DEVOPS ON SNOWFLAKE

Database Change Management and CI/CD

Mike Wies | Data Architect

AGENDA

- Brief overview of DevOps
- Snowflake DevOps Lifecycle
- Database Change Management Approaches And Tools
- Standards

BRIEF OVERVIEW OF DEVOPS

Continuous Delivery

Continuous Integration

CI/CD

Docker

Containers

Agile

Build Agent

Git

GitHub

Continuous Deployment

DevOps

Unit Testing

DevSecOps

Automation

Release Pipelines

Test Automation

Jenkins

GitHub Actions

Release Automation

Azure DevOps

WHAT IS DEVOPS

AWS defines DevOps as:

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

Azure defines DevOps as:

A compound of development (Dev) and operations (Ops), DevOps is the union of people, process, and technology to continually provide value to customers.

What does DevOps mean for teams? DevOps enables formerly siloed roles—development, IT operations, quality engineering, and security—to coordinate and collaborate to produce better, more reliable products. By adopting a DevOps culture along with DevOps practices and tools, teams gain the ability to better respond to customer needs, increase confidence in the applications they build, and achieve business goals faster.

More Simply:

DevOps is a set of processes, procedures and technology which perform automatic and/or programmatic integration (CI) and deployment (CD) of code

CI/CD

Continuous Integration

How do you manage concurrent development of things

Key Concepts:

- Source Control
- Branching Strategies
- Code merging
- Declaring a release Candidate
- (Database/Schema) Change Management

Continuous Deployment

How are those changes validated and deployed to environments in a reliable way

Key Concepts:

- Automatic/programmatic validation
- Environment promotion
- Release Rollback

Orchestration

Technology which automates the CI & CD steps

Examples:

- Azure DevOps (Visual Studio TFS)
- Jenkins
- GitHub Actions

DevOps Orchestration tool as a type of scheduler. Natively, they do not perform the steps - they trigger the steps.

Typically take a variety of configuration files(JSON/YAML) and scripts(bash, powershell, python) which the orchestration chains together in a predetermined way

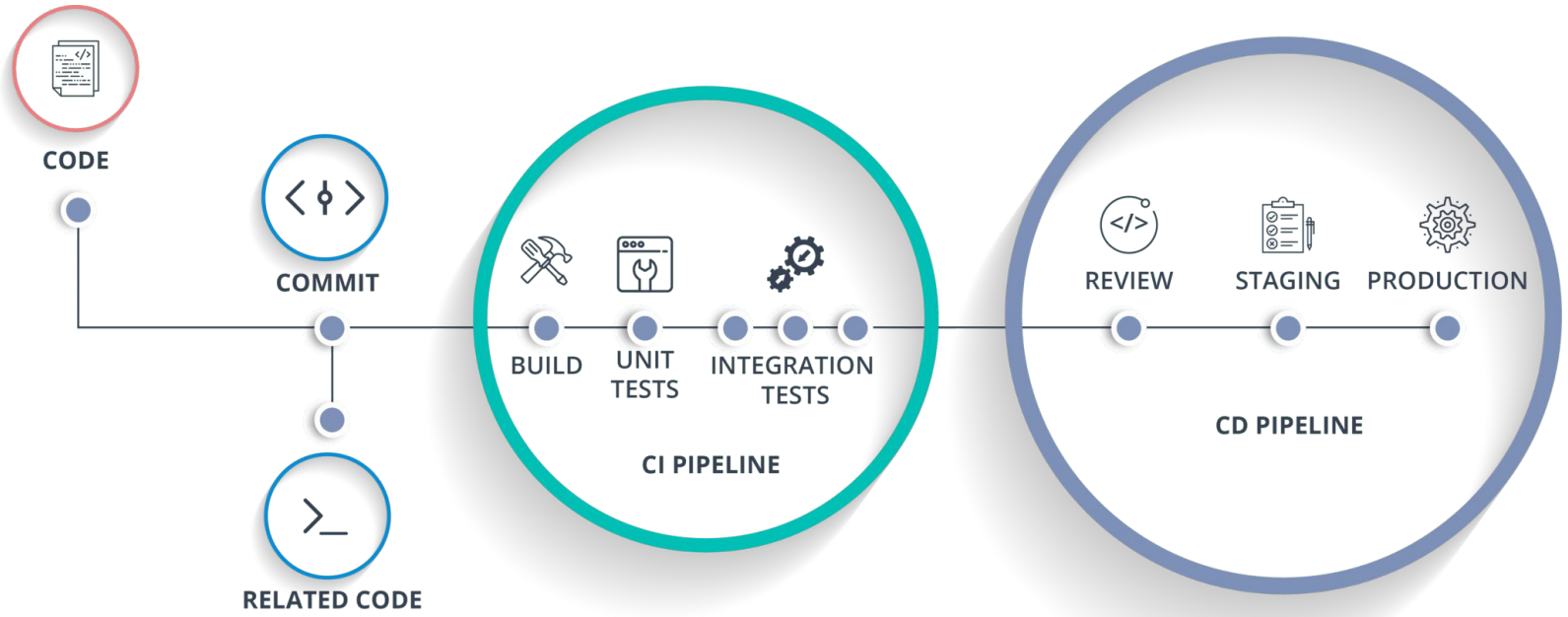
WHY DOES DEVOPS MATTER?

DevOps is one of the key pillars that enables agile software delivery and the following business results:

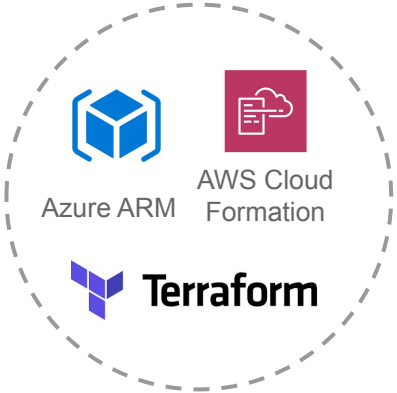
- ✓ Faster delivery of software
- ✓ Greater efficiency
- ✓ Repeatable, measurable processes
- ✓ Improved quality
- ✓ Faster innovation
- ✓ Ability to build the right solution for a given problem
- ✓ Cheaper delivery
- ✓ Greater return on investment and profitability

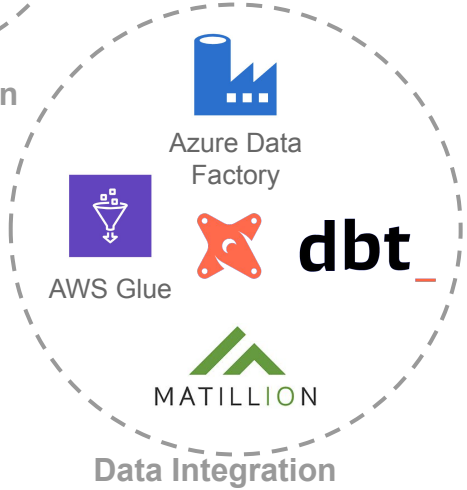
CI/CD PIPELINES

This diagram shows the scope and relationship of CI and CD activities in a release pipeline.



Taken from CI/CD Pipelines by Solidstudio





Snowflake DevOps Playbook

SNOWFLAKE DEVOPS LIFECYCLE

THE CHALLENGES WITH DATABASE DEVOPS

Database DevOps (especially for data engineering) is difficult and has unique challenges compared with DevOps for applications.





State

Application code is stateless

Database contains valuable business data

Change structure and data without loss

Handcrafting release scripts is error prone

Downtime

Application servers are easy to swap in/out

Database servers are very difficult to swap in/out (even in a cluster)

Can sometimes swap database or tables in/out

Roll Back

Applications are easy to roll back from source control

Databases must be explicitly backed up and restored

Very time consuming

Database unavailable during restore

Testing

Application code is easy to test with unit tests

Unit testing for databases is challenging

Unit testing requires test data generation and management, which gets complicated quickly

Other

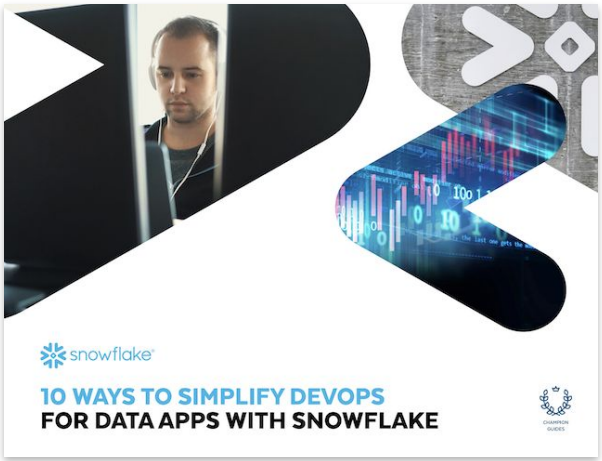
Traditionally, only DBAs modify the database (control)

Prod databases don't match source control (drift)

Database change management is difficult

Adapted from Francois Delport's blog post Continuous Deployment Of Databases: Part 1

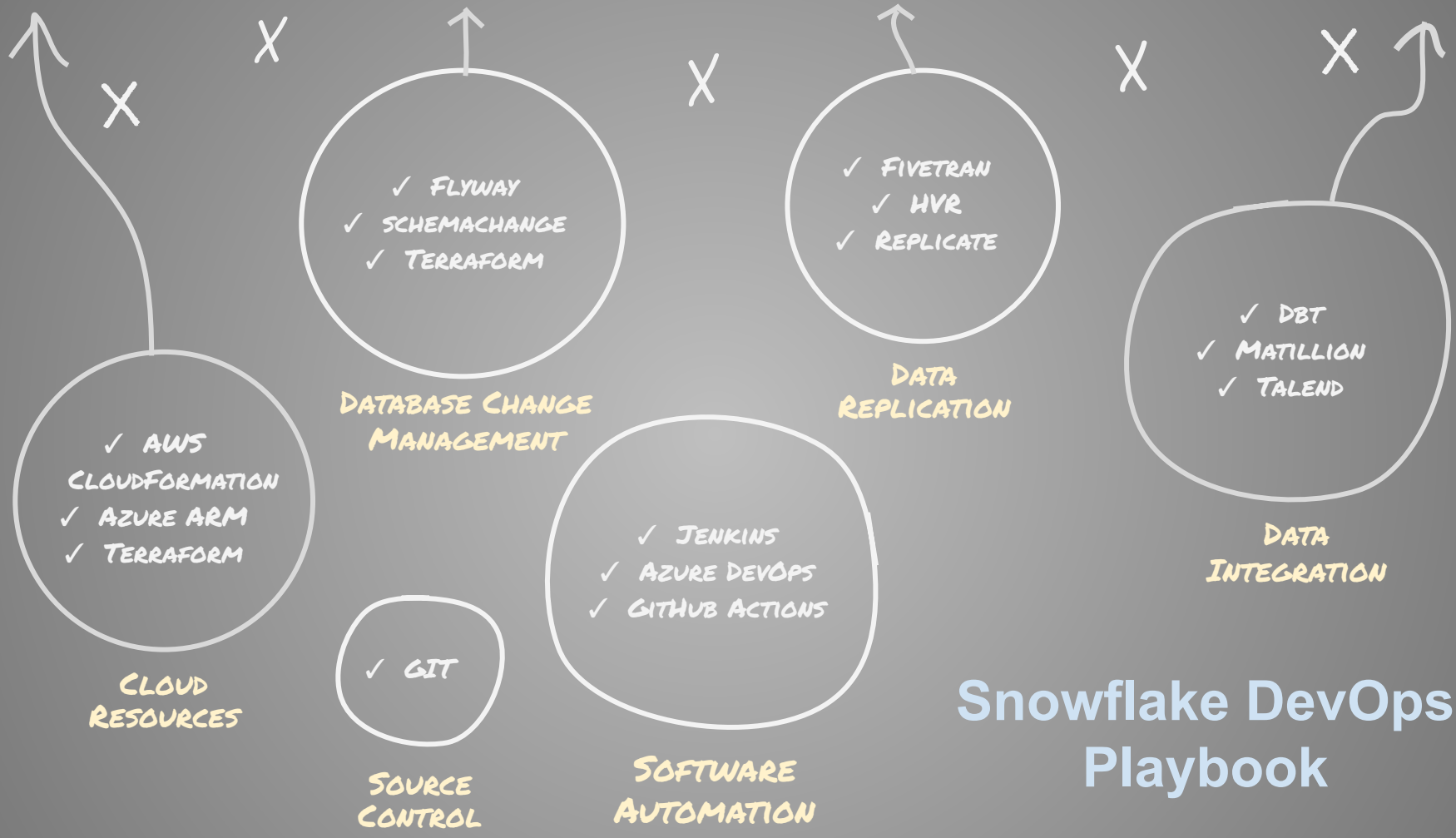
10 WAYS SNOWFLAKE SIMPLIFIES DEVOPS



EBOOK:
https://resources.snowflake.com/ebooks/10-ways-to-simplify-devops-for-data-apps-with-snowflake

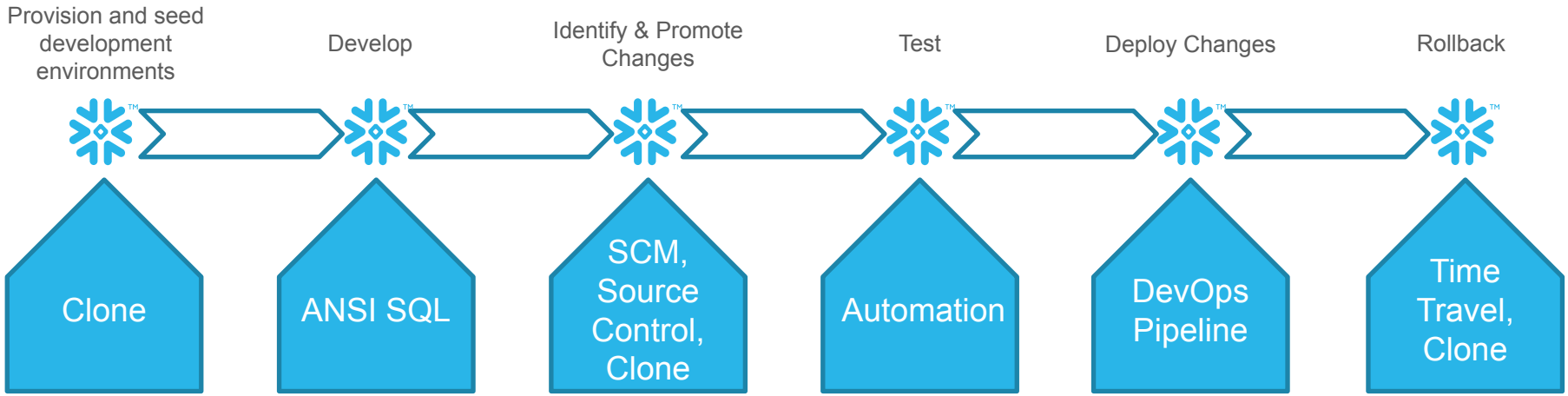
RELATED WEBINAR:
https://www.snowflake.com/webinar/9-ways-snowflake-simplifies-devops-for-data-apps/

1. Rapidly create any number of isolated environments
2. Reduce schema change frequency with VARIANT data type
3. Rapidly seed pre-production environments with production data
4. Easily roll back with Time Travel
5. Instantly scale environments to run jobs fast and cost-effectively
6. Increase velocity with standard SQL
7. Use your preferred programming language
8. Reduce DevOps burden with near-zero maintenance
9. Simplify data pipelines with Streams and Tasks
10. Save time on real-time integration with external services



Snowflake DevOps Playbook

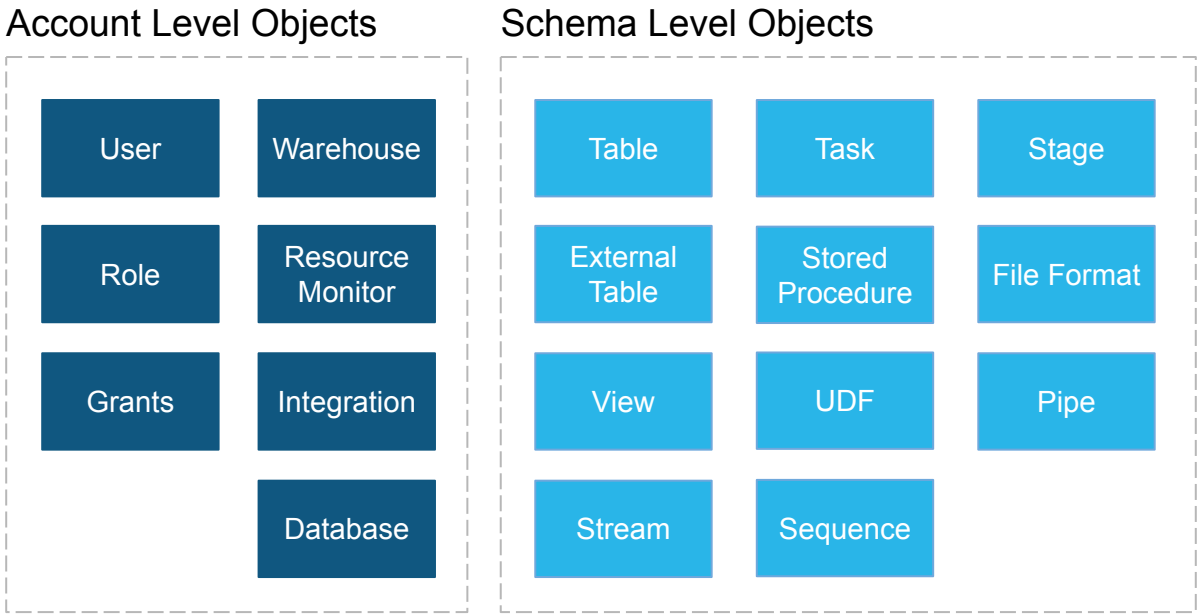
SNOWFLAKE DEVOPS LIFECYCLE



DATABASE CHANGE MANAGEMENT APPROACHES AND TOOLS

SNOWFLAKE OBJECTS

Here is a list of Snowflake objects which must be managed during the development lifecycle



THE TWO DIFFERENT APPROACHES

At the highest level there are two different approaches for Database Change Management (or Database Schema Migration or Schema Change Management).

Imperative

v1

```
CREATE TABLE FOO
(
  Column1     INT
 ,Column2     DATE
);
```

v2

```
ALTER TABLE FOO ADD Column3
VARCHAR;
```

- Series of change scripts that must be applied in the right order
- Must maintain version/state of each database
- Very flexible
- Cumbersome and error-prone

Declarative

```
CREATE TABLE FOO
(
  Column1     INT
 ,Column2     DATE
 ,Column3     VARCHAR
);
```

- Single definition of objects
- Less error-prone
- Data migrations are challenging
- Schema diff tool required
- Great for development

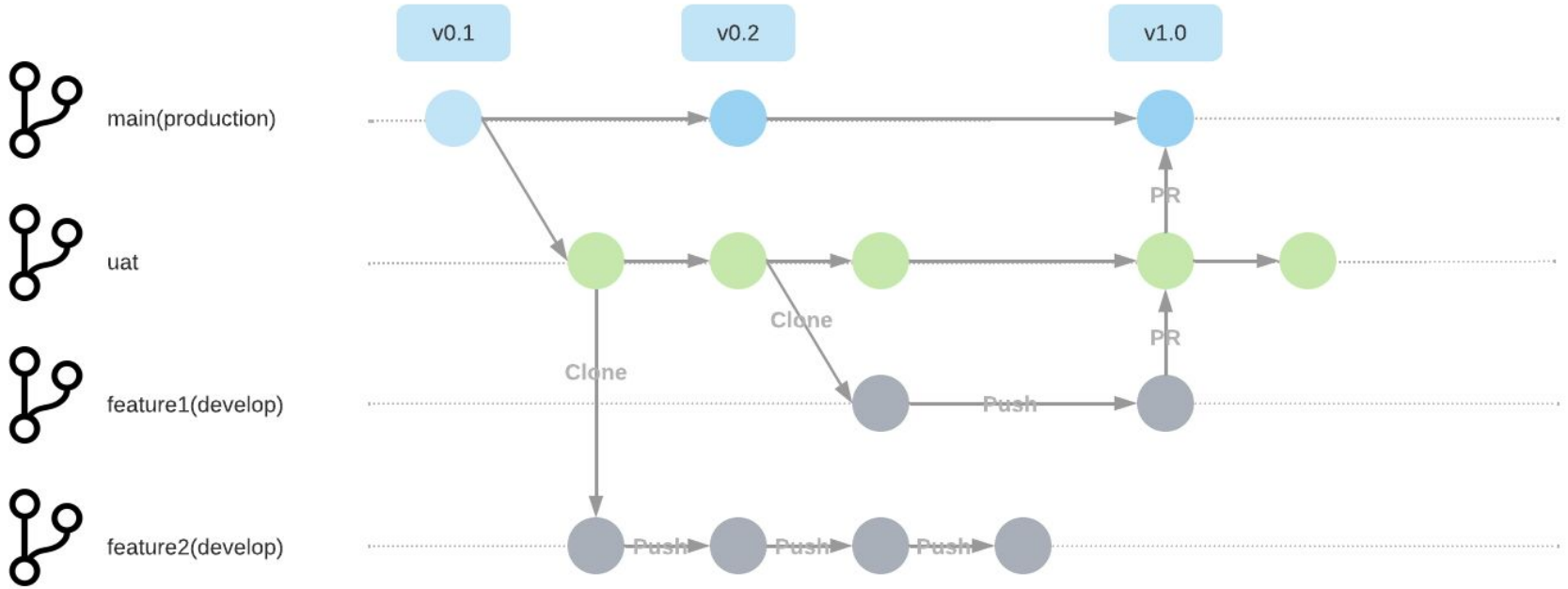
DCM TOOLS FOR SNOWFLAKE

This matrix summarizes the current Database Change Management (DCM) tools that work with Snowflake

	Approach	Tool Language	Script Language	Notes
schemachange	Imperative	python	SQL	Very simple, lightweight flexible tool, similar to Flyway, open source, not an official Snowflake offering. Previously known as snowchange.
Flyway	Imperative	Java	SQL	Mature tool, lightweight/flexible naming, freemium model, undo and other features cost
Sqitch	Imperative	Perl	SQL	Mature tool, supports validating and reverting, why Perl?, pain to install for CI/CD
dbt	Declarative?	python	SQL + Jinja	Primarily a data transformation tool, can use Custom Materializations to manage certain objects (tables, views, etc.)
SQLAlchemy Migration / Alembic	Imperative	python?	python	Based on SQLAlchemy (ORM tool), writing migration scripts in python not desirable
SqlDBM	Declarative	N/A	SQL	SaaS tool, began as modeling tool, developing more robust DCM tools, looks promising, very early still
Liquibase	Declarative?	Java	SQL	Mature tool, open source database-independent library for tracking, managing and applying database schema changes.

STANDARDS

Branching model

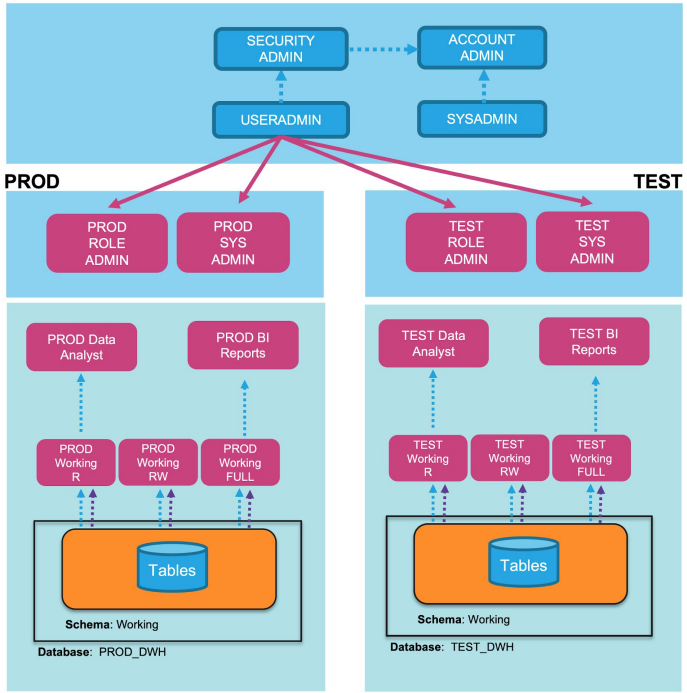


Naming conventions

Business Entity :	Business unit, subordinate entity within the BU or corporate function.
Environment Type :	POC, dev, test, production, etc.
Layer/Type :	Data architecture layer or classifying type within an object type.
Team/Function :	Departmental team, sub-team, or functional area.
Modifier :	Any additional modifier needed to clarify or make the name unique. For the raw layer this should be the name of the source database, otherwise use to remove any ambiguity regarding what type of object is being named.
Privileges :	Only applies to access roles and specifies a logical group of privileges granted to the object.
Object Type :	Database, warehouse, access role, functional role, etc.

Object name pattern : <Business Entity>_<Environment Type>_<Layer/Type>_<Team/Function>_<Modifier>_<Privileges>_<Object Type>

RBAC



DETAILS

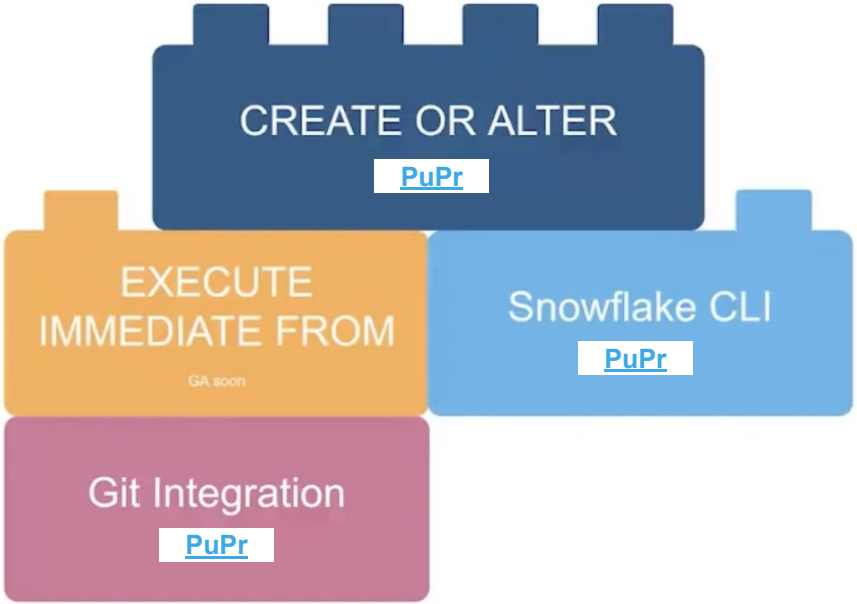
Feature:

- Multi-level Administration.

Benefits:

- Can delegate control.
- Scalable – Multiple Teams.

The Pieces



REFERENCES

- DevOps: Database Change Management with schemachange and GitHub (snowflake.com)
- DevOps: Database Change Management with schemachange and Azure DevOps (snowflake.com)
- Building Snowflake CI/CD Pipelines with Azure DevOps | Medium
- Embracing Agile Software Delivery and DevOps with Snowflake - Snowflake Blog
- Don’t Do Analytics Engineering in Snowflake Until You Read This (Hint: dbt) | by Hashmap | HashmapInc | Medium

THANK YOU