Altinity

Testing Support for Multiple Authentication Methods in ClickHouse® Using Combinatorics and Behavioral Models

About me

QA Engineer for ClickHouse®, Altinity (since 2023)

M.Sc. in Data Science, Ludwig Maximilian University of Munich (2024-2026)
B.Sc. in Applied Mathematics and Computer Science, Moscow State University (2019-2023)

LinkedIn: Alsu Giliazova
https://www.linkedin.com/in/alsugiliazova/

About Altinity

Provides managed services and support for ClickHouse®, develops and maintains the Kubernetes operator for ClickHouse®, and runs other open-source projects such as:

- Altinity Stable Builds for ClickHouse®
- Altinity Backup for ClickHouse®
- Altinity Grafana Plugin for ClickHouse®
- Altinity Regression Test Suite for ClickHouse®

Join our Slack community!

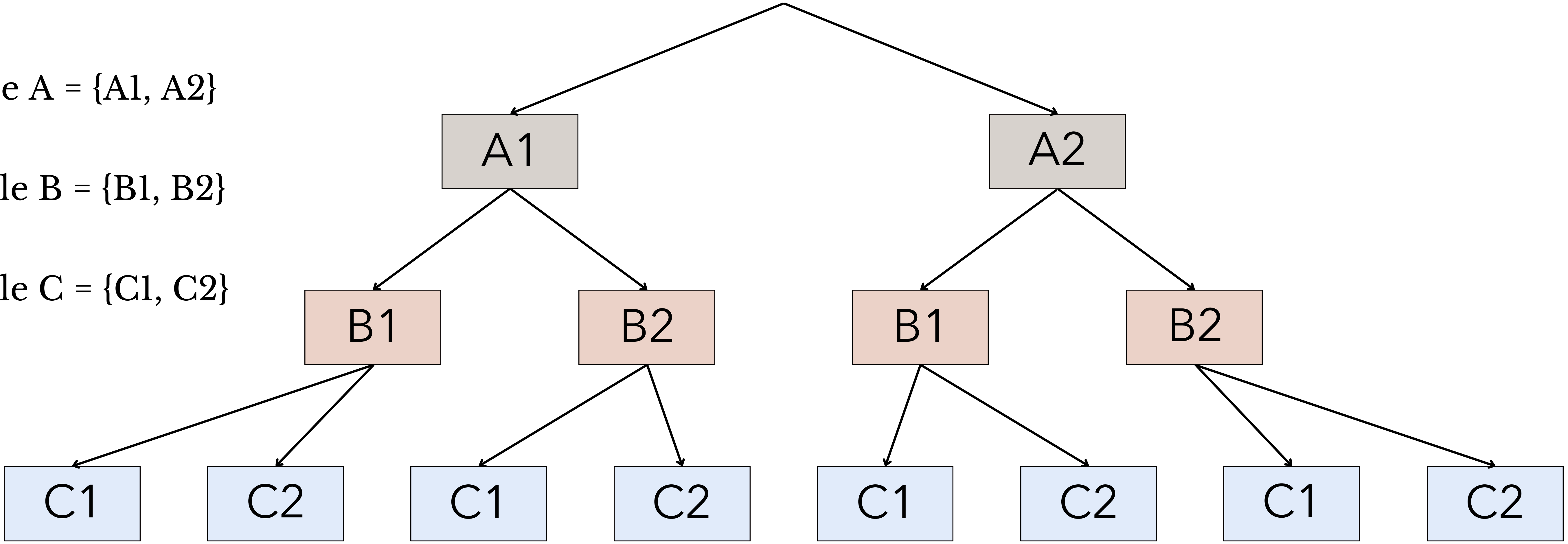
altinity.com/slack

What is combinatorial testing?

A Variable A = {A1, A2}

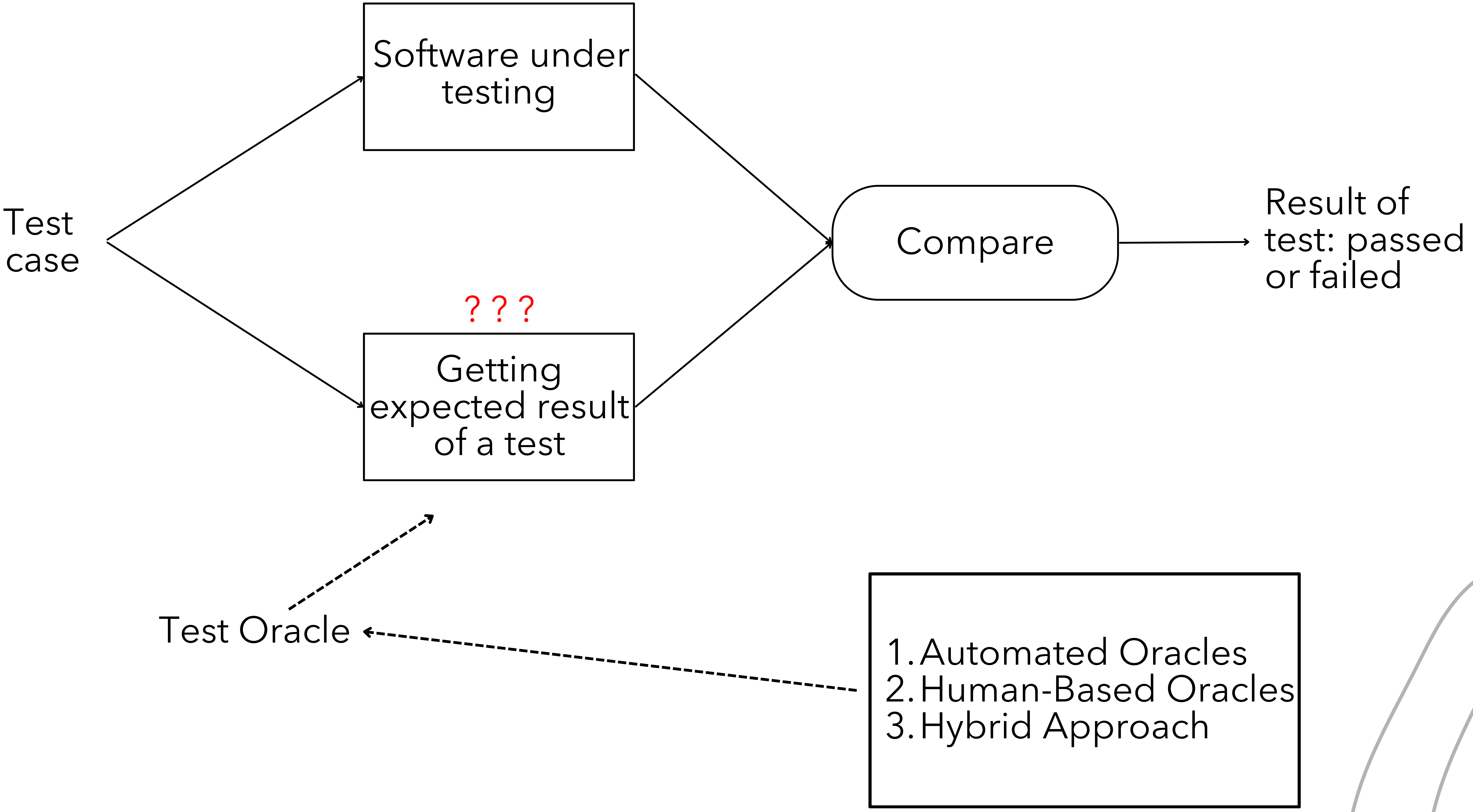
B Variable B = {B1, B2}

C Variable C = {C1, C2}



Test cases: {(A1,B1,C1), (A1,B1,C2), (A1,B2,C1), (A1,B2,C2), ...}

Test Oracle Problem



Test case

Software under testing

? ? ?

Getting expected result of a test

Compare

Result of test: passed or failed

Test Oracle

1. Automated Oracles
2. Human-Based Oracles
3. Hybrid Approach

ClickHouse®

- An open-source, columnar database designed for real-time analytics
- Known for its blazing-fast performance
- Handles large volumes of data efficiently, perfect for real-time use cases
- Highly scalable, suitable for both small projects and enterprise systems
- Widely used across industries for user behavior tracking, financial analytics, and monitoring systems
- Offers flexibility and control with open-source

The Feature: **Multiple Authentication Methods**

- A recent addition to ClickHouse® by Altinity for better security and flexibility
- Allows a user to have multiple authentication methods, either of the same type or of different types

Multiple Authentication Methods Feature

CREATE USER statement

Before: CREATE USER name1 IDENTIFIED WITH plaintext_password BY 'my_password'

Now: CREATE USER name2 IDENTIFIED WITH plaintext_password BY '1', bcrypt_password BY '2', plaintext_password BY '3'

Before:

name	String
id	UUID
auth_type	**Enum8**
auth_params	**String** (JSON format)
...	...

After:

name	String
id	UUID
auth_type	**Array(Enum8)**
auth_params	**Array(String)**
...	...

Multiple Authentication Methods Feature

ALTER USER statement

1. ALTER USER IDENTIFIED WITH statement

Before: `ALTER USER name1 IDENTIFIED WITH plaintext_password BY 'another_password'`

Now: `ALTER USER name2 IDENTIFIED WITH plaintext_password BY '4', bcrypt_password BY '5'`

2. ALTER USER **ADD** IDENTIFIED WITH statement

```
ALTER USER name2 ADD IDENTIFIED WITH plaintext_password BY '6', bcrypt_password BY '7'
```

3. ALTER USER **RESET AUTHENTICATION METHODS TO NEW** statement

```
ALTER USER name2 RESET AUTHENTICATION METHODS TO NEW
```

VALID UNTIL clause

```
ALTER USER name1 IDENTIFIED WITH plaintext_password BY 'some_password' VALID UNTIL '2026-01-01'
```

```
ALTER USER name2 IDENTIFIED WITH plaintext_password BY '1' VALID UNTIL '2026-01-01',
bcrypt_password BY '7' VALID UNTIL '2029-01-01'
```

```
ALTER USER name2 VALID UNTIL '2027-01-01'
```

Example

```
CREATE USER Bob IDENTIFIED WITH plaintext_password BY
'1', bcrypt_password BY '2', plaintext_password BY '3'
```

Created user bob with three authentication methods, Bob can login to clickhouse server with passwords **‘1’, ‘2’** and **‘3’**

```
ALTER USER Bob IDENTIFIED WITH plaintext_password BY
'4', bcrypt_password BY '5'
```

Changed Bob’s authentication methods, now Bob can only login with passwords **‘4’** and **‘5’**

```
ALTER USER Bob ADD IDENTIFIED WITH
plaintext_password BY '6', bcrypt_password BY '7'
```

Added two new authentication methods to Bob, now he can login with passwords **‘4’, ‘5’, ‘6’** and **‘7’**

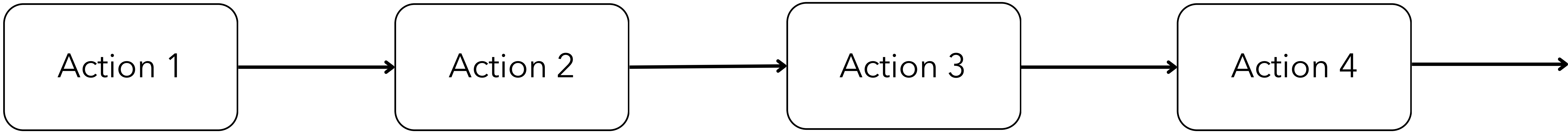
```
ALTER USER Bob RESET AUTHENTICATION METHODS TO
NEW
```

Reset Bob’s authentication methods to the most recently added one, Bob can login only with password **‘7’**

Defining user actions

What are the possible **actions** that user can perform with the feature?

1. Create user with multiple authentication methods
2. Change user’s authentication methods
3. Add new authentication methods to user
4. Reset user’s authentication methods to the most recently added method
5. Drop user



Validation should be performed after each action to ensure the correctness of the entire sequence of operations!

Calculating the Number of Combinations

Assumptions for the sake of simplicity in this explanation:

1. A user can have no more than two authentication methods assigned or changed per action
2. The two authentication methods can only be selected from the following 5 types:
   - no_password
   - plaintext_password BY ‘some_password’
   - sha256_hash BY 'hash' SALT 'salt'
   - bcrypt_hash BY 'hash'
   - double_sha1_hash BY 'hash'

CREATE USER BOB IDENTIFIED WITH ______ 5 ways

CREATE USER BOB IDENTIFIED WITH ______ , ______ C(n,r)=n!/(r!*(n−r)!) = 5!/(3!*2!) = 10 ways

There are 15 ways to create a user, with each user having no more than 2 auth methods, and each auth method being selected from the 5 available types.

Note: 5! = 1*2*3*4*5; 3!=1*2*3; 2!=1*2

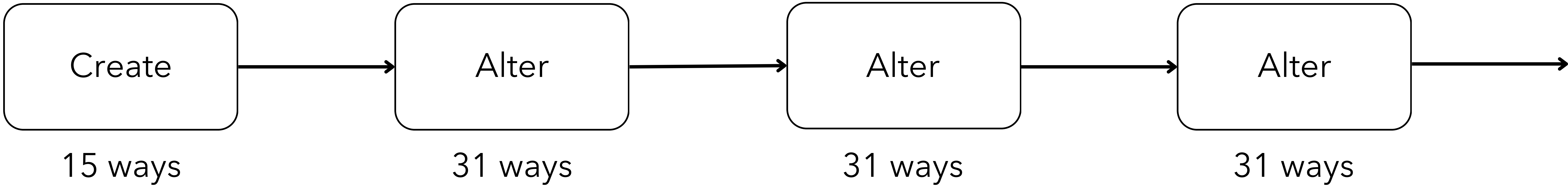
Calculating the Number of Combinations

ALTER USER BOB IDENTIFIED WITH ______ ALTER USER BOB IDENTIFIED WITH ______ , ______	15 ways
ALTER USER BOB ADD IDENTIFIED WITH ______ ALTER USER BOB ADD IDENTIFIED WITH ______ , ______	15 ways
ALTER USER BOB RESET AUTHENTICATION METHODS TO NEW	1 way

So we have 31 different ways of changing user's authentication methods with ALTER USER statement.

Determining the Minimum Number of Calls

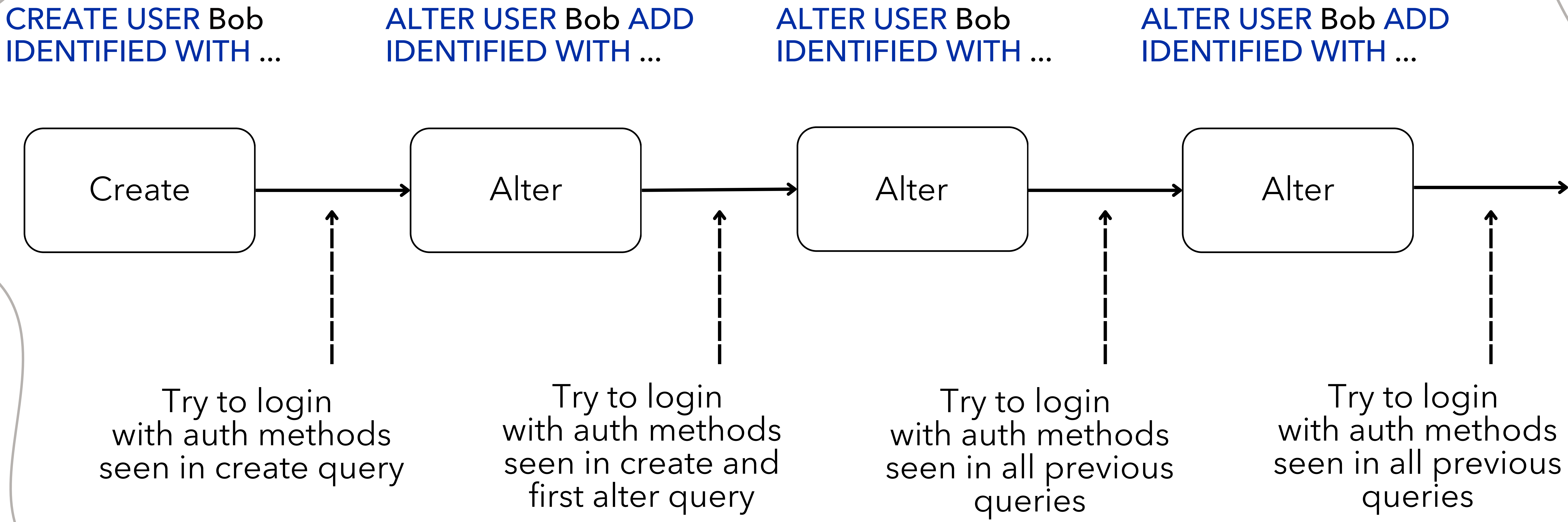
1. User was created
2. Authentication methods were changed
3. New authentication methods were added
4. Authentication methods were reset



The total number of combinations: 15 * 31 *31 *31 = **446865**

Efficient coverage without unnecessary complexity!

Sketching a Combinatorial Test



CREATE USER Bob IDENTIFIED WITH …

Create

Try to login with auth methods seen in create query

ALTER USER Bob ADD IDENTIFIED WITH …

Alter

Try to login with auth methods seen in create and first alter query

ALTER USER Bob IDENTIFIED WITH …

Alter

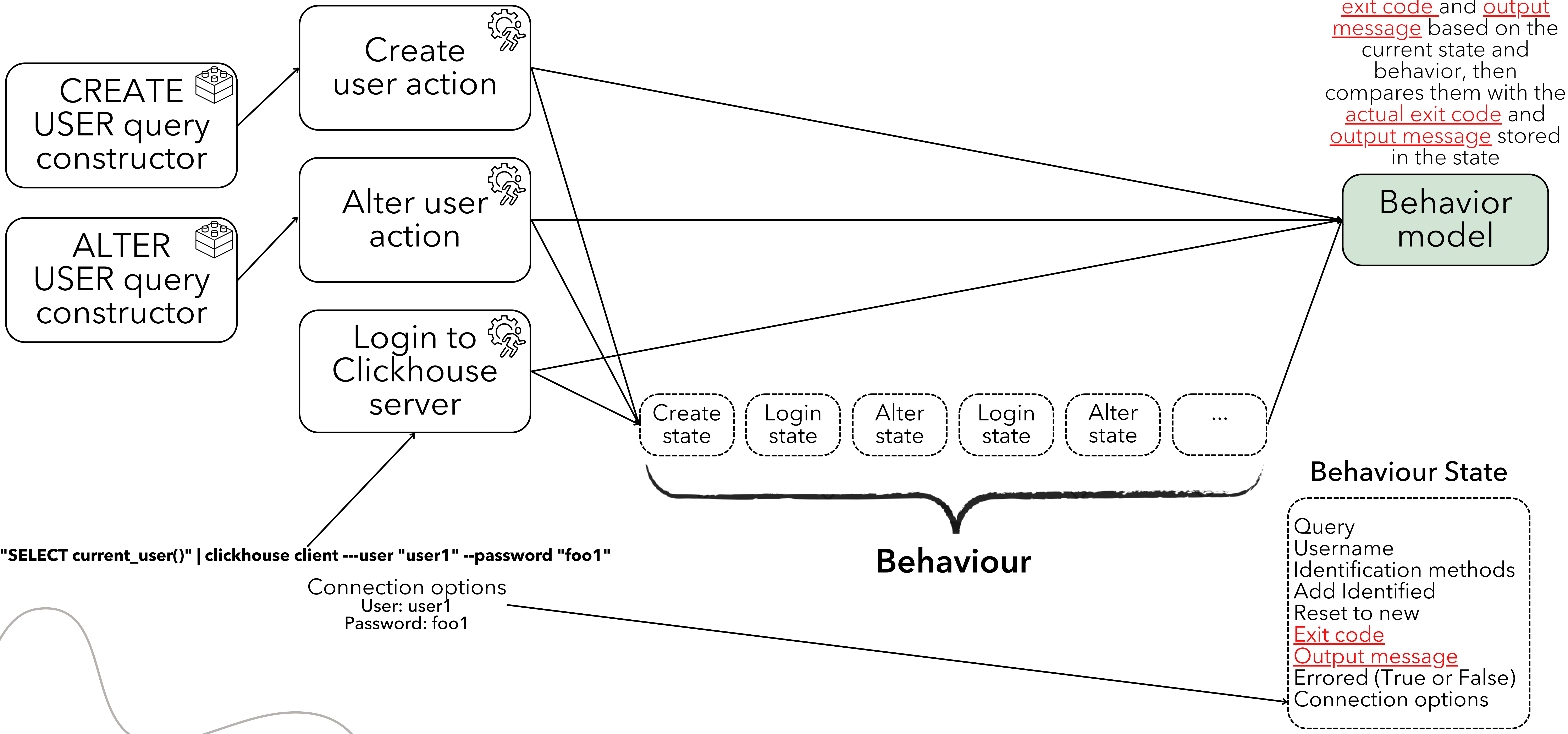
Try to login with auth methods seen in all previous queries

ALTER USER Bob ADD IDENTIFIED WITH …

Alter

Try to login with auth methods seen in all previous queries

Sketching a Combinatorial Test



CREATE USER query constructor

ALTER USER query constructor

Create user action

Alter user action

Login to Clickhouse server

Create state

Login state

Alter state

Login state

Alter state

...

Behaviour

Behavior model

Computes the expected exit code and output message based on the current state and behavior, then compares them with the actual exit code and output message stored in the state

```
"SELECT current_user()" | clickhouse client ---user "user1" --password "foo1"
```

Connection options
User: user1
Password: foo1

Behaviour State

Query
Username
Identification methods
Add Identified
Reset to new
Exit code
Output message
Errored (True or False)
Connection options

testflows.com

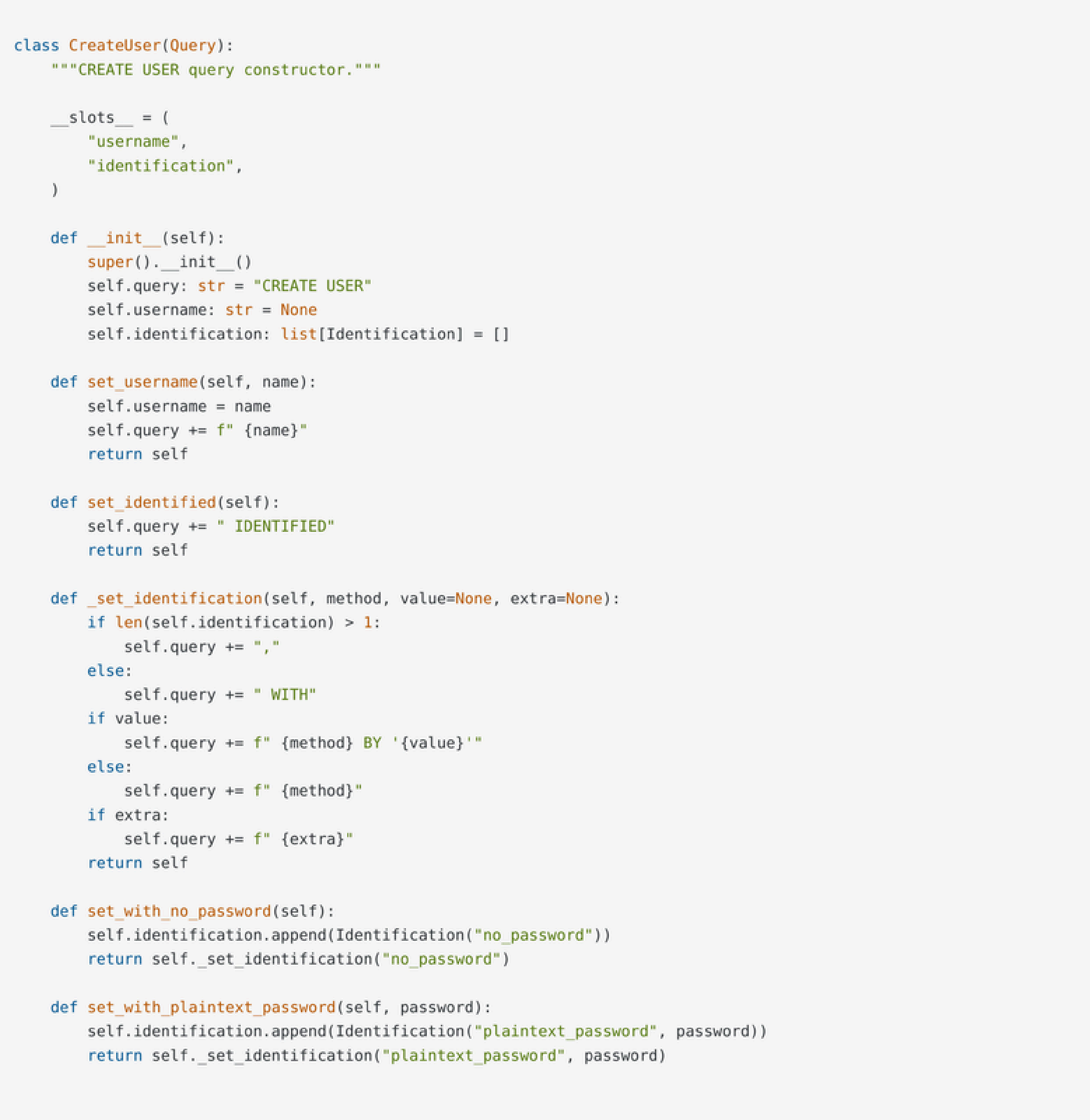
open-source software testing framework

Python open-source testing framework that allows you to write test programs, not just tests. Supports advanced behavioral, parallel, combinatorial, and requirements-driven testing. Used in testing ClickHouse®, Altinity.Cloud web UI, Graphana plugin, API services, Terraform provider, and more.

TestFlows website

https://testflows.com/

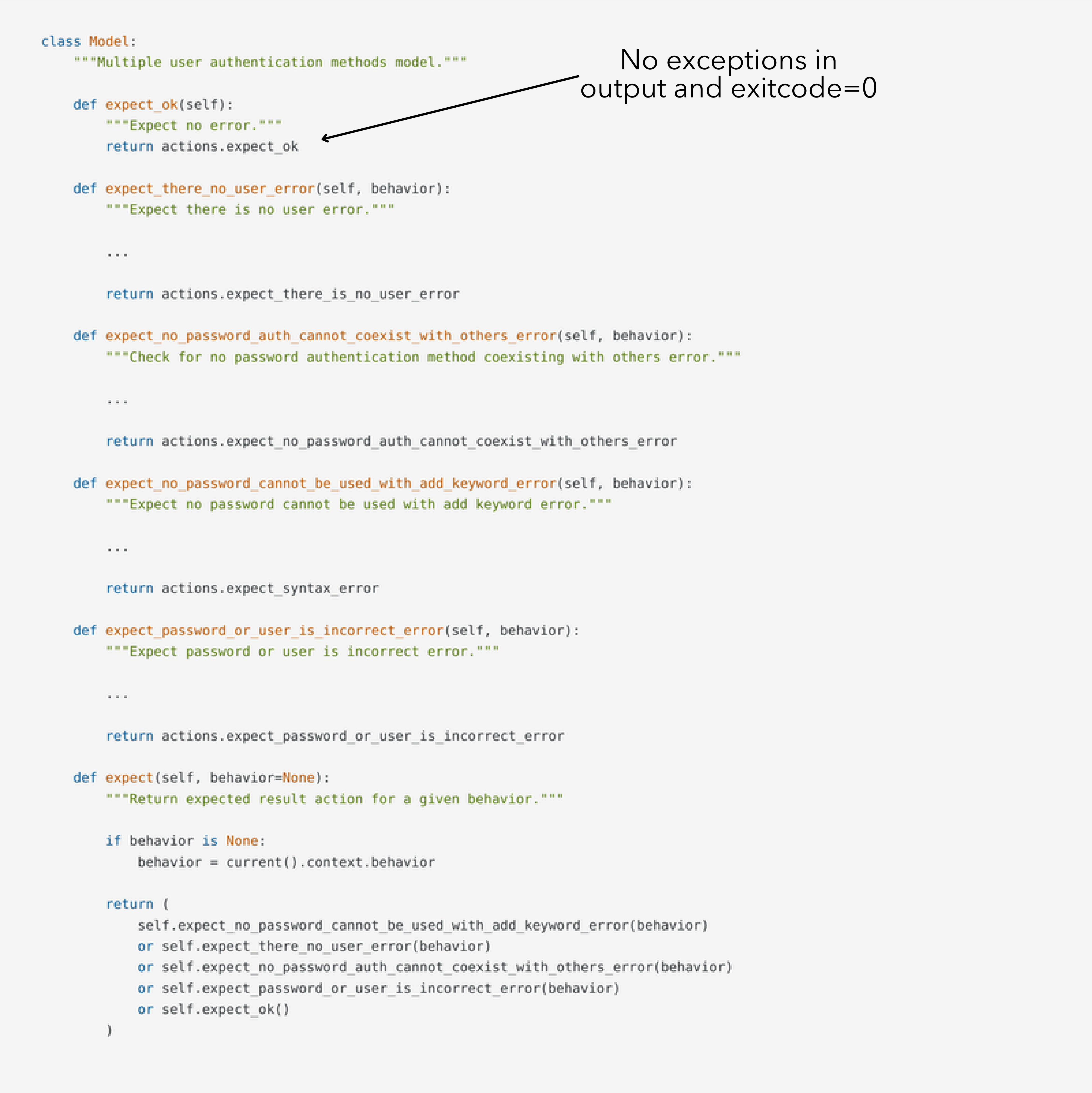
CREATE USER query constructor



```python
class CreateUser(Query):
    """CREATE USER query constructor."""

    __slots__ = (
        "username",
        "identification",
    )

    def __init__(self):
        super().__init__()
        self.query: str = "CREATE USER"
        self.username: str = None
        self.identification: list[Identification] = []

    def set_username(self, name):
        self.username = name
        self.query += f" {name}"
        return self

    def set_identified(self):
        self.query += " IDENTIFIED"
        return self

    def _set_identification(self, method, value=None, extra=None):
        if len(self.identification) > 1:
            self.query += ","
        else:
            self.query += " WITH"
        if value:
            self.query += f" {method} BY '{value}'"
        else:
            self.query += f" {method}"
        if extra:
            self.query += f" {extra}"
        return self

    def set_with_no_password(self):
        self.identification.append(Identification("no_password"))
        return self._set_identification("no_password")

    def set_with_plaintext_password(self, password):
        self.identification.append(Identification("plaintext_password", password))
        return self._set_identification("plaintext_password", password)
```

	self.query
1. Creating an instance of CreateUser class query = CreateUser()	CREATE USER
2. Call set_username method query.set_username(“Bob”)	CREATE USER Bob
3. Call set_identified method query.set_username(“Bob”)	CREATE USER Bob IDENTIFIED
4. Call set_with_no_password method query.set_username(“Bob”)	CREATE USER Bob IDENTIFIED WITH no_password

Model Definition

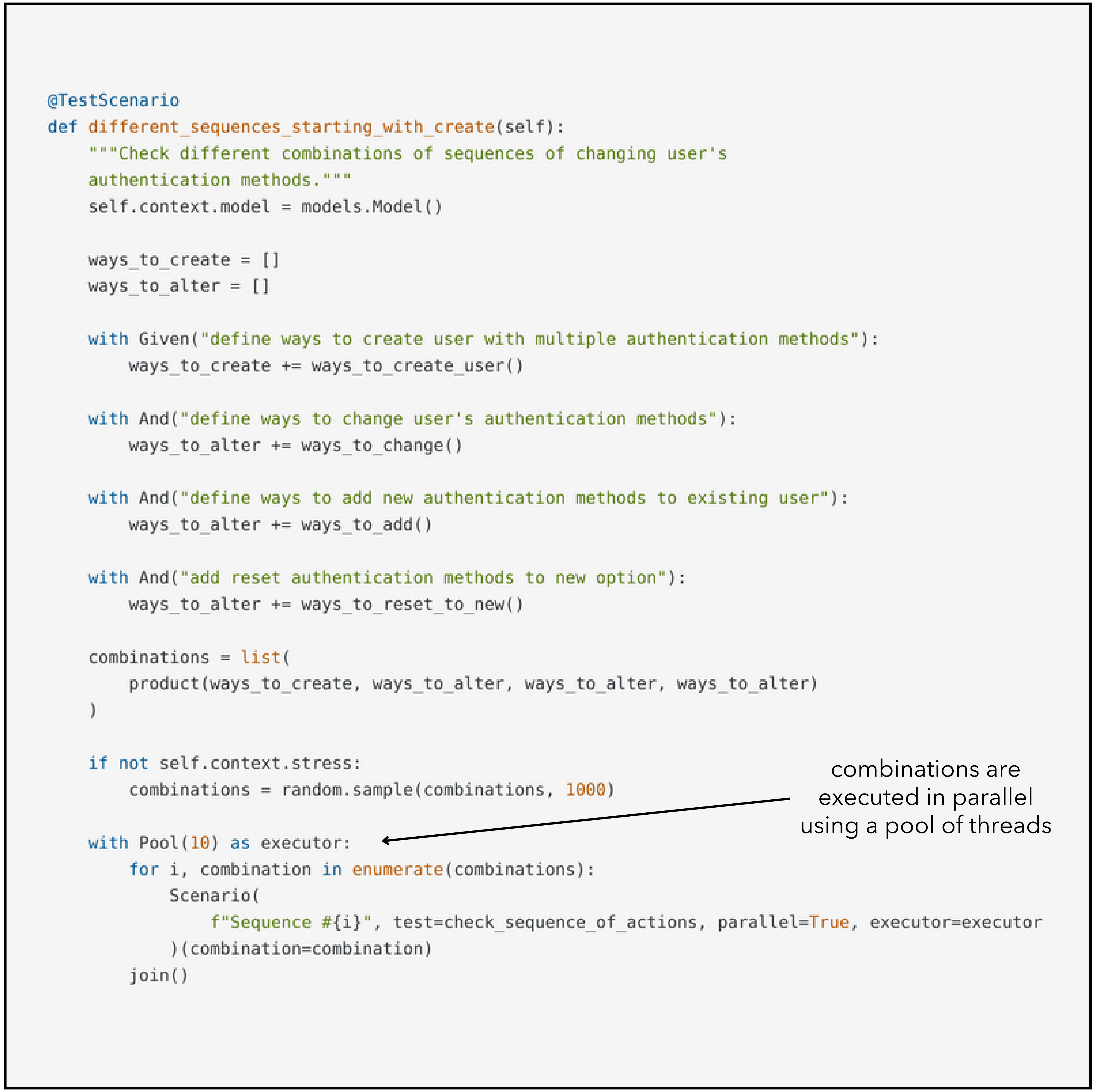


```
class Model:
    """Multiple user authentication methods model."""

    def expect_ok(self):
        """Expect no error."""
        return actions.expect_ok

    def expect_there_no_user_error(self, behavior):
        """Expect there is no user error."""

        ...

        return actions.expect_there_is_no_user_error

    def expect_no_password_auth_cannot_coexist_with_others_error(self, behavior):
        """Check for no password authentication method coexisting with others error."""

        ...

        return actions.expect_no_password_auth_cannot_coexist_with_others_error

    def expect_no_password_cannot_be_used_with_add_keyword_error(self, behavior):
        """Expect no password cannot be used with add keyword error."""

        ...

        return actions.expect_syntax_error

    def expect_password_or_user_is_incorrect_error(self, behavior):
        """Expect password or user is incorrect error."""

        ...

        return actions.expect_password_or_user_is_incorrect_error

    def expect(self, behavior=None):
        """Return expected result action for a given behavior."""

        if behavior is None:
            behavior = current().context.behavior

        return (
            self.expect_no_password_cannot_be_used_with_add_keyword_error(behavior)
            or self.expect_there_no_user_error(behavior)
            or self.expect_no_password_auth_cannot_coexist_with_others_error(behavior)
            or self.expect_password_or_user_is_incorrect_error(behavior)
            or self.expect_ok()
        )
```

No exceptions in
output and exitcode=0

All possible expected outputs:

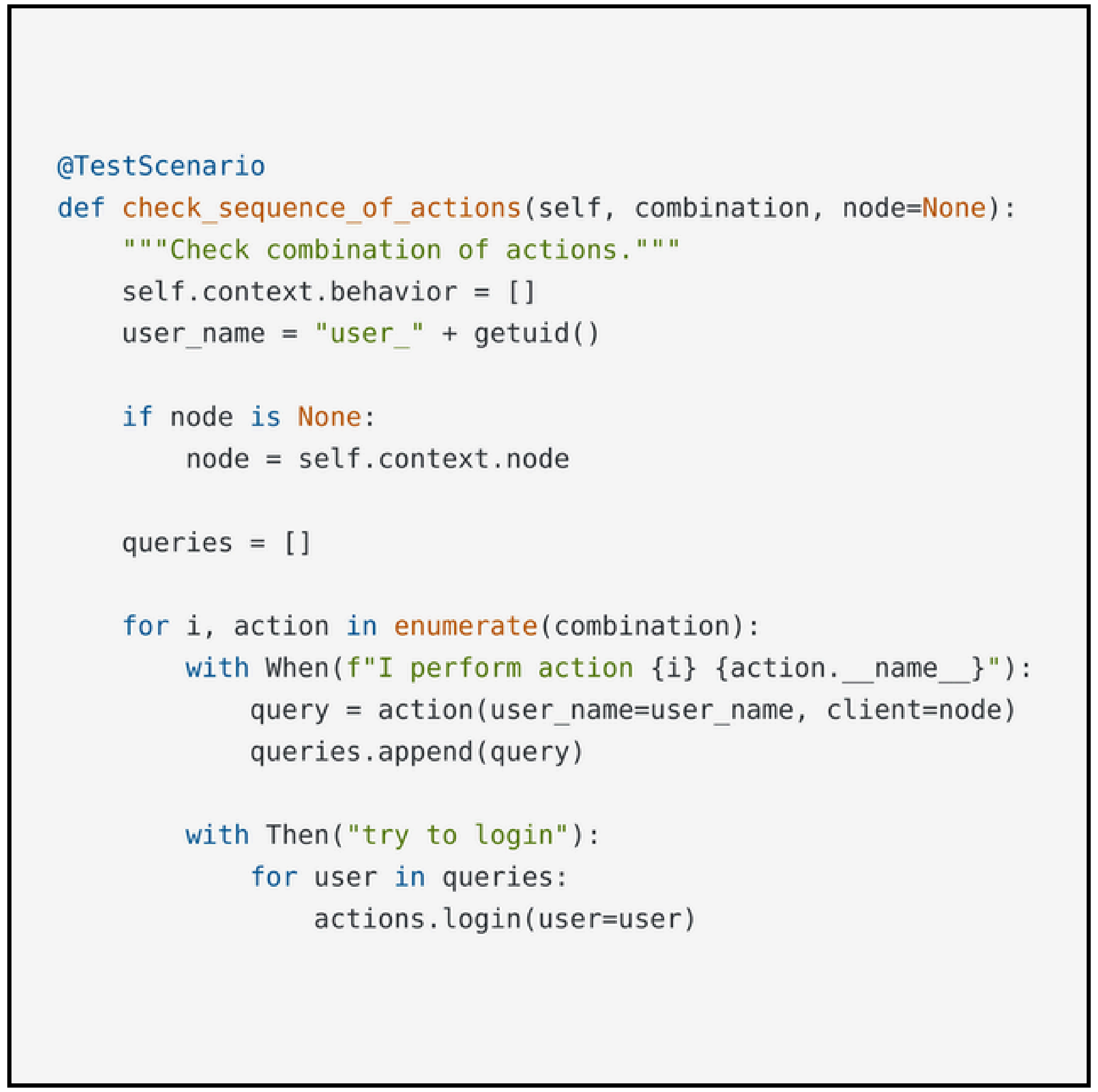
1. "NO_PASSWORD" cannot be used in an ADD IDENTIFIED statement
2. A non-existing user cannot be altered
3. "NO_PASSWORD" cannot be used with another authentication method
4. The wrong password was used to log in with the specified username
5. No exceptions; the query is valid, and the exit code is 0



```
@TestScenario
def different_sequences_starting_with_create(self):
    """Check different combinations of sequences of changing user's
    authentication methods."""
    self.context.model = models.Model()

    ways_to_create = []
    ways_to_alter = []

    with Given("define ways to create user with multiple authentication methods"):
        ways_to_create += ways_to_create_user()

    with And("define ways to change user's authentication methods"):
        ways_to_alter += ways_to_change()

    with And("define ways to add new authentication methods to existing user"):
        ways_to_alter += ways_to_add()

    with And("add reset authentication methods to new option"):
        ways_to_alter += ways_to_reset_to_new()

    combinations = list(
        product(ways_to_create, ways_to_alter, ways_to_alter, ways_to_alter)
    )

    if not self.context.stress:
        combinations = random.sample(combinations, 1000)

    with Pool(10) as executor:
        for i, combination in enumerate(combinations):
            Scenario(
                f"Sequence #{i}", test=check_sequence_of_actions, parallel=True, executor=executor
            )(combination=combination)
        join()
```

combinations are executed in parallel using a pool of threads

Test Definition



```
@TestScenario
def check_sequence_of_actions(self, combination, node=None):
    """Check combination of actions."""
    self.context.behavior = []
    user_name = "user_" + getuid()

    if node is None:
        node = self.context.node

    queries = []

    for i, action in enumerate(combination):
        with When(f"I perform action {i} {action.__name__}"):
            query = action(user_name=user_name, client=node)
            queries.append(query)

        with Then("try to login"):
            for user in queries:
                actions.login(user=user)
```

Behaviour State

Query
Username
Identification methods
Add Identified
Reset to new
Exit code
Output message
Errored (True or False)
Connection options

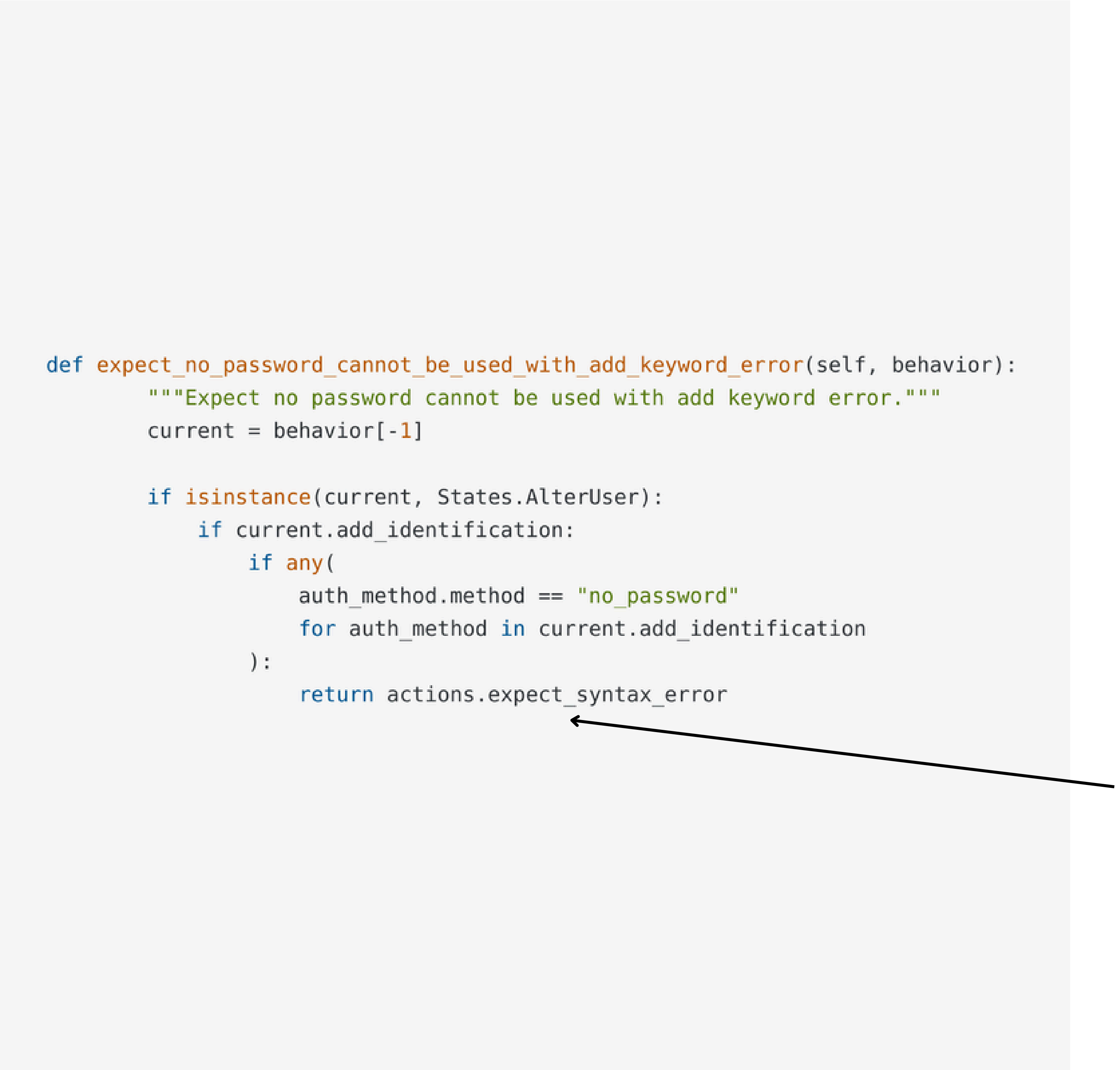
Expect Methods of the Model

ALTER USER Bob ADD IDENTIFIED WITH plaintext_password BY '4', bcrypt_password BY '5', NO_PASSWORD; 👎

Here: current.add_identification = [”plaintext_password”, “bcrypt_password”, “no_password”]

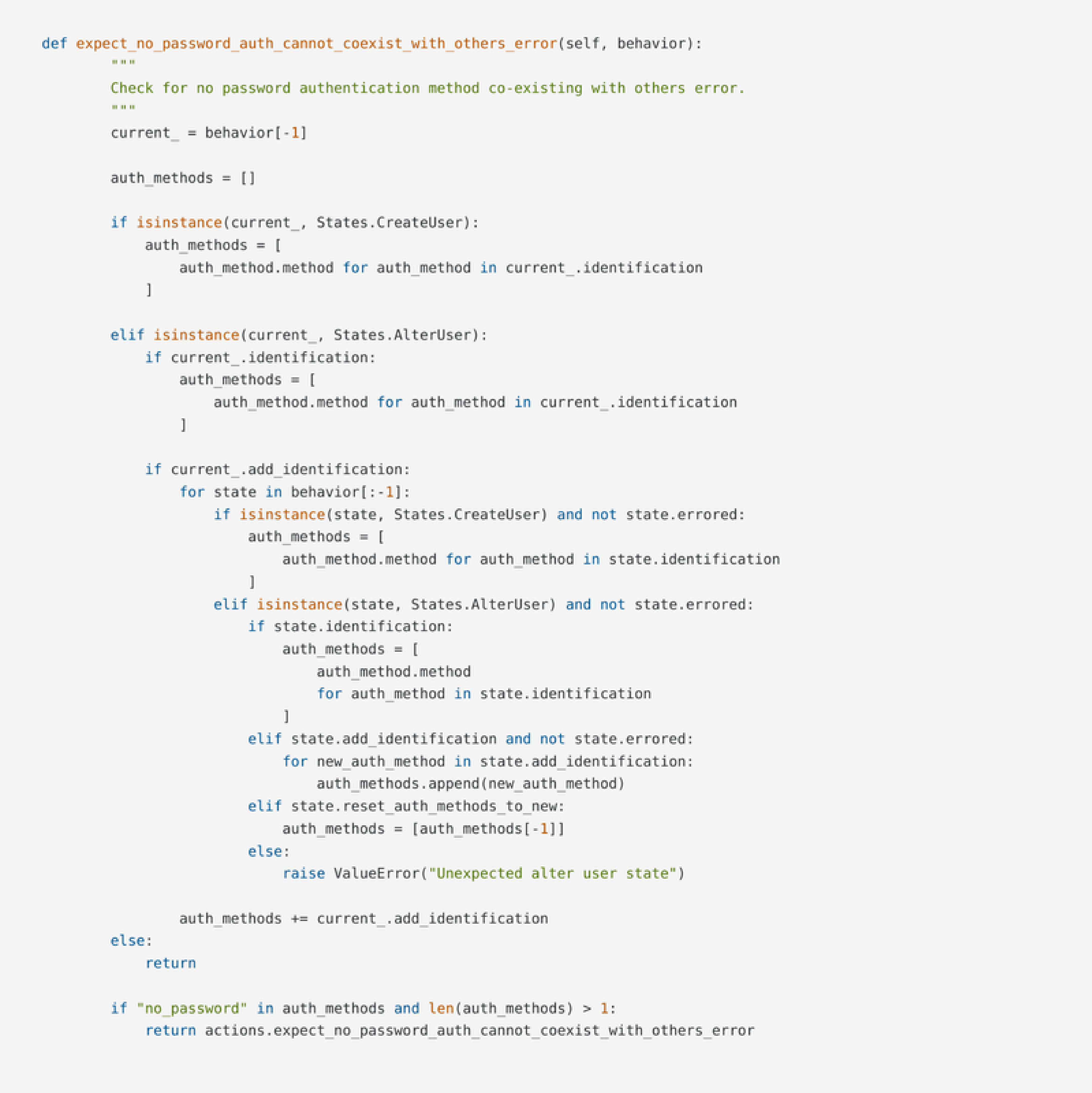
ALTER USER Bob ADD IDENTIFIED WITH plaintext_password BY '1', bcrypt_password BY '2'; 👍

Here: current.add_identification = [”plaintext_password”, “bcrypt_password”]

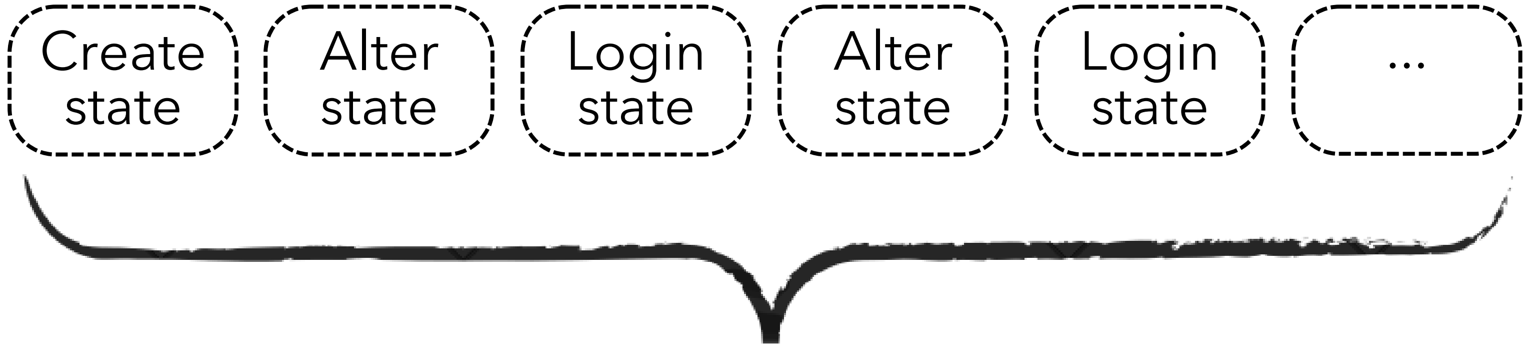


```
def expect_no_password_cannot_be_used_with_add_keyword_error(self, behavior):
        """Expect no password cannot be used with add keyword error."""
        current = behavior[-1]

        if isinstance(current, States.AlterUser):
            if current.add_identification:
                if any(
                    auth_method.method == "no_password"
                    for auth_method in current.add_identification
                ):
                    return actions.expect_syntax_error
```

Code: 62. DB::Exception: Syntax error: failed at position 87 ('NO_PASSWORD'): NO_PASSWORD;. Expected one of: PLAINTEXT_PASSWORD, SHA256_PASSWORD, DOUBLE_SHA1_PASSWORD, LDAP, KERBEROS, SSL_CERTIFICATE, BCRYPT_PASSWORD, SSH_KEY, HTTP, JWT, SHA256_HASH, DOUBLE_SHA1_HASH, BCRYPT_HASH, BY, end of query. (SYNTAX_ERROR)



```python
def expect_no_password_auth_cannot_coexist_with_others_error(self, behavior):
        """
        Check for no password authentication method co-existing with others error.
        """
        current_ = behavior[-1]

        auth_methods = []

        if isinstance(current_, States.CreateUser):
            auth_methods = [
                auth_method.method for auth_method in current_.identification
            ]

        elif isinstance(current_, States.AlterUser):
            if current_.identification:
                auth_methods = [
                    auth_method.method for auth_method in current_.identification
                ]

            if current_.add_identification:
                for state in behavior[:-1]:
                    if isinstance(state, States.CreateUser) and not state.errored:
                        auth_methods = [
                            auth_method.method for auth_method in state.identification
                        ]
                    elif isinstance(state, States.AlterUser) and not state.errored:
                        if state.identification:
                            auth_methods = [
                                auth_method.method
                                for auth_method in state.identification
                            ]
                        elif state.add_identification and not state.errored:
                            for new_auth_method in state.add_identification:
                                auth_methods.append(new_auth_method)
                        elif state.reset_auth_methods_to_new:
                            auth_methods = [auth_methods[-1]]
                        else:
                            raise ValueError("Unexpected alter user state")

                auth_methods += current_.add_identification
        else:
            return

        if "no_password" in auth_methods and len(auth_methods) > 1:
            return actions.expect_no_password_auth_cannot_coexist_with_others_error
```



Create state

Alter state

Login state

Alter state

Login state

...

Behaviour

Behaviour State

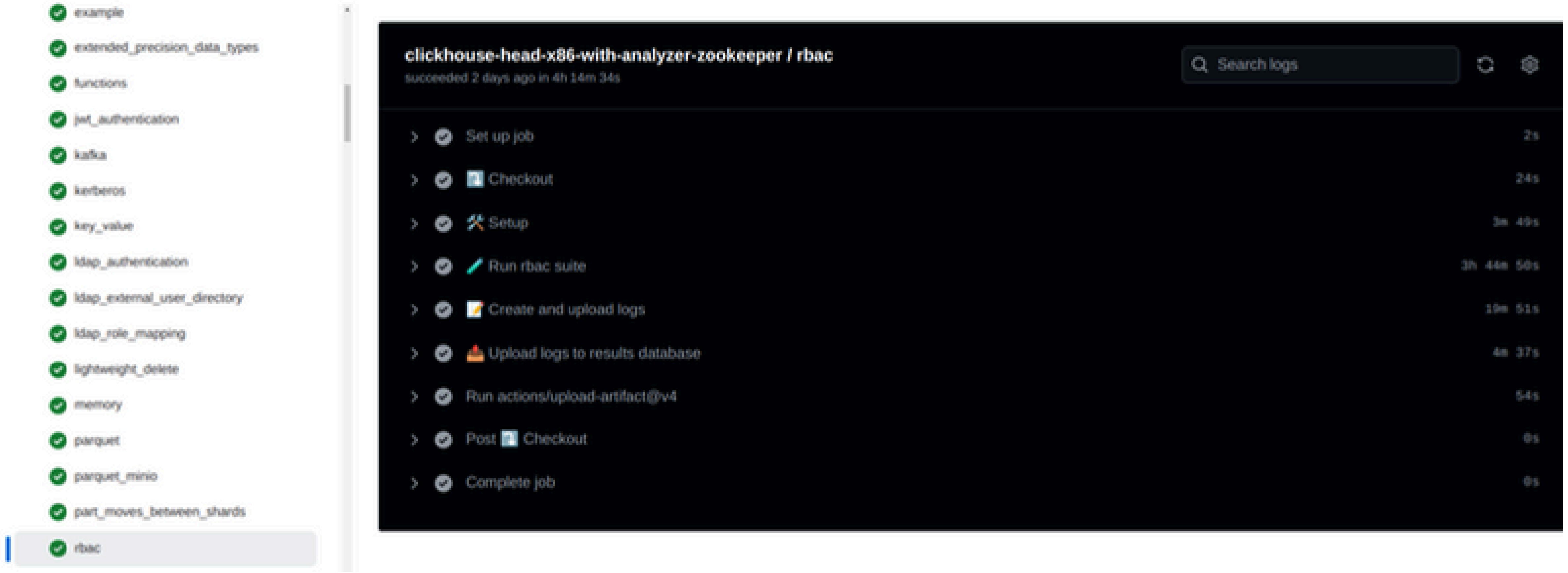
Query
Username
Identification methods
Add Identified
Reset to new
Exit code
Output message
Errored (True or False)
Connection options

CREATE USER Bob IDENTIFIED WITH plaintext_password BY '1', NO_PASSWORD

ALTER USER Bob IDENTIFIED WITH plaintext_password BY '4', bcrypt_password BY '5', NO_PASSWORD;

Issues Found by Combinatorics

- The VALID UNTIL clause didn't work correctly with bcrypt_password; I was able to log in to ClickHouse with an expired password
- Using NOT IDENTIFIED with the VALID UNTIL clause in a single query threw an unexpected exception message, which should not have happened
- In some configuration, I could not log in with a valid password using the sha256_hash authentication method
- There were issues logging in on one cluster when the user was created with the ON CLUSTER clause on another cluster



✓ example
✓ extended_precision_data_types
✓ functions
✓ jwt_authentication
✓ kafka
✓ kerberos
✓ key_value
✓ ldap_authentication
✓ ldap_external_user_directory
✓ ldap_role_mapping
✓ lightweight_delete
✓ memory
✓ parquet
✓ parquet_minio
✓ part_moves_between_shards
✓ rbac

clickhouse-head-x86-with-analyzer-zookeeper / rbac
succeeded 2 days ago in 4h 14m 34s

```
> ✓ Set up job                              2s
> ✓ Checkout                               24s
> ✓ Setup                              3m 49s
> ✓ Run rbac suite                 3h 44m 50s
> ✓ Create and upload logs            19m 51s
> ✓ Upload logs to results database    4m 37s
> ✓ Run actions/upload-artifact@v4        54s
> ✓ Post Checkout                          0s
> ✓ Complete job                           0s
```

The full test code is available here:

Thank you for your time! 😊
I’m happy to answer any questions!

Join our Slack community!

altinity.com/slack

LinkedIn: Alsu Giliazova
https://www.linkedin.com/in/alsugiliazova/