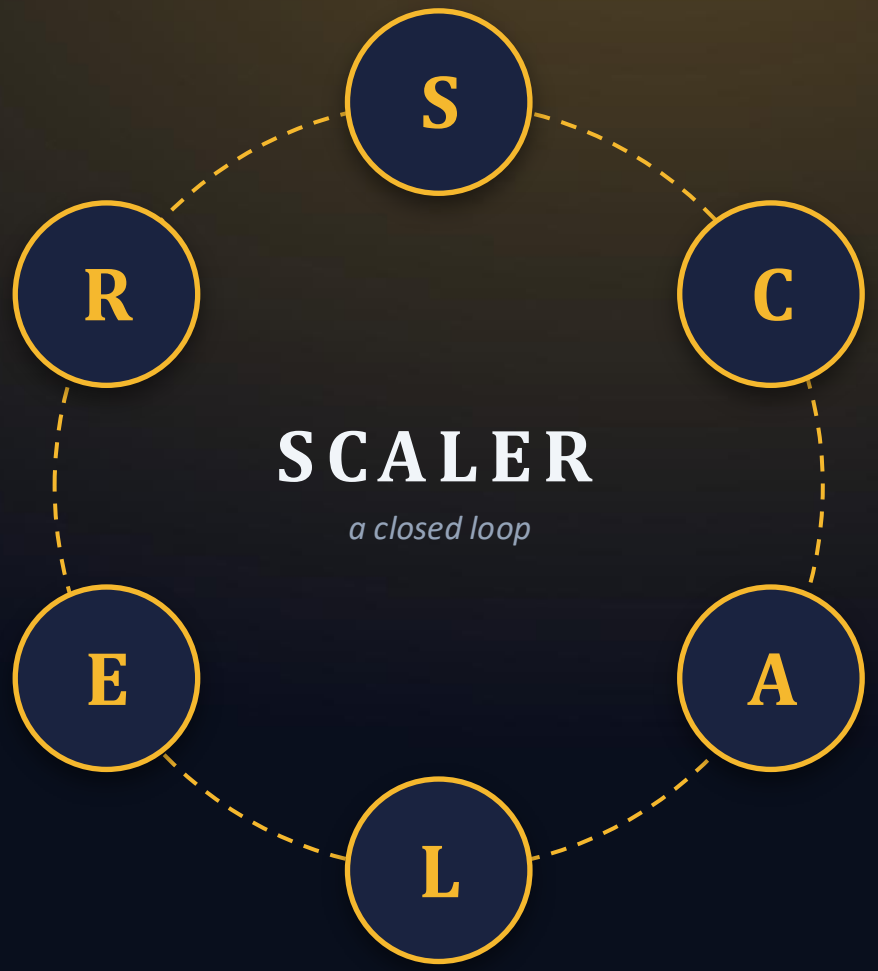
CORNCON 12 · DAVENPORT · OCTOBER 2, 2026

From Click Ops to Closed Loop

Rotating **1,000+** database credentials without an outage



Mayank Sethi

Principal Data Operations Engineer

This work was done at a previous employer. Details are generalised. Opinions expressed are my own and do not represent those of any current or former employer.

BEFORE WE START

This autumn, Snowflake is ending password sign-in for service accounts.

Most teams are about to find out the credential was never the hard part.



Who uses this account?



Who owns it?



Do we even need it?

This talk is about those three questions, told through a 1,000+ account case study from 2024.

How the platform was actually run

A stale inventory

A list existed. It had drifted from what the platform actually held.

Stale owner records

Every account had a named owner. Many of them had moved on years earlier.

No schedule

Nothing expired. Nothing was reviewed. Credential ages ran into years.

No rotation

Not one service credential had ever been changed.

On paper it was all documented.

None of it was still true.

The usual controls were in place.

2024: the Snowflake credential incident broke. Every security team spent a week answering the same question.

ALREADY IN PLACE

- SSO
- MFA
- Network policies
- PrivateLink
- SSL decryption
- Tri-Secret Secure
- Encryption
- Admin monitoring

THEN THE DUST SETTLED



The real finding surfaced somewhere else.

The controls everybody talks about were fine. The exposure was in the object class nobody looks at.

0

service credentials had ever been rotated.

Not one, in the life of the platform.

Two instincts. Both wrong.



✕ OUTAGES

Rotate everything now

Fastest path to closing the exposure — and to breaking production across teams who had no warning.



✕ EXPOSURE STAYS OPEN

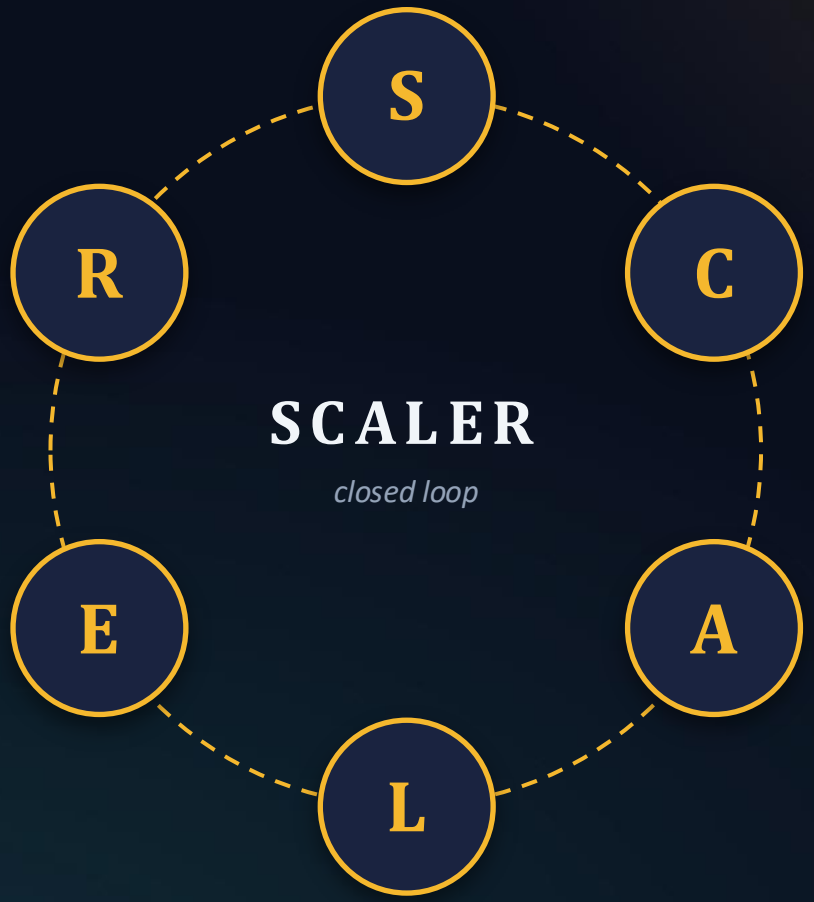
Stand up a working group

Feels responsible. Leaves the exposure open for months while the deck gets reviewed.

Both are common. Both are wrong.

The question is not how fast you rotate — it is whether you know what you are rotating.

SCALER — six stages, in this order



01	**SCALE**	Find every account
02	**CLASSIFY**	Resolve it to an app
03	**ASSIGN**	Give it a human owner
04	**LIFECYCLE**	Set a policy that expires
05	**ENFORCE**	Make the safe path the easy one
06	**ROTATE**	Change it, reversibly

The order is the point. Every stage that gets skipped shows up later as an outage — or an account nobody can explain.

Finding what the records had lost

1,000+
service accounts enumerated

150+
application teams affected

25%
with a stale owner, or none

RESOLVED AUTOMATICALLY · 75%

THE ENTIRE PROJECT · 25%

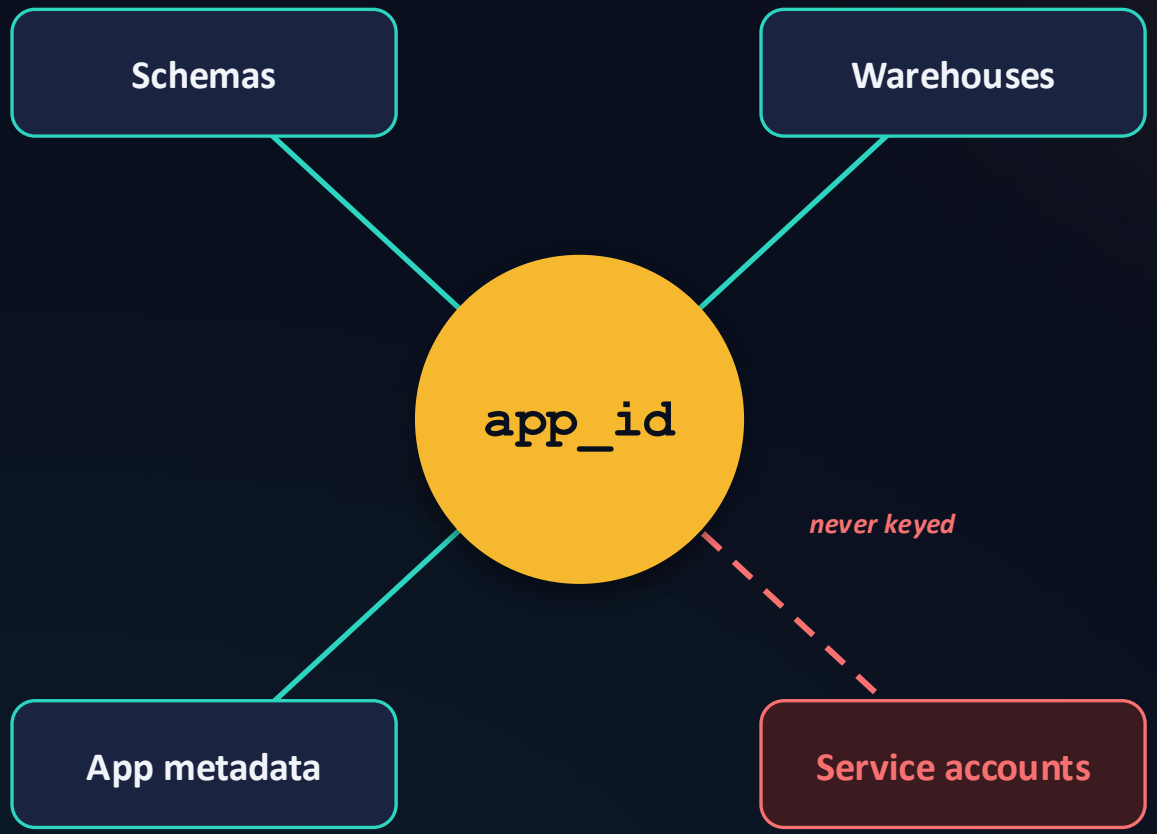


Three quarters of the estate resolved automatically. The last quarter was the entire project.

Enumerated from ACCOUNT_USAGE.USERS rather than the inventory — the platform is the one source that cannot be out of date.

Login history gave us **last successful use**, and where the connection came from.

The join key already existed



`app_id` already tied together schemas, warehouses and application metadata. It had never been applied to service accounts.

Every other object class was keyed. Credentials were the one left out of the model.

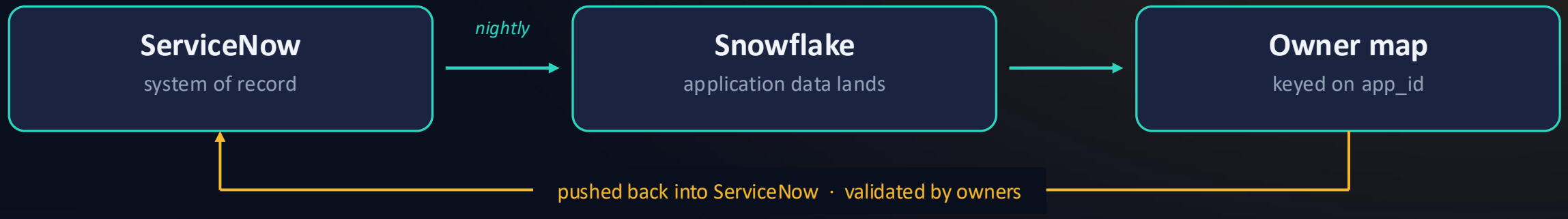
ONE JOIN RETURNED

Application **Environment** **Criticality**

The environment split is what later allowed different deadlines for non-production and production.

This was not a data problem. It was a missing join.

Ownership you confirm, not guess



Then we gave owners their own data back

- Their accounts
- Credential age in days
- Connection origin
- **Last successful login**

Stored procedures to pull it themselves, any time, without asking us — plus a dashboard filterable by app id, app name, business owner and tech owner.

THE SIDE EFFECT WE DID NOT PLAN FOR

Once owners could see last successful login, they started asking us to disable accounts.

The quarter that would not resolve



The requesters were gone

Left the firm or moved teams. The trail ended at a name that no longer applied.



The naming standard paid off

Domains were enforced in service account names from day one. That discipline became a second join key.



It came down to people

The domain narrowed the orphans to a handful of humans to go and ask.



Some we did not chase

No login in 90+ days — disabled, not chased, with a self-service request to turn them back on.

The model gave us the 75%.

The naming convention gave us the rest.

Policy as the clock

DURING THE PROGRAMME

MAX_AGE = 0

Deliberate. Nothing expires underneath you while you are still rotating. Oldest accounts first.

ONE POLICY, EVERY SERVICE ACCOUNT

MAX_AGE = 365

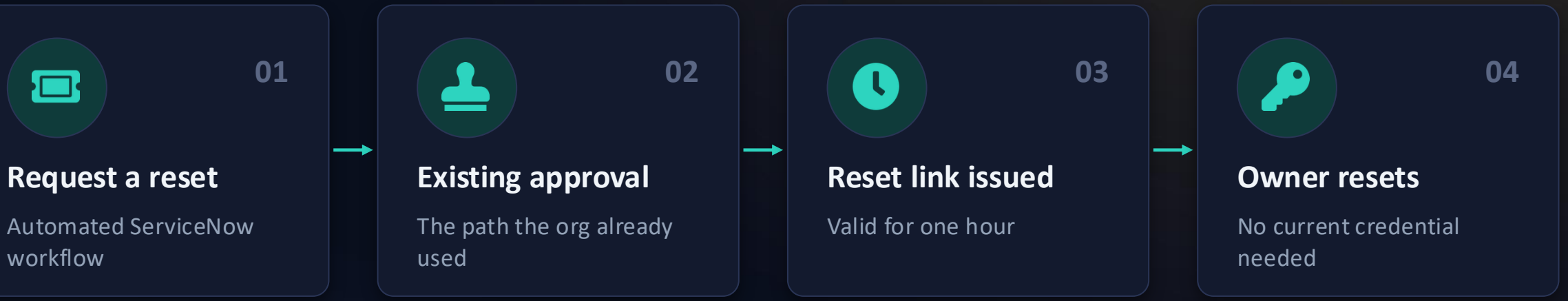
A single 365-day expiry. The policy becomes the deadline for everyone remaining. No chasing required.

THE COUNTDOWN EVERY OWNER RECEIVED



Firm-wide comms carried separate non-production and production dates.

Make the safe path the easy path



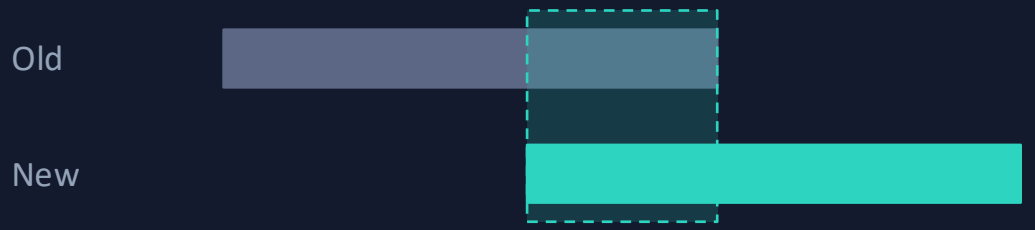
So what secures it
The link is admin-generated, time-boxed to an hour, and only issued against an approved request.

Never on the table
Emailing a credential in plaintext.

Self-service is not the opposite of governed. It is what governance looks like when it works at scale.

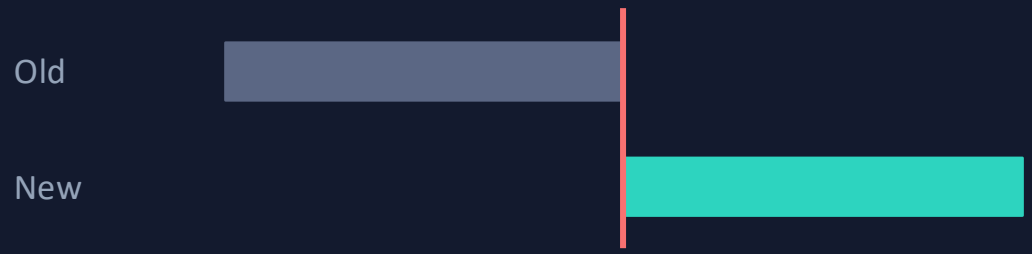
Reversible, because the platform is not

WHAT EVERY SECRETS MANAGER ASSUMES



overlap window: cut over, then retire

WHAT SNOWFLAKE GIVES YOU



no overlap: one instant, one credential

Revert path: temporarily allow the prior credential back

A control we loosened on purpose, for the programme only, so a failed rotation was recoverable in minutes rather than escalated. We used it on fewer than five accounts.

A trade-off, not a best practice.

We chose recoverability over strict history.

Validation belongs to the owner

Owner rotates

Through the automated ServiceNow workflow

Owner confirms

The application reconnects and behaves normally

Works done, the policy clock starts

Fails revert path, fixed in minutes

THE DATA OWNERS HAD TO CHECK FOR THEMSELVES

Last successful login

Credential age in days

Connection origin

WHY THE OWNER, NOT THE PLATFORM TEAM

We can see a login succeed. Only the owner knows the job downstream still works.

What we did not see coming

1. **Credential rotated** →
2. **Apps retry with stale credentials** →
3. **Accounts hit the lockout threshold** →
4. **Shared network egress hides the source**

Rotation locked people out — and connections arriving through shared network egress were almost impossible to attribute to a source.

The fix was not elegant

1. A separate, more tolerant lockout setting for the affected users
2. A scheduled job clearing lockouts every fifteen minutes while owners fixed their apps

Intended for a few days.

It ran for weeks.

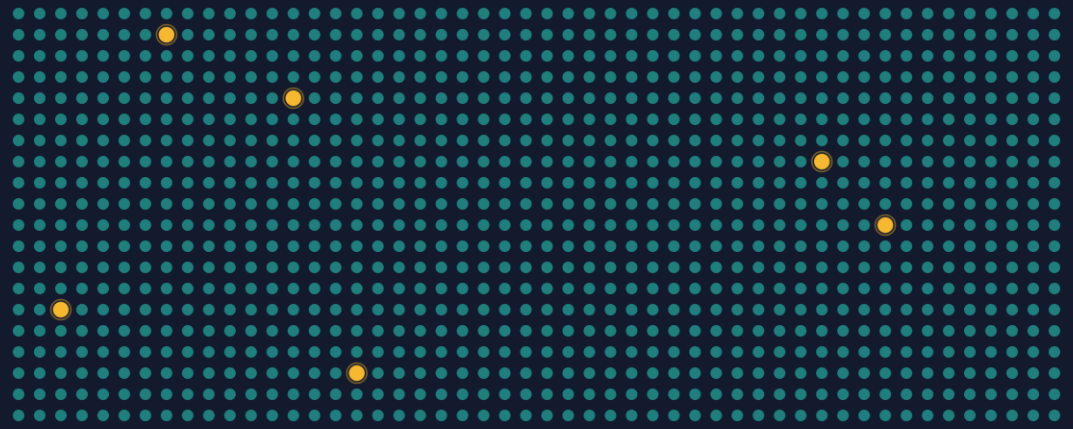
What three people delivered

1,000+
CREDENTIALS ROTATED

150+
APPLICATION TEAMS

80%
DONE IN 3 MONTHS

0
PRODUCTION OUTAGES



Every dot is one credential. Gold never traced cleanly.

THE HANDFUL THAT WOULD NOT RESOLVE

Fewer than five accounts we never traced cleanly to a source — the same ones that kept locking out, and the only ones we had to revert. The unlock job ran every fifteen minutes for several weeks to cover them.

Reporting it is the only reason to trust the other number.

Rotation bought time. Inventory is the real work.



CONTAIN FIRST

Rotation was the immediate fix available.

Moving 1,000+ connections to key-pair or OAuth means code changes in every team. Rotation closed the exposure now.

THIS MONTH, FOR EVERYONE ON SNOWFLAKE

Swapping the credential is the mechanical part. Finding each account's consumers, owner and needed access is stages 1–4.

S C A L E R

Same loop, stronger credentials

STAGE	2024		TODAY
Scale · Classify · Assign · Lifecycle			Unchanged. Still the hard part, whatever the credential.
Enforce	Self-service credential reset via ServiceNow	→	Service users on key-pair or OAuth. No passwords.
Rotate	No overlap window; revert to the prior credential	→	Two active keys: add the new one, cut over, retire the old
Validate	Owners confirm manually	→	Automated first check: login history shows the account reconnecting
Scan	Trust Center arrived mid-project	→	Scanners on a schedule from day one

Four questions for your team on Monday

How many?

How many service accounts do we have? Nobody knows the number.

How old?

When was the oldest one last rotated? Ask for the distribution, not the average.

Who owns them?

How many have an owner who still works here? And which have not logged in for 90 days?

Still on passwords?

Which still authenticate with a password? That list is your Phase 3 exposure.

All four live in the user catalogue and the login history — Snowflake, Postgres, SQL Server, Oracle, Active Directory. Every platform has both.

Four questions. One afternoon.

You will not like the orphan rate.

What generalises

> The data was already there.
>
> The join was not.

app_id tied together every other object class in the platform. It had never been applied to service credentials — and that single omission was the whole ownership problem. The naming convention was the second key, in plain sight the whole time.

1. **Sequence beats speed**

   Discovery before rotation. Ownership before automation. Policy as the deadline, not the announcement.

2. **Maps rot — feed them**

   An ownership map you build once is stale the week after. Ours refreshed nightly from the system of record.

3. **Visibility creates action**

   Give owners their own data and they will disable accounts you did not know to ask about.

4. **Publish the mess**

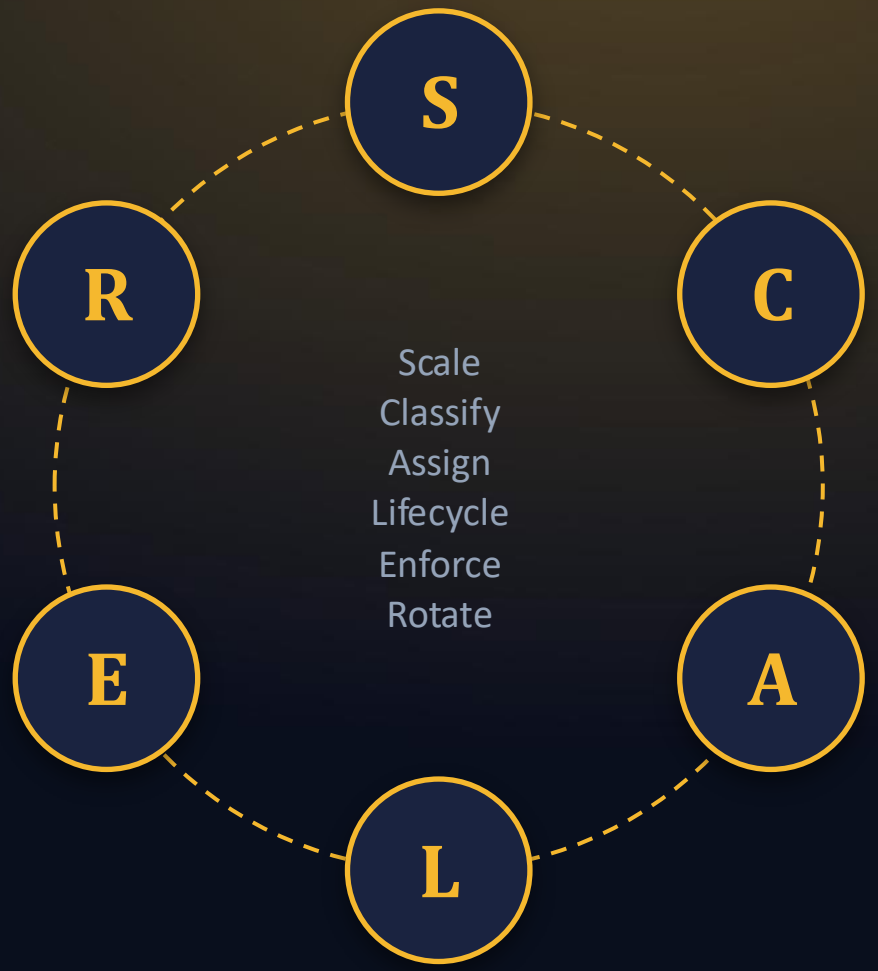
   Report the accounts you could not resolve. The clean number is only believable next to the messy one.

ONE LAST THING

Facing Snowflake’s password cutoff this month?

Rotation isn’t the hard part. The inventory is.

Any platform with a programmatic API has this problem. Most have the join key too.



Thank you · Questions?

Mayank Sethi · Principal Data Operations Engineer

Opinions expressed are my own and do not represent those of any current or former employer.