Pyne Data Engineer Candidate Case

1. Introduction

Design, build, and ship a reliable end-to-end data pipeline and analytics product on Google Cloud Platform (GCP). You will ingest semi-structured data from a public API, model it with dbt, automate deployments through CI/CD, and communicate insights with a concise data story. Estimated effort: 6–10 focused hours; a time-boxed proof-of-concept is sufficient.

2. Scenario

Product Management wants a public **"Dog Breed Explorer."** They've asked for a curated analytics layer exposing facts about dog breeds (life span, size class, temperament, etc.) and an automated pipeline that refreshes daily.
Source API: `https://api.thedogapi.com/v1/breeds`

3. Tech Scope

Layer	Required / Suggested Services & Tools
Ingestion	**You must use dlt (Data Load Tool) from www.dlthub.com to extract and load the Dog API data.** dlt is an open-source Python library that lets you declaratively pull data from APIs and other sources and load them into warehouses with automatic schema management, incremental loading, and secrets/config handling. Run it from Cloud Functions (Python 3.12) or Cloud Run.
Storage	Cloud Storage (raw JSON) & BigQuery (raw → curated)
Orchestration	Cloud Scheduler or Cloud Workflows
Transformation	**Create a dbt project targeting BigQuery.** You may use **dbt Core *or* dbt Cloud**—your choice.
CI/CD	GitHub Actions or Cloud Build
Observability (bonus)	dbt tests, Great Expectations, Cloud Monitoring
Visualisation	Looker Studio (Data Studio) or similar

*You may substitute equivalent OSS/managed services elsewhere if justified—**except ingestion; using dlt is mandatory**.*

4. Tasks & Deliverables

Task 1 – Environment & Repository

1. Create a new GCP project; enable billing and required APIs.
2. Provision least-privilege service account(s).
3. Initialise a Git repo for code, configs, and docs.
   Deliverables: project ID, repo link, brief README describing bootstrap steps.

Task 2 – Data Ingestion (with dlt)

1. Implement a dlt pipeline that:
   - Calls the Dog API breeds endpoint.
   - Writes the full JSON to Cloud Storage (partitioned by run date).
   - Loads the raw payload into a BigQuery `bronze.dog_api_raw` table (schema-on-read acceptable).
2. Schedule the job to run daily at 02:00 UTC via Cloud Scheduler/Workflows.
   Deliverables: source code (dlt pipeline + runner), IaC (Terraform/YAML) if used, Scheduler config, screenshot of data in BigQuery.

Task 3 – Data Modelling with dbt

1. **Create a dbt project targeting BigQuery** (Core or Cloud—your choice).
2. Build:
   - **Staging** models to normalise and type-cast the raw JSON.
   - **Mart** models (e.g., `dim_breed`, `fact_weight_life_span`).
   - ≥3 tests (schema/custom) + auto-generated docs.
3. Parameterise for dev/prod datasets.
   Deliverables: dbt project, compiled docs or hosted site.

Task 4 – CI/CD Pipeline

1. Configure GitHub Actions or Cloud Build to:
   - Run unit tests (if any) and `dbt run` + `dbt test` on every PR.
   - On merge to `main`, deploy the ingestion job (dlt runner) and execute `dbt run` against prod.

2. Include a badge or build log in the README.
   Deliverables: workflow YAML, screenshot of a successful build.

Task 5 – Data Quality & Observability (Optional)

- Add Great Expectations or extend dbt tests to validate row counts, null rates, value ranges.
- Send failures to Cloud Monitoring & an alerting channel.

Task 6 – Visualisation & Storytelling

1. Build a dashboard answering at least two:
   - Which breeds have the longest predicted life span?
   - Distribution of breeds by weight class.
   - Top temperaments among family-friendly breeds.
2. Add a ≤300-word narrative in the README on findings & business impact.
   Deliverables: dashboard link or PDF export, narrative section.

5. Evaluation Rubric

Category	Weight	What we look for
Cloud Architecture	20%	Secure, cost-efficient GCP usage; IaC is a plus
Code Quality	15%	Idiomatic Python/dbt, meaningful commits, tests
Automation	15%	Functional CI/CD; reproducibility from scratch
Data Modelling	20%	Clear marts, naming, documented lineage
Data Quality	10%	Tests/monitoring catching bad data early
Storytelling & Viz	10%	Insightful, well-designed dashboard & narrative
Documentation	10%	Clear README, run-book, architecture diagram

Score ≥70% passes. Bonus features can offset minor gaps.

6. Submission & Presentation

Submit the Git repo URL and a service-account key (or Workload Identity) with viewer rights 48 hours before debrief. Be prepared (~30 min) to:

- Demo the dashboard.
- Walk through code & pipeline.
- Discuss trade-offs & future improvements.

Good luck—we look forward to your solution!