DataTran POC Presentation — Speaker Notes (Storytelling & Influence)

> **Audience:** Engineering, Architecture, DevOps, Product Owners, Security
>
> **Slides:** 12–16 (suggested) **Duration:** 25–30 minutes (15–20 mins presentation + 7–10 min demo + Q&A)

Slide 1 — Title

Title: Onboarding *DataTran* — POC Story & Results **Subtitle:** From Snowflake → Oracle to a reusable, pip-installable cloud-native transfer framework **Presenter:** <Your name> — <Team> **Date:** <presentation date>

Speaker notes (30s): Open with a confident one-liner: "Today I'll show how we turned a one-off Snowflake→Oracle tool into DataTran — a secure, pip-packaged, cloud-native transfer framework that we deployed as a repeatable solution for heterogeneous database transfers." Pause and smile, set expectations for demo & decisions at the end.

Slide 2 — Opening story: the friction we lived with

Visual: Two-column slide: Left — "Before" (ad-hoc scripts, inconsistent logs, manual runs). Right — "After POC" (DataTran running via Jenkins → Batch, tracked, secure).

Narrative bullets: - We had many custom scripts for each migration — slow onboarding, fragile failures. - Teams spent cycles debugging credentials, mounts, and ad-hoc retry logic. - We piloted DataTran to change that and prove a repeatable pattern.

Speaker notes (45s): Tell a 1-minute anecdote: a failed migration that blocked analytics for a day due to missing retries or secrets — then say, "we built the POC to prevent that pain from repeating." This frames the technical parts as problem-solving.

Slide 3 — What is DataTran? (short & punchy)

Bullets (copy-ready): - Python-based transfer framework created by XU team; now packaged as a pip artifact in JFrog Artifactory. - Connector-driven: supports Snowflake, Oracle, MSSQL, DB2 (and more via connectors). - Cloud-native: packaged as containers, runs on AWS Batch/EC2 with S3 staging and centralized secrets. - Designed for repeatable bulk/incremental transfers with standardized logging and retries.

Speaker notes: Underline the pip package line — it means easy distribution and versioning across teams.

Slide 4 — Why DataTran? (the blunt value props)

Bullets: - Speed: deploy new pipelines quickly using the same framework and connectors. - Reliability: standardized retries, structured logs, and job correlation IDs. - Security: secrets in Secrets Manager, KMS encryption, Parameter Store for config. - Maintainability: single framework to upgrade vs dozens of scripts. - Cost control: driven by AWS Batch/EC2 vCPU-hours and S3 staging (we can tune concurrency and spot usage).

Speaker notes: Keep it concise: repeat one strong claim: "less toil, faster migrations, less risk." Say it twice.

Slide 5 — Primary Use Cases (story-driven)

Bullets: - One-time large-volume migrations (historical data onboarding). - Scheduled incremental loads for reporting or consolidation. - Cross-DB migrations during application refactoring. - Disaster recovery seeding and ad-hoc analytics extracts.

Speaker notes: Give an example: "We used DataTran to migrate a reporting schema from Snowflake to Oracle for <team>, reducing manual validation time from X hours to Y hours." (Insert numbers if available.)

Slide 6 — Architecture: the hero's journey (visual suggested)

Bullets (copy-ready): - Jenkins Core (Terracore pipelines) drives CI/CD and orchestration. - Two EC2 images are maintained and the images pushed to **ECR**: - Image A: spins ephemeral EC2 for DataTran worker runs. - Image B: creates two small EC2 instances (always-on) hosting **Control-M agents**. - Control-M jobs invoke ksh scripts on the Control-M EC2 instances, which execute `.prm` files. - `.prm` files invoke **AWS Batch** job with parameters. - AWS Batch launches EC2 instances (worker nodes) as needed; DataTran runs inside containers. - Scripts/artifacts stored in **S3** and temporarily mounted on ECS/EC2 workers. - Logs shipped to **CloudWatch**; notifications via **SNS**. - Secrets in **Secrets Manager**; application parameters in **Parameter Store**; KMS for encryption. - Job metadata (jobs & steps & arguments) stored in an Oracle DB in the cloud — DataTran reads this table at runtime.

Visual suggestion: Create a flow diagram: Jenkins → ECR (images) → Control-M agents (EC2 always-on) → ksh/.prm → AWS Batch → EC2 worker spin-up → DataTran container → Source DB / Target DB; S3 and SecretsManager/KMS/CloudWatch/SNS as supporting services.

Speaker notes: Walk the audience through a single transfer: Jenkins triggers pipeline → Control-M picks job → executes .prm → Batch creates worker → DataTran pulls secrets & params → reads from source → writes to S3 staging → loads into Oracle

---

Slide 7 — Deep-dive: Control-M, .prm, Batch, and how they dance

Bullets: - Control-M jobs are our enterprise scheduler; two agents run always-on on small EC2 instances. - Jobs call ksh scripts which in turn read `.prm` files — these `.prm` contain job-specific args (source/target/table, chunk size, concurrency). - `.prm` triggers Jenkins/Terracore pipeline to invoke an AWS Batch job. - Batch launches a worker EC2 instance with DataTran container; S3 scripts are mounted temporarily. - After completion, logs are pushed to CloudWatch and job status updated back into Oracle job table for auditing.

Speaker notes: This slide reassures ops/infra folks that enterprise scheduling and auditing remain intact while moving compute to AWS Batch.

Slide 8 — Security & Data Governance (calm, confident tone)

Bullets: - **Secrets Manager** holds DB credentials; rotated as per policy. - **Parameter Store** manages non-sensitive job configs. - **KMS** encrypts secrets and S3 objects; IAM roles follow least privilege. - VPC/private subnets and VPC endpoints for S3/Batch to avoid public exposure. - Job and audit trail stored in Oracle — centralized traceability.

Speaker notes: Emphasize compliance readiness: encryption, rotation, least privilege, and centralized job audit in Oracle.

Slide 9 — Observability & Runbooks (reduces fear)

Bullets: - Structured logs in CloudWatch correlated by job_id/request_id. - Batch job metrics (vCPU hrs, duration) and CloudWatch dashboards for SLA monitoring. - SNS alerts on failures; escalation playbook in runbook (what to check: CloudWatch logs, S3 staging, DB connectivity, secrets). - Logs and intermediate data archived in S3 for forensic analysis.

Speaker notes: Reassure that incidents will be investigated quickly using job_id tracing across systems — show how long mean-time-to-detect/resolution might improve.

Slide 10 — Advantages & Differentiators (emphasize wins)

Bullets: - **Pip-packaged** in JFrog: simple distribution and version control across teams. - **Enterprise scheduler integration** (Control-M) preserved — no disruption to existing ops. - **Ephemeral worker model**: spin-up EC2 for heavy jobs, tear down after — controls cost. - **S3 staging + Batch scaling** enables parallel chunked transfers to accelerate throughput. - **Single source of truth** for jobs/args in Oracle — reduces mismatch between scripts and scheduler.

Speaker notes: Speak to ROI: fewer broken runs, faster onboarding of new pipelines, easier upgrades.

Slide 11 — High-level AWS Cost (POC vs Production) — make it practical

Bullets (copy-ready): - Main cost drivers: - **AWS Batch / EC2 vCPU-hours** for worker runs (variable, tunable via concurrency and spot instances). - **S3** for staging and archived logs (GB-month + request costs). - **CloudWatch** logs ingestion and retention costs. - **Secrets Manager** and **KMS** request/storage costs. - **ECR** storage for images and occasional data transfer. - **Always-on EC2** for Control-M agents (baseline fixed cost).

Example ballpark (adjust with real metrics): - **POC (low concurrency, intermittent):** `$200–$1,000/month`. - **Pilot / small production:** `$1,500–$5,000/month`. - **Medium production (steady pipelines):** `$5,000–$20,000+/month` depending on vCPU-hours and retention policy.

Speaker notes: Explain these are estimates; propose a short cost-measurement pilot (2 weeks of real runs) to provide exact numbers.

Slide 12 — Demo Goals & Success Criteria (set expectations)

Bullets: - Demonstrate an end-to-end run launched via Control-M → .prm → AWS Batch → DataTran. - Validate: job status in Oracle, CloudWatch logs, S3 staging artifacts, and row counts in target Oracle. - Show secret retrieval from Secrets Manager and param retrieval from Parameter Store. - (Optional) Force a controlled failure to show retry and alert path.

Speaker notes: Be explicit: "This demo is scripted. If anything goes wrong I'll show the logs and explain how we diagnose and fix it quickly."

Slide 13 — Demo Script (copy-ready, step-by-step)

Steps to perform (copy & paste): 1. Open Control-M and show the scheduled job for DataTran (job name and last run status). 2. Show `.prm` file contents (parameters: source_conn, target_conn, table, chunk_size, concurrency). 3. Trigger the job (or show a recent successful run) — show the parameter flow into Jenkins/ Terracore. 4. Watch AWS Batch job lifecycle (Queued → Running → Succeeded) in Console. 5. Tail CloudWatch logs for job_id and highlight: start, chunk progress, completed rows. 6. Inspect S3 staging for intermediate CSVs or scripts mounted. 7. Run `SELECT COUNT(*)` on source and target Oracle tables and show the counts match. 8. Check Oracle job table to show job step history and arguments recorded. 9. (Optional) Simulate failure (missing secret): show SNS alert and CloudWatch log with retry attempts.

Speaker notes: Keep the demo tight (7–9 minutes). Have screenshots ready as fallback and a pre-run for heavy operations.

Slide 14 — Limitations & Risk Mitigation (be candid)

Bullets: - Not intended as a full CDC/real-time streaming replacement — best for bulk and scheduled incremental loads. - Complex transformations still better in an ETL/ELT tool — DataTran focuses on movement and lightweight transforms. - Connector maturity varies — we should validate connectors for each new source. - Mitigations: add connector tests in CI, run short production pilots, define SLOs and runbooks.

Speaker notes: Transparency builds credibility — present risks and how we will address them.

Slide 15 — Ask & Next Steps (clear call to action)

Bullets (copy-ready): - Approve a non-prod pilot: run 2 weeks of realistic loads (provide sample dataset, target schema, and credentials). - Assign: Security reviewer for KMS/Secrets, Ops owner for runbooks and Control-M integration. - Deliverables after pilot: exact AWS cost report, connector tuning notes, runbooks, and production readiness checklist. - Proposed timeline: Pilot (2–3 weeks) → Harden & SLOs (2–4 weeks) → Production cutover (as scheduled).

Speaker notes: End with a confident close: "If the team signs off on the pilot, I'll lead execution and deliver a full readiness report at the end." Ask for approvals.

Slide 16 — Appendix: Useful snippets & runbook excerpts

Items to include for engineers: - Example `.prm` file template and parameter descriptions. - Sample CloudWatch query to find failed job runs by job_id. - Example SQL to validate row counts and checksum comparisons. - Retry/backoff policy: retry N times with exponential backoff; escalate on persistent failure. - ECR image tags and pip package coordinates in JFrog.

Speaker notes: Offer to share the repo and templates after the meeting.

Short, persuasive speaker script (copy-ready)

Opening (20–30s): "Hi — I'm <name>. We've taken DataTran, originally a Snowflake→Oracle tool from XU team, and built a pip-distributable, cloud-native, enterprise-ready framework that plugs into Control-M and AWS. In this POC we proved the pattern — now I'll show the architecture and a short demo."

Transition to demo (10s): "I'll show an end-to-end run, and then ask for approval to run a non-prod pilot so we can measure costs and SLA behavior with real workloads."

Closing ask (15–20s): "If you approve the pilot, we'll validate costs and connector behavior within two weeks and deliver a production readiness report. I'll own the execution and report back with concrete recommendations."

Designer hints (quick)

- Use a storytelling arc: Pain → Solution → Evidence (architecture & demo) → Ask.
- Put one clear CTA on the final slide (Approve pilot / Assign security & ops owner).
- Use screenshots for demo slides and a simple AWS architecture diagram (icons for ECR, Batch, S3, Secrets Manager, KMS, CloudWatch, SNS).
- Keep slides visually light; speaker notes contain the details.

End of document