may not be those of Goldman Sachs unless otherwise expressly noted. As a condition to Goldman Sachs presenting the Materials to you, you agree to treat the Materials in a confidential manner and not disclose the contents thereof without the permission of Goldman Sachs.

© Copyright 2017 The Goldman Sachs Group, Inc. All rights reserved.

ABOUT US

- Investment Management Division
- \$1.4 Trillion Assets Under Management
- Core Front-Office Platform Team
- Interface with 15+ teams
- Kafka, Event Topology, Data Fabric, Akka, etc...
- Sounds fun? Talk to me!
- What if...

What If? This presentation is boring

- Topology Strategy, Context & Deployment Goals
- What If? Hosts / Network / DC Failures
- Deployment & Monitoring Strategy
- What If? Framework & Software Failure



KAFKA DEPLOYMENT USAGE PATTERNS

Most used clusters serve ~1.5Tb a week to

consumers

- However message count relatively low order of millions per week; avg. several hundred a second
- At peak periods
 - ~1,500 messages produced/second
 - ~2.5Mb produced/second
 - ~12.5Mb consumed/second

DEPLOYMENT GOALS

- No data-loss even in case of DC outage
- No Primary/Back-Up notion
- No "failover" scenarios
- Minimize Outage time



DATALOSS 101

- Tape Backup
- Nightly Batch Replication
- Async Replication
- Sync Replication (ex: SRDF)



Speed of Light Network Roundtrips

NYC to SF ~60ms

Virginia to Ohio ~12ms

NYC to NJ ~4ms



What If? Datacenters were near...

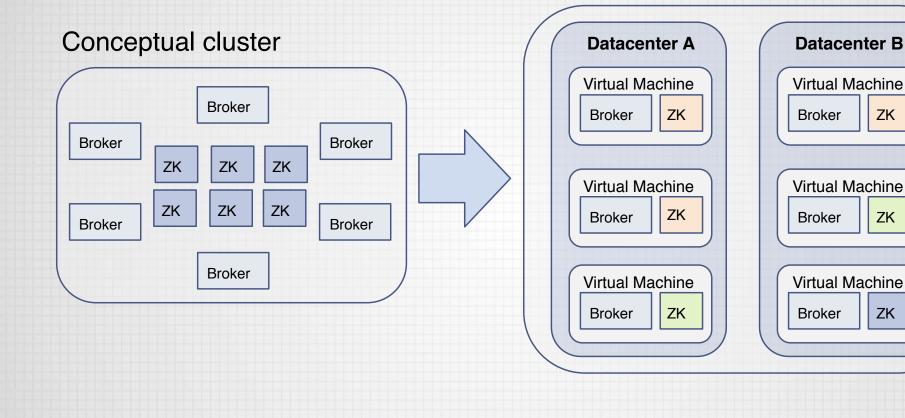
Treat multi-DC environment as

a single redundant data-center

where half could go down

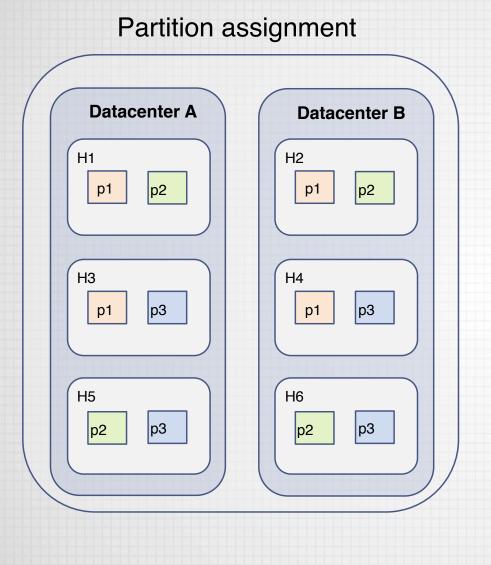


DEPLOYMENT STRATEGY



Physical cluster

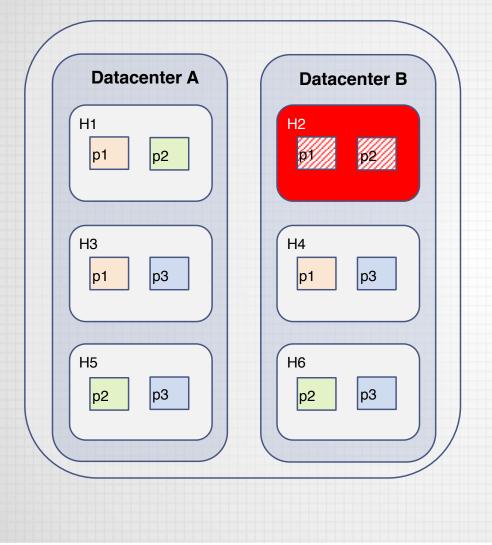
EXAMPLE Single Topic Setup



- 1 topic
- 3 partitions (p1-p3)
- Replication factor of 4
- Ensure even replicas between DCs
- Min.Insync.Replicas 3
- Cross-DC latency is low!

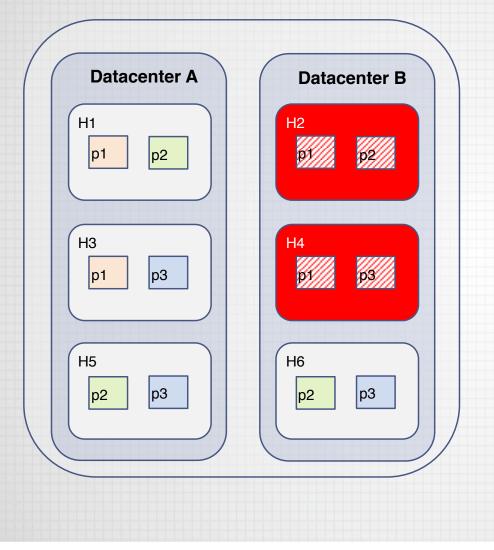


What If? A host fails



- 1-5 times / year
- No impact to producers/consumers (still able to satisfy 3 ISR)
- No manual recovery beyond replacing host

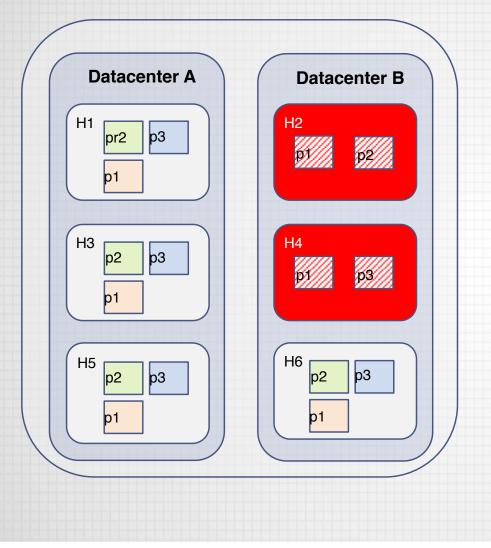
What If? Two hosts fail



- 1/year depending where hosts are (e.g. bad hypervisor)
- Processing for some topics will be halted
- Short-term: Add replicas for affected partitions on remaining hosts
- ASAP: Replace bad hosts
- GS Dynamic Compute allows seamless VM replace with no need to re-point dns aliases/change kafka config



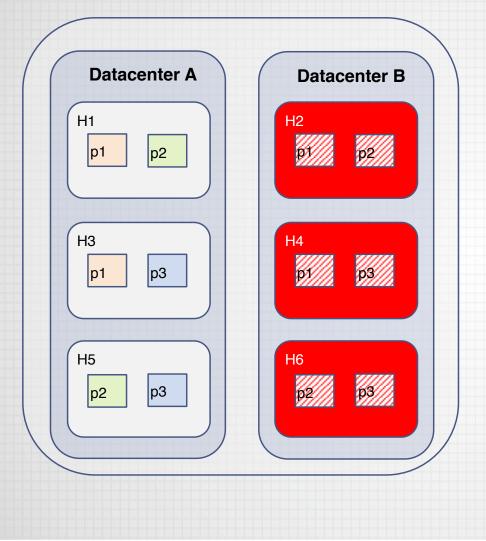
What If? Two hosts fail



- 1/year depending where hosts are (e.g. bad hypervisor)
- Processing for some topics will be halted
- Short-term: Add replicas for affected partitions on remaining hosts
- ASAP: Replace bad hosts
- GS Dynamic Compute allows seamless VM replace with no need to re-point dns aliases/change kafka config



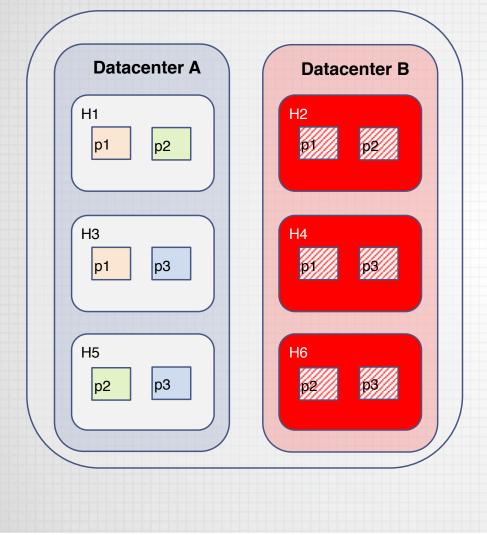
What If? Three hosts fail



• 1 / few years

- Cluster processing halted as cannot satisfy in-sync replica requirements
- Proceed with immediate host replacement

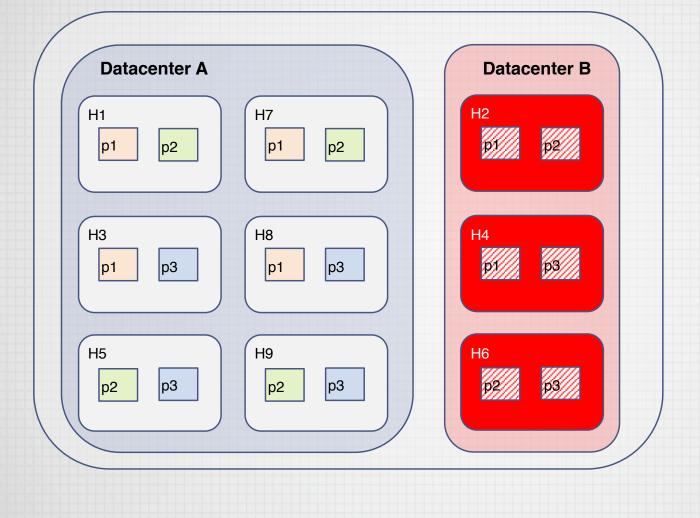
What If? Datacenter Failed / Network Partition



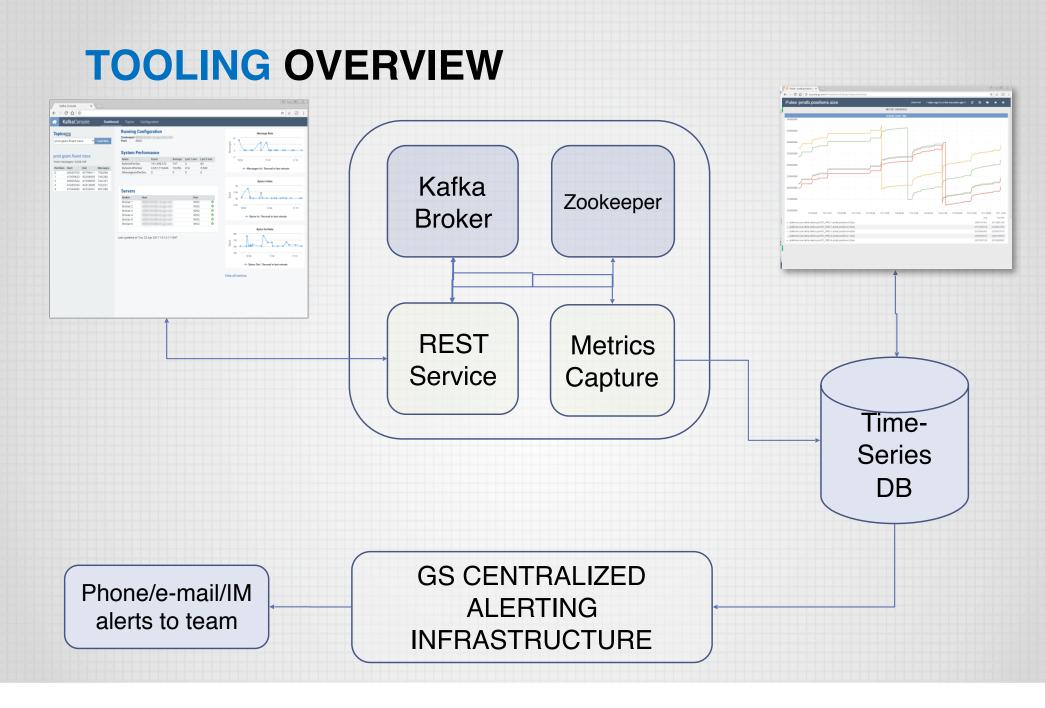
- Once a 20-year event
- Short-term strategy: add additional machines in Datacenter A
- Largest impact on recovery time is how long to get new hosts provisioned

What If? Datacenter Failed / Network Partition

After adding additional hosts



- Short-term strategy: add additional machines in Datacenter A
- Largest impact on recovery time is how long to get new hosts provisioned
 - Stand-by Hosts?



KAFKA@GS

Images are shown for illustrative purposes only and should not be relied upon as representative of actual or future information for Goldman Sachs products and services.

TOOLING CLUSTER DASHBOARD

| Kafka Console × | |
|---|--|
| ÷ → C ∆ 0 | * 🔺 🖬 |
| KafkaConsole Dashboard | Topics Configuration |
| Topics 72 ref.runs.watermark Coad data | Starting offset Ending offset Encoding 19533551 19533570 String Reload messages Partition 0 has 3 replicas across the following nodes 1 5 2 |
| Partition Start End Messages O 7809094 19533570 11,724,476 Topic specific config Property Value min.insync.replicas 2 | Offset: 19533570 Link key: watermark payload: { "start5eqNo": "268413148", "endSeqNo": "268413158", "queue5ize": "1", "maxUniversalId": "0x00060506ed6dea7200024e59414d46535031666973685f726566000051af00000000000000000000000000000000 |
| | Offset: 19533569 Offset: 19533568 Offset: 19533567 |
| | Offset: 19533566 |
| | Offset: 19533564 |
| | Offset: 19533563 Offset: 19533562 |
| | Offset: 19533561 |
| | Offset: 19533550 Offset: 19533559 |
| | Offset: 19533558 * |

Endpoints include:

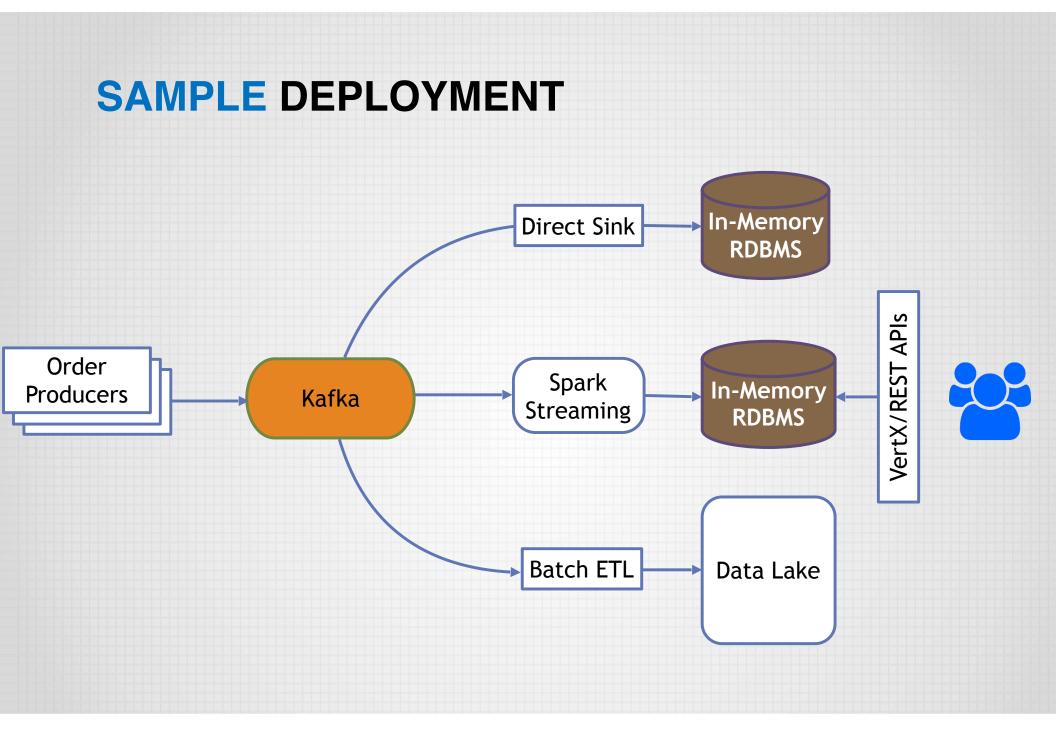
- View messages on topic
- Consumer lag
- Leader & ISRs for topic
- Highwatermark for topic
- Run-Time Broker & zookeeper configuration



TOOLING HEALTHCHECK

- Topic sizes monitoring
- Alert when at risk of losing data due to truncation
- If partitions breach threshold, escalate via GS alerting infrastructure





What If? I need to Replay

- Message Remains on Queue?
- Delivery Semantics
 - == 1
 - =< 1
 - >=1
- Unique ID on everything
- Making Replay Easy
- KIP-98



What If? Spark Streaming Fails

Jobs Stages Storage Environment Executors Streaming

ErrorStreaming2 application UI

Spark Jobs ^(?)

Total Uptime: 47.1 h Scheduling Mode: FIFO Active Jobs: 7

Completed Jobs: 11273, only showing 193

Event Timeline

Active Jobs (7)

| Job Id | Description | Submitted | Duration | Stages: Succeeded/Total | Tasks (for all stages): Succeeded/Total |
|--------|--|---------------------|----------|-------------------------|---|
| 11305 | transform at ErrorStreaming2.scala:396 | 2016/03/14 13:20:07 | 12 s | 0/6 | 0/805 |
| 9816 | transform at ErrorStreaming2.scala:396 | 2016/03/14 07:14:56 | 6.1 h | 1/6 | 126/874 |
| 9812 | transform at ErrorStreaming2.scala:422 | 2016/03/14 07:14:15 | 6.1 h | 2/10 | 79/947 |
| 9804 | transform at ErrorStreaming2.scala:422 | 2016/03/14 07:10:43 | 6.2 h | 5/10 | 228/947 |
| 9796 | transform at ErrorStreaming2.scala:422 | 2016/03/14 07:06:54 | 6.2 h | 5/10 | 263/947 |
| 2064 | transform at ErrorStreaming2.scala:396 | 2016/03/12 22:50:55 | 38.5 h | 0/6 | 0/874 |
| 2060 | transform at ErrorStreaming2.scala:422 | 2016/03/12 22:50:45 | 38.5 h | 4/10 | 158/947 |

Completed Jobs (11273, only showing 193)

| Job Id | Description | Submitted | Duration | Stages: Succeeded/Total | Tasks (for all stages): Succeeded/Total |
|--------|--|---------------------|----------|-------------------------|---|
| 11304 | transform at ErrorStreaming2.scala:396 | 2016/03/14 13:20:00 | 7 s | 6/6 | 874/874 |
| 11303 | transform at ErrorStreaming2.scala:422 | 2016/03/14 13:18:29 | 3 s | 1/1 (9 skipped) | 70/70 (946 skipped) |
| 11302 | transform at ErrorStreaming2.scala:422 | 2016/03/14 13:18:26 | 3 s | 6/6 (4 skipped) | 298/298 (649 skipped) |
| 11301 | transform at ErrorStreaming2.scala:422 | 2016/03/14 13:18:17 | 9 s | 1/1 (9 skipped) | 70/70 (946 skipped) |
| 11300 | transform at ErrorStreaming2.scala:422 | 2016/03/14 13:18:13 | 4 s | 7/7 (3 skipped) | 299/299 (648 skipped) |
| 11299 | transform at ErrorStreaming2.scala:396 | 2016/03/14 13:18:07 | 6 s | 1/1 (5 skipped) | 70/70 (804 skipped) |
| 11298 | transform at ErrorStreaming2.scala:396 | 2016/03/14 13:18:07 | 50 ms | 1/1 (5 skipped) | 1/1 (804 skipped) |
| 11297 | transform at ErrorStreaming2.scala:396 | 2016/03/14 13:18:07 | 81 ms | 1/1 (5 skipped) | 1/1 (804 skipped) |
| 11296 | transform at ErrorStreaming2.scala:396 | 2016/03/14 13:18:00 | 7 s | 6/6 | 874/874 |
| 11295 | transform at ErrorStreaming2.scala:422 | 2016/03/14 13:16:27 | 4 s | 1/1 (9 skipped) | 70/70 (946 skipped) |
| 11294 | transform at ErrorStreaming2.scala:422 | 2016/03/14 13:16:25 | 2 s | 6/6 (4 skipped) | 298/298 (649 skipped) |
| 11293 | transform at ErrorStreaming2.scala:422 | 2016/03/14 13:16:18 | 7 s | 1/1 (9 skipped) | 70/70 (946 skipped) |
| 11292 | transform at ErrorStreaming2.scala:422 | 2016/03/14 13:16:16 | 2 s | 7/7 (3 skipped) | 299/299 (648 skipped) |
| 11291 | transform at ErrorStreaming2.scala:396 | 2016/03/14 13:16:05 | 11 s | 1/1 (5 skipped) | 70/70 (804 skipped) |
| 11290 | transform at ErrorStreaming2.scala:396 | 2016/03/14 13:16:05 | 32 ms | 1/1 (5 skipped) | 1/1 (804 skipped) |
| 11289 | transform at ErrorStreaming2.scala:396 | 2016/03/14 13:16:05 | 30 ms | 1/1 (5 skipped) | 1/1 (804 skipped) |
| 11288 | transform at ErrorStreaming2.scala:396 | 2016/03/14 13:16:00 | 5 s | 6/6 | 874/874 |

SUMMARY

- Failure <u>will</u> occur
- Belt & Suspenders for everything
- Ultimate Throughput vs Ultimate Reliability
- Kafka has many Knobs, perhaps too many, hide some
- Resilience through Transparency





text time...

