Delta Lake & Open Data Sharing

Modernizing BC Wildfire Service Weather Data Infrastructure

How This Started: A Climatology Request

The Ask

Eric Kopetski requested **climatology visualizations** — compare current fire weather conditions against historical norms.

What We Discovered

- Jake Lee presented an **R Shiny dashboard** with similar visualizations
- Sam Siddall had been doing **overlapping analyses** independently
- Everyone was spending time **sourcing data** instead of building tools

Pattern emerged: smart people, duplicated effort, data bottleneck.

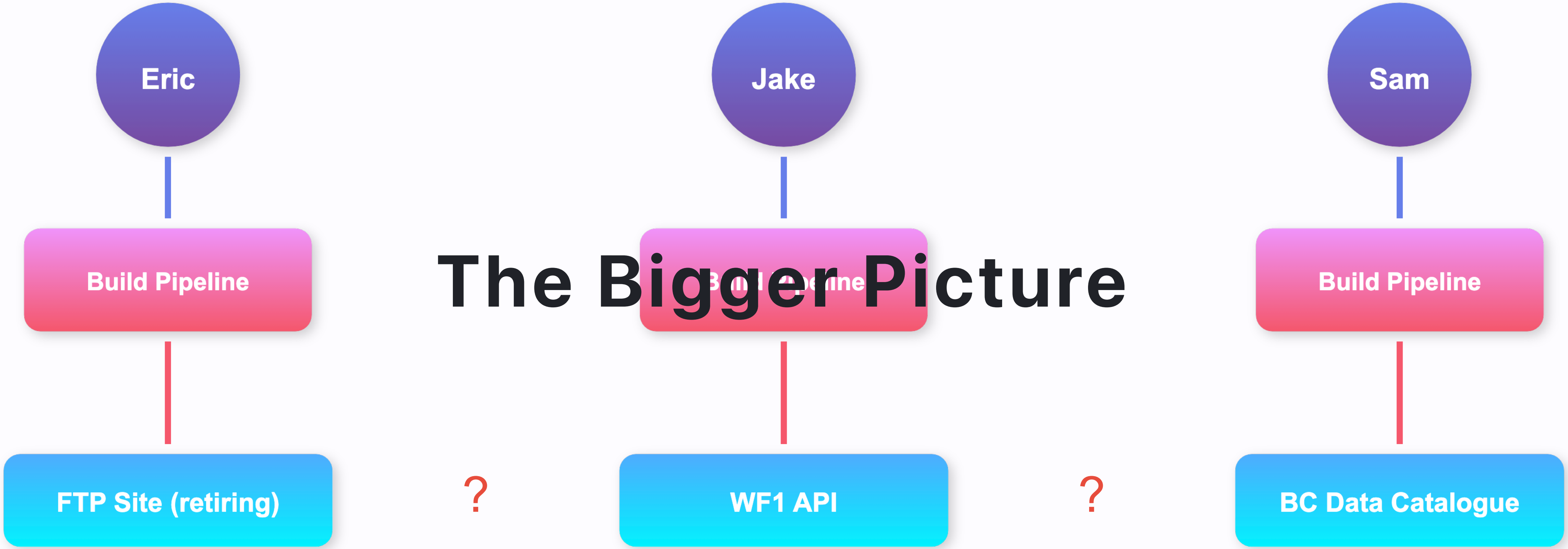
The Data Landscape Today

Source	What It Has	Problem
BC Data Catalogue	Metadata, some CSVs	Stale, hard to find
WF1 API	Operational data	Not designed for historical analysis
FTP Site	Historical CSVs	**Retiring soon**
Jaspersoft Server	Reports, exports	Another copy, limited access

The Real Issue

- **4+ copies** of weather data across systems
- Everyone builds their own pipeline
- No single source of truth

Current State: Duplicated Effort



Eric

Jake

Sam

Build Pipeline

Build Pipeline

Build Pipeline

FTP Site (retiring)

?

WF1 API

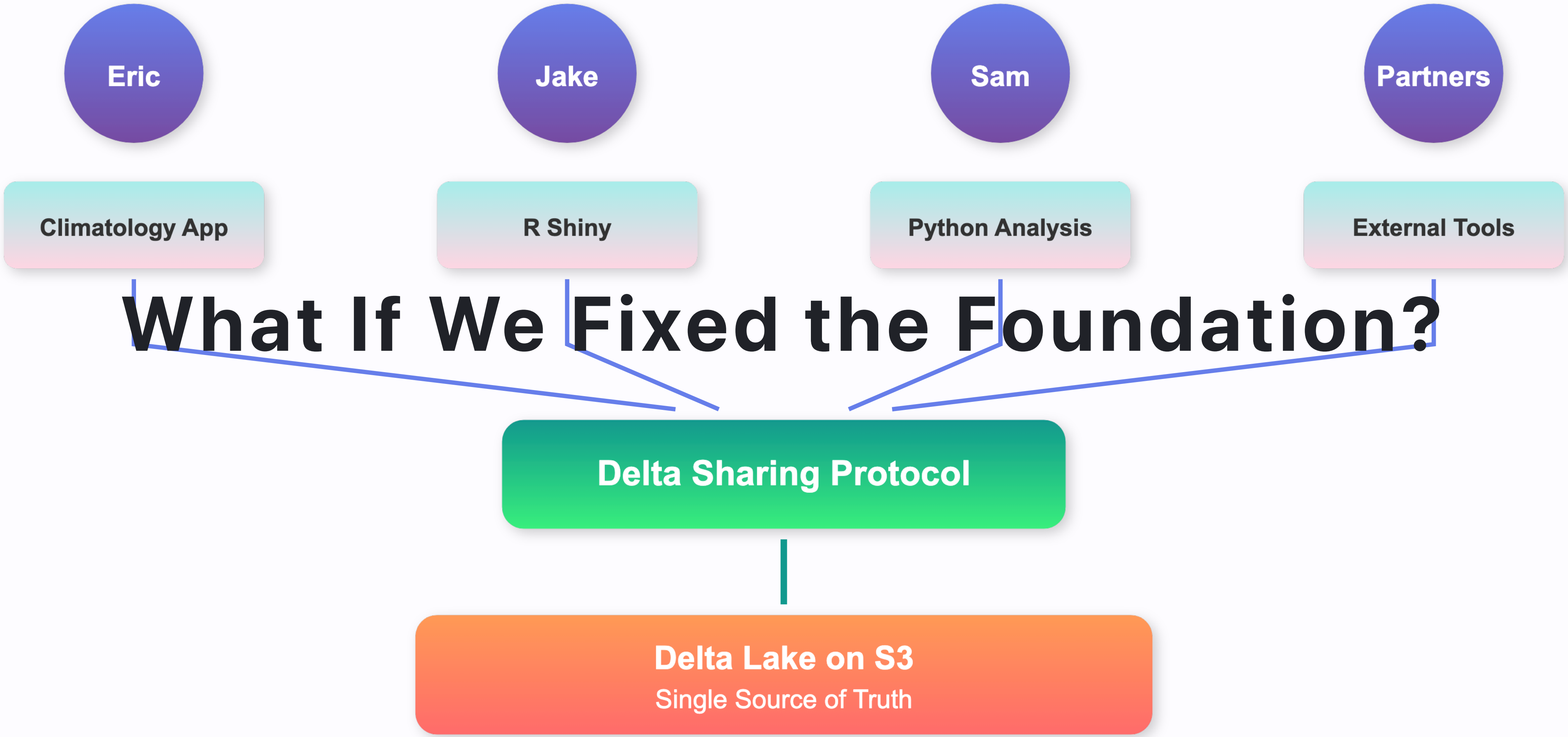
?

BC Data Catalogue

The Bigger Picture

Same problem solved 3 different ways

Future State: Shared Foundation



What If We Fixed the Foundation?

✓ Focus on insights, not plumbing

BC Government Mandate: Manage Data Effectively

> "Ethical, accurate, accessible data forms the foundation of digital government"
> — **BC Digital Code of Practice**

DCOP Requirements We're Addressing

- **Reduce duplication** — Single source of truth, not scattered CSV copies
- **Interoperability** — Standard metadata and interfaces for easy exchange
- **Findability & Reusability** — Catalogued tables, documented schemas
- **Data lifecycle management** — From collection to final disposition
- **Quality assurance** — Validated, consistent data formats

Source: digital.gov.bc.ca/policies-standards/dcop/data/

The Old Way: Expensive & Complex

- **Dedicated databases** — Licensing, maintenance, DBAs
- **ETL pipelines** — Custom code for every consumer
- **Data warehouses** — Costly infrastructure to scale
- **Multiple copies** — Storage costs multiply

The Hidden Costs Add Up

Component	What You Pay For
Storage	Database servers, SAN storage
Compute	Always-on clusters, reserved capacity
Scaling	Buy bigger hardware, migrations
Licensing	Per-core, per-user fees

The New Way: Simple & Scalable

Delta Lake on Object Storage

Component	Delta Lake Approach
Storage	**S3** — pennies per GB
Compute	**Serverless** — pay per query
Scaling	**Automatic** — cloud-native
Licensing	**Open source** — free

Cost Impact

Type	Cost/TB/month
Enterprise database	$500 - $2,000
S3 object storage	**$23** (96% savings)

What About the BC Data Catalogue?

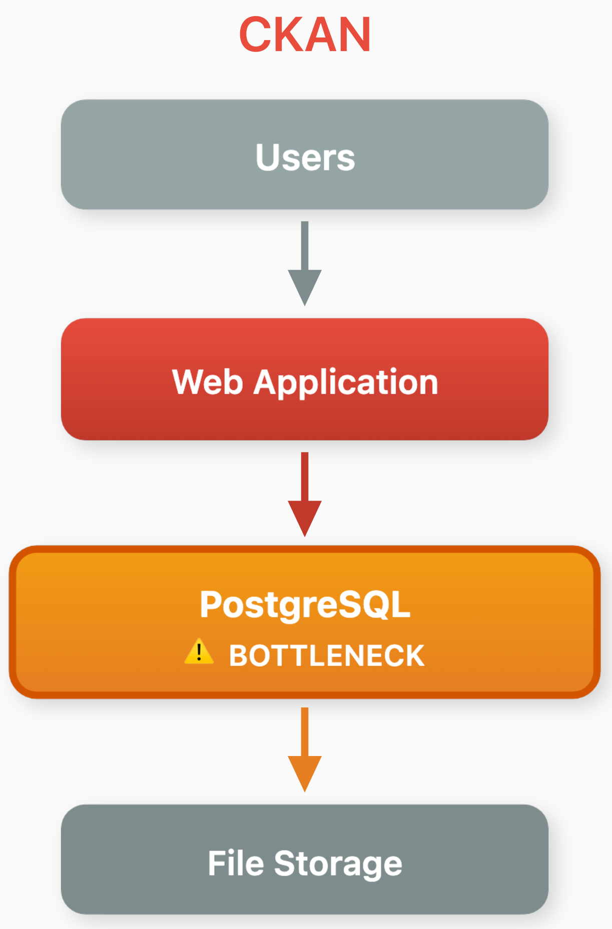
Aspect	BC Data Catalogue (CKAN)	Delta Sharing
Purpose	Compliance (publish metadata)	Usability (access data)
Search	Basic, hard to find things	Programmatic API
Data access	Download CSVs manually	Query directly
Freshness	Stale uploads	Always live
Extensibility	Limited	Add any discovery layer

Delta Sharing does what matters: gets data to users fast.

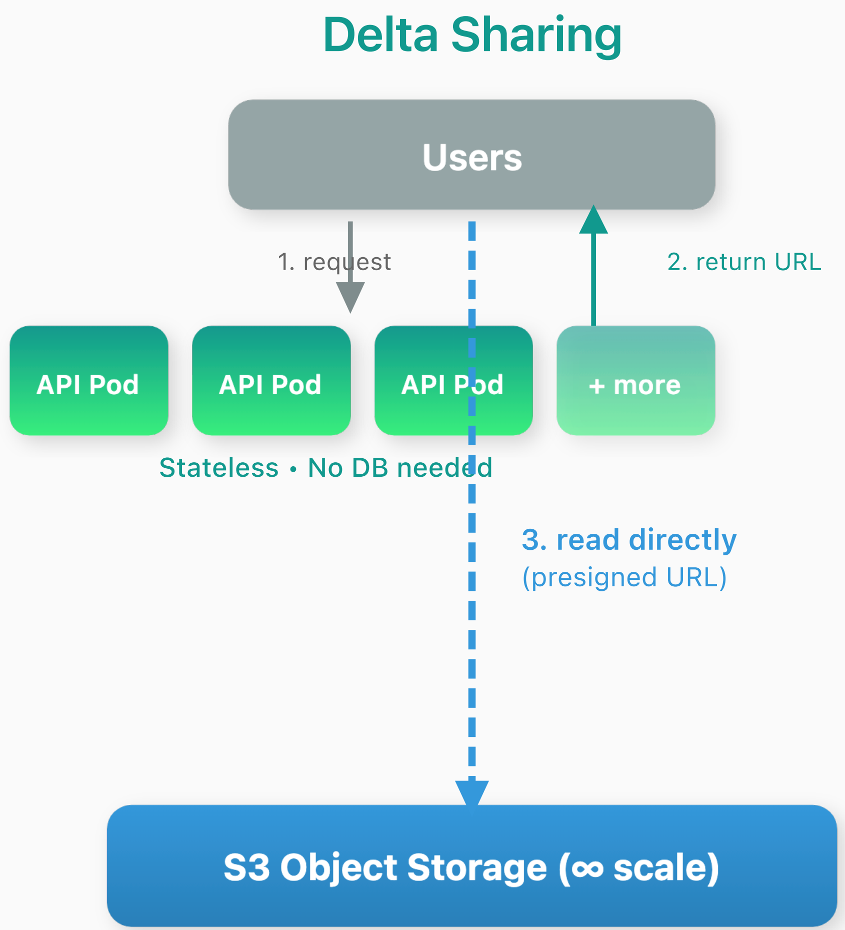
Scalability: CKAN vs Delta Sharing

Aspect	CKAN	Delta Sharing
Database	PostgreSQL (single point of failure)	**None** — stateless API
Storage	File server + DB storage	**S3** — infinitely scalable
API scaling	Limited by DB connections	**Horizontal** — add pods
Data transfer	Through web server	**Direct from S3** (presigned URLs)
10x more users	Upgrade DB, add caching	**Add API pods**
10x more data	Upgrade storage, reindex	**Just more S3 objects**

Why Stateless Scales Better



All data flows through DB
Limited connections • Single point of failure



API just hands out URLs
Data reads bypass the API entirely

Build Better Discovery on Delta Sharing

Capability	Built-in	Add On Top
List shares/schemas/tables	**Yes**	—
Schema & partition metadata	**Yes**	—
Secure live data access	**Yes**	—
Full-text search	—	Elasticsearch, Algolia, etc.
Human descriptions	—	Metadata API or docs
Governance & audit	—	Token scopes, logging

Or Integrate with Existing Catalogues

Link BC Data Catalogue entries → Delta Sharing endpoints

Start with great data access. Add discovery as needed.

The Challenge: 40 Years of Critical Data

Current State:

- ~40 years of hourly weather observations (1987-present)
- Scattered across CSV files on legacy FTP servers
- No standardized access method
- Every consumer downloads, parses, processes independently

Impact:

- Slow, expensive queries for fire weather analysis
- Duplicated effort across teams and partners
- No efficient access for R/Python analysts
- Barriers to research collaboration

The Solution: Delta Lake + Open Sharing

Data Flow

BCWS Data Mart (FTP/CSV)

↓ *automated crawler*

Delta Lake (S3/Parquet)

↓ *Delta Sharing protocol*

Any Tool (Python/R/Spark)

Who Benefits

- **Internal teams** — fast queries
- **External partners** — self-service
- **Researchers** — standard protocols
- **Future tools** — solid foundation

Why Delta Lake?

Open Source Lakehouse Format

- **Parquet-based** — Industry standard, 10-100x compression
- **ACID Transactions** — Safe concurrent writes, no corruption
- **Time Travel** — Query data at any historical point
- **Schema Evolution** — Add columns without breaking queries

Performance Comparison

Operation	CSV (Current)	Delta Lake
Load 1 year	Minutes	**Seconds**
Query single station	Scan all files	**Partition pruning**
Daily updates	Rewrite everything	**Append only**

Why Delta Lake? (cont.)

Open Standards — No Vendor Lock-in

> Backed by **Linux Foundation**
> Supported by industry leaders

Cloud Providers

- AWS
- Azure
- Google Cloud

Data Platforms

- Databricks
- Snowflake
- Apache Spark

Native Libraries: Python, R, Rust, Java, Scala

What We Built: Automated Pipeline

Data Crawler

```
python -m wps_deltalake.crawler --all
```

Capabilities:

- Crawls BCWS Data Mart automatically
- Incremental updates (only new data)
- Partitioned by year/month for fast queries
- Enriches station metadata

What We Built: Four Optimized Tables

Table	Purpose	Partitioning
`observations`	All hourly weather data	year, month
`stations`	Station metadata	—
`observations_by_station`	Per-station analysis	station_code
`climatology_stats`	Pre-computed percentiles	—

Maintenance Built-in

```
python -m wps_deltalake.maintenance --all
```

Checkpoint • Optimize • Vacuum

Delta Sharing: Open Data Protocol

Aspect	Traditional Sharing	Delta Sharing
Process	Extract → Copy → Share → Repeat	Query live data
Data copies	Many (one per consumer)	**None** (presigned URLs)
Freshness	Stale (point-in-time exports)	**Always live**
Governance	Who has what version?	**Single source of truth**
Languages	Whatever you export to	**Python, R, Spark, etc.**
Security	File permissions	**Token-based auth**

Delta Sharing: Industry Standard

> "Delta Sharing is the **first open protocol** for secure data sharing, making it simple to share live data from any platform."
> — **Linux Foundation**

How It Works

```
Client Request → API validates token → Returns presigned S3 URLs
                                              ↓
                            Client reads Parquet directly from S3
```

No intermediate copies. No stale data.

How Partners Access Data: Python

```
import delta_sharing

client = delta_sharing.SharingClient("wps.share")

# List available tables
tables = client.list_all_tables()
# → historical.default.observations
# → historical.default.stations

# Load as pandas DataFrame
df = delta_sharing.load_as_pandas(
    "wps.share#historical.default.observations"
)
```

How Partners Access Data: R

```
library(httr)
library(arrow)

# Query the Delta Sharing endpoint
response <- POST(
  "https://api.example.com/delta-sharing/.../query",
  add_headers(Authorization = "Bearer <token>")
)

# Read parquet files directly from presigned URLs
df <- read_parquet(presigned_url)
```

Live Application: Climatology Dashboard

What It Does

Compare current fire weather against **30-year historical norms**

Powered by Delta Lake

- **Pre-computed statistics**: p10, p25, p50, p75, p90 by station/day
- **Sub-second queries** for any station
- **Multi-year comparison** overlays

Variables Available

Temperature • Humidity • Wind • Precipitation • FFMC • ISI • FWI

Demo: `/climatology`

Business Value

For BCWS Operations

- Query 40 years in **seconds**
- Automated daily ingestion
- Smaller storage footprint

For Research

- Universities query directly
- **Reproducible** analyses
- Scales laptop → cluster

For External Partners

- **Self-service** data access
- Works with existing tools
- Always current data

Cost Savings

- No duplicate pipelines
- Reduced manual effort
- Standard protocols

Security & Governance

Access Control

- Bearer token authentication
- Per-share access grants
- Audit logging

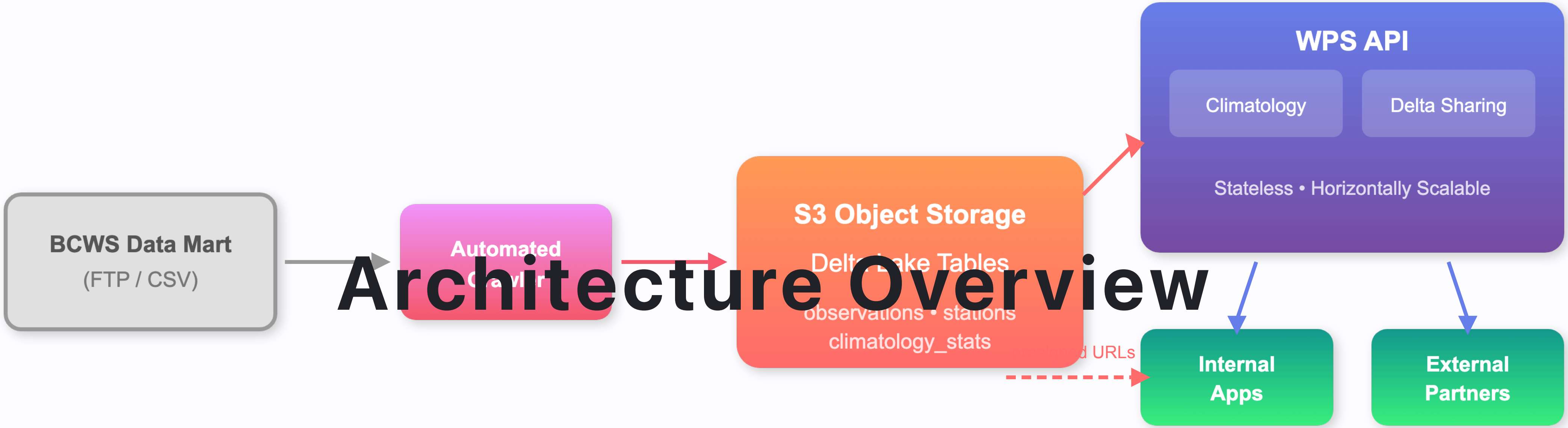
Data Governance

- Single source of truth
- Version history
- Schema enforcement

Compliance

- Data stays in **your** S3 bucket
- Pre-signed URLs expire (1 hour)
- No external systems store data

Architecture Overview



Architecture Overview

Data flows through API → Clients read directly from S3

No database bottleneck

Next Steps

Immediate

1. Deploy Delta Sharing to production
2. Document partner onboarding
3. Establish refresh schedule

Near-term

- Expand climatology stats
- Partner authentication workflow
- Sample analysis notebooks

Future

- Extend to other BCWS datasets
- Federation with external servers
- Real-time streaming updates

DCOP Alignment Summary

DCOP Mandate	How We Deliver
Reduce duplication	Single Delta Lake source, no CSV copies
Interoperability	Open standards (Parquet, Delta Sharing)
Easy exchange & reuse	Python/R/Spark clients, presigned URLs
Metadata standards	Schema versioning, partition metadata
Lifecycle management	Automated ETL, maintenance, vacuum
Quality assurance	ACID transactions, schema enforcement

This project directly advances BC's digital transformation goals.

Key Takeaways

1. Policy Aligned

- Delivers on **DCOP "Manage Data Effectively"** mandate
- Reduces duplication, improves interoperability

2. Fiscally Responsible

- **90%+ cost savings** vs traditional databases
- Open source — no licensing fees
- Scales without infrastructure investment

3. Open Standards & Immediate Value

- **Linux Foundation** backed, no vendor lock-in
- **10-100x faster** queries today
- Foundation for **data mesh** architecture

Questions?

Resources

- Delta Lake: delta.io
- Delta Sharing: github.com/delta-io/delta-sharing

Demo

- Climatology: `/climatology`
- Delta Sharing: `/api/delta–sharing`