ATK (Architecture & Tooling Kit) – Technical Manual

1. What ATK Is

ATK (Architecture & Tooling Kit) is a **data-driven cloud architecture decision system** combined with a **Cloud Technology Navigator**. It is designed to:

- Maintain a **master catalog of cloud services and third-party platforms**
- Encode **architecture rules and constraints** as data, not hard-coded logic
- Produce **repeatable, explainable decisions** for cloud designs
- Visually explore providers, services, and tools in a browser

ATK is intentionally **static + deterministic**: - No backend services required - No runtime API calls - Everything is driven from JSON → generated JS → UI

This makes ATK safe for offline use, audits, GovCloud environments, and long-term maintainability.

---

2. High-Level Components

ATK is composed of four major layers:

1. **Master Data Layer** – canonical source of truth
2. **Build / Generation Layer** – converts data into runtime artifacts
3. **Decision Engine** – evaluates requirements and produces architecture outputs
4. **Navigator UI** – human-friendly exploration and validation

```
[ Master Data (JSON) ]
          ↓
[ Build Scripts (Node.js) ]
          ↓
[ Generated JS Artifacts ]
          ↓
[ Decision Engine ] ↔ [ Cloud Tech Navigator UI ]
```

---

3. Repository Structure (Canonical)

```
ATK/
├─ data/
```

```
│   └─ master/
│      └─ master-matrix.json
│
├─ scripts/
│   ├─ build.mjs
│   ├─ decide.mjs
│   ├─ decide-resolve.mjs
│   └─ report.mjs
│
├─ web/
│   ├─ generated/
│   │   ├─ catalog.generated.js
│   │   ├─ decision.generated.js
│   │   ├─ capability-map.generated.js
│   │   └─ version.generated.js
│   │
│   └─ cloud-tech-navigator/
│      ├─ cloud-tech-navigator.html
│      ├─ cloud-tech-navigator.aws-first.html
│      ├─ aws-services.js
│      ├─ gcp-services.js
│      ├─ azure-services.js
│      ├─ oci-services.js
│      ├─ f5-services.js
│      ├─ ocp-services.js
│      └─ *.js (tool / option modules)
│
├─ docs/
│   └─ (this manual)
│
├─ package.json
└─ VERSION.json
```

---

4. Master Data Layer (Source of Truth)

`data/master/master-matrix.json`

This is the **most important file in ATK**.

It defines: - Capabilities (what systems must do) - Patterns (architectural approaches) - Service mappings per cloud - Third-party tools (F5, OCP, etc.) - Metadata used by both the decision engine and navigator

Design rules

- **Nothing is hard-coded** in JS
- New services are added here
- New clouds/tools extend the schema
- Everything downstream is regenerated

This file is updated **quarterly** (recommended) as cloud services evolve.

---

5. Build / Generation Layer

`scripts/build.mjs`

The build script:

1. Loads `master-matrix.json`
2. Validates schema integrity
3. Generates runtime JS files under `web/generated/`

Generated outputs: - `catalog.generated.js` – navigator index - `decision.generated.js` – rules + patterns - `capability-map.generated.js` – capability → service resolution - `version.generated.js` – build/version metadata

Run:

```
npm run build
```

This step is **mandatory** after any master data change.

---

6. Decision Engine

Concept

The decision engine answers:

> "Given these constraints, what architecture patterns and services make sense?"

Inputs

Provided as JSON:

```
{
  "cloud": "aws",
  "data_classification": ["CJIS"],
  "internet_exposed": true,
  "rto_minutes": 30
}
```

Core logic

1. Evaluate **rules** against inputs
2. Determine **required capabilities**
3. Score **architecture patterns**
4. Resolve **capabilities → services** per cloud
5. Identify **gaps** (unmapped capabilities)

Scripts

- `decide.mjs` – evaluates rules & patterns
- `decide-resolve.mjs` – resolves services
- `report.mjs` – outputs structured reports

Output

- Ranked architecture patterns
- Explicit reasoning (why chosen)
- Concrete cloud services
- Audit-friendly explanation

---

7. Cloud Tech Navigator

Purpose

The Navigator answers:

> "What services exist, and how do they relate?"

It is **not** a decision engine — it is a **visual exploration tool**.

How it works

- Loads generated catalog + provider/tool modules
- Populates providers, categories, services dynamically
- Supports AWS, GCP, Azure, OCI
- Supports third-party platforms (F5, OpenShift)

Providers vs Tools (important)

- **Providers:** `window.CDK.providers.*`
- aws, gcp, azure, oci
- **Tools:** `window.CDK.tools.*`
- f5, ocp, vmware, etc.

These are intentionally separated to prevent data collisions.

---

8. Data Flow (End-to-End)

```
master-matrix.json
   ↓ (build.mjs)
web/generated/*.js
   ↓
Decision Engine (rules + patterns)
   ↓
Resolved Capabilities → Services
   ↓
Navigator (visual validation)
```

Everything flows **one direction**. The UI never mutates data.

---

9. Updates & Maintenance

Quarterly update process

1. Update `master-matrix.json`
2. Run `npm run build`
3. Review generated artifacts
4. Commit changes (optional git)

Adding a new cloud

- Add provider block to master data
- Create `*-services.js`
- Include in navigator HTML

Adding a new tool (example: Palo Alto)

- Add tool entry to master data
- Create `paloalto-services.js`
- Register under `window.CDK.tools`

---

10. Versioning

ATK uses **explicit version files**, not git-derived magic:

- `VERSION.json` – authoritative version
- `version.generated.js` – runtime visibility

This allows ATK to run: - without git - offline - in restricted environments

---

11. What ATK Is NOT

- Not a live cloud pricing engine
- Not an IaC generator (by design)
- Not tied to any single provider
- Not dependent on SaaS APIs

ATK is a **decision support and architectural reasoning system**.

---

12. Design Philosophy (Why this works)

- Data > code
- Deterministic > dynamic
- Explainable > opaque
- Static > fragile

ATK is built to survive: - cloud churn - vendor renames - long certification cycles - air-gapped environments

---

13. Next Enhancements (Optional)

- CI pipeline to validate master schema
- Automated diff reports between quarters
- Export to PDF / Markdown bundles
- Link decision output → IaC templates

---

14. Summary

ATK provides: - A **single source of architectural truth** - A **repeatable decision engine** - A **human-friendly navigator** - A **future-proof update model**

This manual should live in `docs/` and be updated alongside major version changes.