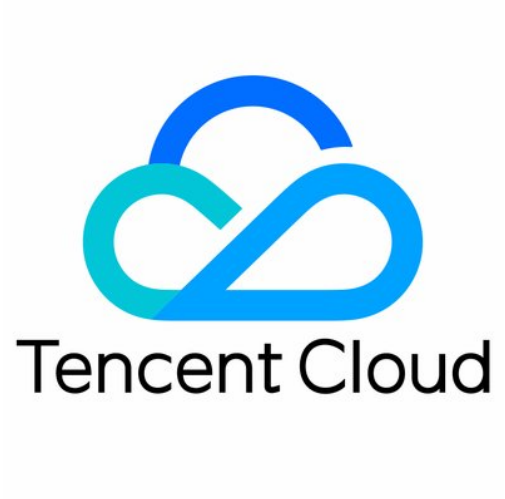
Data Transfer Service

FAQs

Product Documentation



Contents

FAQs
Data Migration

Last updated：2024-07-08 15:37:11

General

How does data migration/sync with DTS affect the source database?

The data in the source database will not be affected and can be normally read/written. Data migration with DTS essentially replicates the data from the source database to the target database, without deleting or affecting such data. The performance (**especially the CPU**) of the source database will be slightly affected. When DTS performs full data migration/sync, it will read the full data of the source database, which may increase the pressure of the source database.
If the source database is MySQL (8-core and 16 GB MEM), the DTS task uses eight concurrent threads by default, and there is no network bottleneck, then the task will impact the source database performances as follows:
Full export stage: DTS occupies about 18%–45% of the source database CPU, increases the query pressure on the source database by about 40-60 MB/s, and occupies about 8 active session connections.
Incremental export stage: There is almost no pressure on the source database, and only one connection is used to listen on the source database binlogs in real time.
In the DTS full export stage, the connection details of the source database are as follows:
1. In the initial stage of a task, one thread queries the system table to obtain the export information with the following SQL statements:

```
SELECT TABLE_NAME, TABLE_TYPE, ENGINE, TABLE_ROWS FROM INFORMATION_SCHEMA.TABLES WH

SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS WHERE ABLE_SCHEMA='db'
```

2. In the structural export stage, a connection with eight (by default) export threads executes the following SQL statements:

```
SELECT COLUMN_NAME, DATA_TYPE, COLUMN_TYPE, NUMERIC_PRECISION, NUMERIC_SCALE, CHARA
   "where TABLE_SCHEMA='db' and TABLE_NAME='db';
```

3. In the data export stage, a connection with eight (by default) export threads executes the following SQL statements:

```
SELECT /*!40001 SQL_NO_CACHE */ column FROM db.table where id>= 'id' AND id <
'id';
```

4. For data export without locks, a table lock SQL statement like the following will be used to obtain the consistency offset of tables without a primary key. Once the offset is obtained, the tables will be unlocked.

```
lock table xxx read
```

5. For data export with locks, a global lock will be added with a SQL statement like the following.

```
flush table xxx with read lock
```

How does data migration with DTS affect the target database?

When DTS writes full data to the target database, **it mainly impacts the CPU and IOPS of the target database**. If the source database is MySQL (8-core and 16 GB MEM), the DTS task uses eight concurrent threads by default, and there is no network bottleneck, then the task will impact the target database performances in the full import stage as follows: DTS occupies about 20%–49% of the target database CPU, 1,200-3,100 IOPS, and less than 8 active session connections.
In the full import stage, the connection details of the target database are as follows:
Less than eight connections are creating structures in batches.
Less than eight connections are writing data in batches with a SQL statement like the following:

```
insert into xxx (id,name,msg) values (xxx);
```

In the incremental import stage, DTS parses the incremental data in the source database binlogs into SQL statements and then executes them in the target database. The total number of connections/sessions is up to 32.
DDL statements are executed sequentially, with no parallel execution of other DML statements.
A DML statement supports up to 32 non-persistent connections that time out in 30 seconds. DML statements include INSERT, UPDATE, DELETE, and REPLACE.

Does the target database need to be empty when I use DTS to migrate data?

It depends on whether you choose to migrate the entire instance or specified objects.
Migrating the entire instance: The target database needs to be empty; otherwise, the system verification will fail, and the task cannot be initiated.
Migrating specified objects: You can specify objects such as databases and tables for migration. The system will verify whether the source and target databases have tables with the same name, and if so, the verification will fail, and you will be prompted to make changes first.

Does DTS allow a read-only source instance in data migration?

Yes. In scenarios where the source database is a self-built one, you can enter the IP address of the read-only instance when configuring the connection to the source database. In scenarios where the source database is a TencentDB instance, read-only instances are only supported for data subscription, and you need to submit a ticket for application.

Does DTS allow a replica or secondary source database in data migration?

In scenarios where the source database is a self-built one, you can enter the IP address of the replica or secondary database when configuring the connection to the source database. In scenarios where the source database is a TencentDB instance, you can only select the instance ID but cannot connect to the replica or secondary database when configuring the source database connection.

Does the source database support data writes during migration?

Yes. Data can be normally written to the source database during the migration process, but the source database doesn't support DDL operations in the structural and full migration stages; otherwise, the migration task may fail. DDL and DML operations can be normally performed in the incremental stage.

Does the target database support data writes during migration?

Although the target database allows data writing, it is advisable to refrain from doing so. Concurrent writes to the target database during the migration process can potentially lead to data inconsistency between the source and target databases. It is not necessary to configure the target database as read-only.

Does DTS support migration from one local database to another?

No. There must be at least one TencentDB database, either the source or target database.

Does DTS support data migration between TencentDB instances under two different Tencent Cloud accounts?

Yes. For migration between TencentDB instances under two different Tencent Cloud accounts, you need to log in to DTS with the Tencent Cloud account of the target instance. For detailed directions, see Cross-Account TencentDB Instance Migration.

Can DTS migrate different tables in the same database instance?

No. DTS only supports data migration between different source and target databases.

Can I configure multiple DTS tasks for migration from the same source database to different TencentDB instances?

Yes. You can migrate data from the same source database to multiple target databases (multiple tasks can run concurrently) and vice versa (a new task can only be initiated after the previous task enters the incremental migration stage). Note that multiple concurrent tasks may increase the access pressure on the source or target databases and thus slow down the migration. If you need to create multiple migration tasks for the same source database, then after creating the first task, you can quickly create similar tasks by clicking **More** > **Create similar task** in the **Operation** column.

Does DTS support scheduled automatic migration?

Yes. When modifying the configuration for a created data migration task, you can select scheduled migration and specify the start time.

Can I monitor the task progress during migration?

Yes. You can log in to the DTS console and view the migration task progress on the **Data Migration** page.

Why is there a 15-day limit on incremental migration?

Currently, incremental migration is performed through the nearest proxy server via Tencent Cloud Direct Connect, which eliminates network jitters and ensures the quality of data transfer. The 15-day limit can reduce the connection pressure on the proxy server, and it is only intended for reasonable utilization of resources for migration. Connections will not be forcibly closed after 15 days.

How is data accuracy ensured during data migration?

DTS uses Tencent Cloud's proprietary data migration architecture to verify data accuracy in real time and quickly detect and correct errors. This guarantees the reliability of the transferred data.

Why does data verification require that the source database instance not be read-only?

This is because data verification requires creating a new database `__tencentdb__` in the source instance and writing the checksum table to the database. If the instance is read-only, data verification will be skipped.

Can I specify tables for migration with DTS?

Yes. You can select an entire instance or specify databases/tables as the migration object.

When does data migration stop?

When you select incremental migration, if it takes a long time before the task stops, you may need to stop it by yourself.

If you select **Structural migration** or **Full migration** as the **Migration Type**, the task will automatically stop upon completion.

If you select **Full + incremental migration** as the **Migration Type**, after full migration is completed, the migration task will automatically enter the incremental data sync stage, which will not stop automatically. You need to click **Complete** to manually stop the incremental data sync.

Manually complete incremental data sync and business switchover at appropriate time.

Check whether the migration task is in the incremental sync stage without any lag. If so, stop writing data from the source database for a few minutes.

Manually complete incremental sync when the data gap between the target and the source databases is 0 MB and the time lag between them is 0 seconds.

Why does the data size change before and after full migration?

This is because the fragmented spaces of the source and target databases are different, and the source database may contain data holes. In this case, after full migration is completed, the table storage space in the target database may be smaller than that in the source database. We recommend that you perform a data consistency check as instructed in Creating Data Consistency Check Task after the migration is completed to check whether the contents of the source database and the target database are consistent.

Does DTS support cross-country/region database migration?

Yes. You can implement cross-country/region data transfer over the public network.

Can I resume a DTS migration task that is interrupted abnormally?

Yes. You can configure an automatic retry policy for the migration task. When the task is interrupted by exceptions (such as the source/target database downtime or network issues), DTS will automatically retry the task in the specified time range.

MySQL

Will tables be locked during MySQL migration?

Locking in the MySQL migration process refers to adding a global lock (FTWRL) to the source database. This will be involved only in case of full data migration.
Currently, migration without locks is implemented for migration links between MySQL, MariaDB, Percona, TDSQL-C for MySQL, TDSQL for MySQL, and TDSQL for TDStore. In this scenario, no global lock (the FTWRL lock) is added, and only tables without a primary key are locked.

Can I migrate tables without a primary key?

Yes. However, tables with a primary key are recommended. You can initiate a task to migrate tables without a primary key, but there may be some problems:
Migrating or syncing tables without a primary key may cause data duplication.
Performing DML operations on such tables may cause a data sync delay.
Tables without a primary will be skipped for data consistency check.
If the source database is Alibaba Cloud ApsaraDB for RDS or PolarDB, it will add an additional primary key column to tables without a primary key or non-null unique key in the binlog. The added primary key column is invisible in the table structure and thus may not be identified by DTS, and the data result may be abnormal.

Can I migrate TencentDB for MySQL single-node (formerly the Basic Edition) instances?

A TencentDB for MySQL single-node instance can be used as the source database for migration over the public network rather than the private network.
It cannot be used as the target database for migration.

How do I ensure that the `binlog_format` takes effect immediately after setting it to `row` for the source database?

After setting `binlog_format` to `row`, you need to reset all business connections to the current database. If the source database is a replica, you also need to reset the master-replica sync SQL thread to prevent current business connections from writing data in the old format.
Before the above operation is completed, do not create or start a migration task; otherwise, data inconsistency may occur.

What should I check if the TokuDB engine is used in the source instance?

In this case, TokuDB will be converted to InnoDB by default during migration. As tables containing clustered indexes or compressed with TokuDB need to be preprocessed before migration, they are not supported currently. DDL operations on TokuDB are not supported either.

Can I migrate user permissions during the migration of MySQL data?

Yes. NewDTS supports migrating user permissions. For detailed directions, see Account Migration.

Data Sync

Last updated：2024-07-08 15:38:21

What's the impact of data sync on the target database? Does the target database need to be empty?

Syncing data with DTS won't impact the target database, and the target database doesn't need to be empty. DTS will check whether there are objects with the same name in the source and target databases. If there is a duplicate name conflict, you can handle it by using the configured conflict resolution policy. You will be prompted of an error, and you can ignore it.

Will the sync service be affected if high availability (HA) switch occurs between the source and target instances?

If the source instance supports and enables global transaction identifier (GTID), when HA switch occurs in the source instance during incremental sync, the service will be automatically reconnected, and the synced data flow will resume soon after HA switch is completed.
If HA switch occurs in the target instance during incremental sync, the service will also be automatically reconnected, and the synced data flow will resume soon after HA switch is completed.

Can data in an instance on a later version be synced to an instance on an earlier version?

No. For data sync between instances of the same type, the major version of the target instance must be later than that of the source instance. Taking MySQL as an example, data cannot be synced from MySQL 5.7 to MySQL 5.6.

Can the source/target instance be a local instance?

Yes. Data can be synced from TencentDB to a local database over the public network and CCN.

Can I perform two-way sync for DDL statements in a two-way sync topology?

No. During sync instance creation, DDL statements in only one instance can be synced; otherwise, the algorithm will detect a DDL loop and then prohibit the creation of one of the instances.

Are non-transactional engines supported?

The current technical solution adds routing information to transactions to mark transaction sources, which relies on the atomicity of transactions. However, databases/tables based on non-transactional engines will corrupt such atomicity, and data consistency cannot be guaranteed. Therefore, we recommend that you not use non-transactional engines.

Can I add or delete sync objects after a sync task is started?

Yes. You can modify the sync task configurations by selecting the target sync task and clicking **Modify Sync Configuration** in the **Operation** column.

On the configuration modification page, you can add/delete sync objects and modify the primary key conflict policy or SQL sync policy. When you modify the sync configuration, the running sync task won't be paused or affected. For more information, see Modifying Sync Configuration.

FAQs for Data Subscription Kafka Edition

Last updated：2024-11-01 10:42:03

Why can't I consume data?

Check the network. The address of the Kafka server is a Tencent Cloud private network address, which can only be accessed in Tencent Cloud VPC that is in the same region of the subscribed instance.
Check whether the subscription topic, private network address, consumer group name, account or password is correct. You can click the subscription name on the Data Subscription console to go to the subscription details page and consumption management page to view such information.
Check whether the encryption parameter is correct. For more information, see What authentication mechanism does Kafka use.

What is the data format?

Data Subscription Kafka Edition uses Protobuf for data serialization. You can click here to download the Protobuf file. A demo project also contains the Protobuf file. For more information, see the “Key Demo Logic Description” section of Data Consumption Demo.

What authentication mechanism does Kafka use?

See the figure below:

```
def block_consume(brokers, topic, group, user, password, trans2sql):
    # create a kafka consumer
    consumer = kafka.KafkaConsumer(topic,
                                   auto_offset_reset='earliest',
                                   enable_auto_commit=False,
                                   group_id=group,
                                   sasl_plain_username=user,
                                   sasl_plain_password=password,
                                   sasl_mechanism='SCRAM-SHA-512',
                                   security_protocol='SASL_PLAINTEXT',
                                   bootstrap_servers=brokers)
```

When does Kafka commit?

Please first set the `enable_auto_commit` parameter of Kafka as `false` to disable auto commit. The producer inserts a checkpoint message at an appropriate position in the message sequence. After the checkpoint message is consumed, the Kafka client will commit feedback indicating the consumption is completed, so as to ensure message integrity.

How long are messages in the Kafka client retained? How do I set the consumer offset?

Messages in the Kafka client are retained for 1 day. You can set the `auto_offset_reset` parameter of Kafka as `earliest` or `latest` as needed. If you need to consume data from a specific offset, you can reset the consumer offset with the seek feature of the Kafka client.

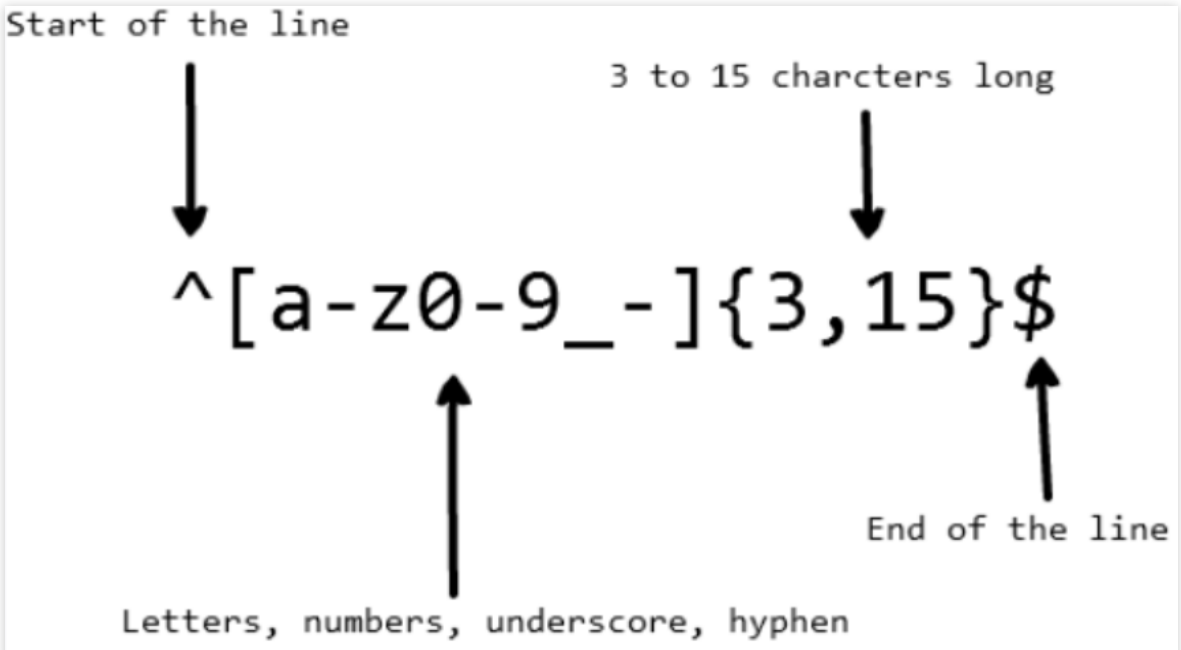
Regular Expressions for Subscription

Last updated：2024-07-08 15:39:24

What is a regular expression?

> A regular expression is used to search for a specific pattern from text.

A regular expression matches a string from left to right. "Regular expression" is often referred to as "regex" or "regexp" for short.
A regex can be used to replace text in strings, validate forms, extract a substring from a string based on a pattern match, and much more.
If you are developing an application, you may want to set rules on eligible usernames, which can contain letters, digits, underscores, and hyphens.
You may also want to limit the number of characters in a username for better display effect. The following regex can be used to validate a username:



The above regex can match the strings `john_doe` , `jo-hn\\_doe` , and `john12\\_as` , but not `Jo` as it contains an uppercase letter and is too short.

Contents

Basic Matchers

A regex is just a pattern of characters used to perform a search in text. For example, the regex `cat` means: the letter `c` , followed by the letter `a` , followed by the letter `t` .

```
"cat" => The cat sat on the mat
```

The regex `123` can match the string "123". A regex is matched against the input string by comparing each character in the regex with each character in the input string one by one.

Regexes are normally case-sensitive, so the regex `Cat` would not match the string "cat".

```
"Cat" => The cat sat on the Cat
```

Metacharacters

Metacharacters are the building blocks of regexes. They do not stand for themselves; instead, they need to be interpreted in certain special ways. Some metacharacters enclosed in square brackets have special meaning. Below are the metacharacters:

Metacharacter	Description
.	Matches any character except a line break.
[ ]	Character class, which matches any character enclosed in square brackets.
[^]	Negated character class, which matches any character not enclosed in square brackets.
*	Matches zero or more repetitions of the preceding subexpression.
+	Matches one or more repetitions of the preceding subexpression.
?	Matches zero or one repetition of the preceding subexpression or specifies a non-greedy qualifier.
{n,m}	Braces, which matches the preceding character at least n times but not more than m times.
(xyz)	Capturing group, which matches the character "xyz" in an exact order.
|	Alternation, which matches the characters before or after the symbol.
&#92;	Escape character, which can restore the original meaning of metacharacters and allows you to match reserved characters [ ] ( ) { } . * + ? ^ $ \\ |
^	Matches the beginning-of-line character.
$	Matches the end-of-line character.

Period

The simplest example of metacharacters is the period `.` , which can match any single character but not a line break or newline character. For example, the regex `.ar` means: any character, followed by the letter `a` , followed by the letter `r` .

```
".ar" => The car parked in the garage.
```

Character Set

A character set is also known as a character class, which is specified by square brackets. A hyphen in a character set is used to specify the character range. The order of the character range inside square brackets does not matter.
For example, the regex `[Tt]he` means: the uppercase letter `T` or the lowercase letter `t`, followed by the letter `h`, followed by the letter `e`.

```
"[Tt]he" => The car parked in the garage.
```

However, the period in character sets is what it means literally. For example, the regex `ar[.]` means: the lowercase letter `a`, followed by the letter `r`, followed by the period `.`

```
"ar[.]" => A garage is a good place to park a car.
```

Negated Character Set

Generally, the caret symbol `^` represents the start of a string, but when enclosed in square brackets, it negates the character set. For example, the regex `[^c]ar` means: any character except the letter `c`, followed by the character `a`,
followed by the letter `r`.

```
"[^c]ar" => The car parked in the garage.
```

Repetition

The metacharacters `+`, `*`, and `?` are used to specify how many times a subpattern can appear. These metacharacters act differently in different situations.

Asterisk

The symbol `*` matches zero or more repetitions of the preceding matcher. For example, the regex `a*` matches zero or more repetitions of the preceding lowercase letter `a`. However, if it appears after a character set, then it finds the repetitions of the whole character set.
For example, the regex `[a-z]*` means: any number of lowercase letters in a row.

```
"[a-z]*" => The car parked in the garage #21.
```

The symbol `*` can be used together with the metacharacter `.` to match the arbitrary string `.*`. It can also be used together with the whitespace character `\\s` to match a string of whitespace characters.
For example, the regex `\\s*cat\\s*` means: zero or more whitespaces, followed by the lowercase letter `c`, followed by the lowercase letter `a`, followed by the lowercase character `t`, followed by zero or more whitespaces.

```
"\\s*cat\\s*" => The fat cat sat on the cat.
```

Plus Sign

The symbol `+` matches one or more repetitions of the preceding character. For example, the regex `c.+t` means: the lowercase letter `c`, followed by at least one character, followed by the lowercase letter `t`.

```
"c.+t" => The fat cat sat on the mat.
```

Question Mark

The metacharacter `?` makes the preceding character optional and matches zero or one repetition of the preceding character.

For example, the regex `[T]?he` means: the optional uppercase character `T`, followed by the lowercase letter `h`, followed by the lowercase letter `e`.

```
"[T]he" => The car is parked in the garage.
```

```
"[T]?he" => The car is parked in the garage.
```

Braces

In regexes, braces, aka quantifiers, are used to specify how many times a character or a group of characters can be repeated. For example, the regex `[0-9]{2,3}` means: match at least 2 digits but not more than 3 digits (characters in the range of 0 to 9).

```
"[0-9]{2,3}" => The number was 9.9997 but we rounded it off to 10.0.
```

The second number can be left out. For example, the regex `[0-9]{2,}` means: match 2 or more digits. If the comma is also removed, the regex [0-9]{2} means: match exactly 2 digits.

```
"[0-9]{2,}" => The number was 9.9997 but we rounded it off to 10.0.
```

```
"[0-9]{2}" => The number was 9.9997 but we rounded it off to 10.0.
```

Capturing Group

A capturing group is a group of subpatterns enclosed in parentheses and is denoted as `(...)`. If a quantifier is placed after a character, it will repeat the preceding character.

However, if a quantifier is placed after a capturing group, it will repeat the whole capturing group.

For example, the regex `(ab)*` matches zero or more repetitions of the string "ab". The metacharacter `|` can be used in a capturing group. For example, the regex `(c|g|p)ar` means: the lowercase letter `c`, `g`, or `p`, followed by the letter `a`, followed by the letter `r`.

```
"(c|g|p)ar" => The car is parked in the garage.
```

Alternation

The vertical bar `|` is used to define alternation that is like a condition between multiple expressions. Alternation seems to work in the same way as character set.
However, the great difference is that alternation can be used at the expression level, while character set at the character level.
For example, the regex `(T|t)he|car` means: the uppercase character `T` or the lowercase letter `t`, followed by `h`, followed by `e` or `c`, followed by `a`, followed by `r`.

```
"(T|t)he|car" => The car is parked in the garage.
```

Escape Character

The backslash `\\` is used to escape the next character, allowing you to specify a symbol as a matching character including reserved characters `{ } [ ] / \\ + * . $ ^ | ?`. To use a special character as a matching character, prepend `\\` before it.
For example, the regex `.` is used to match any character except a line break. To match the character `.` in the input string, the regex `(f|c|m)at\\.?` means: the lowercase letter `f`, `c`, or `m`, followed by the lowercase letter `a`, followed by the lowercase letter `t`, followed by the optional `.` character.

```
"(f|c|m)at\\.?" => The fat cat sat on the mat.
```

Anchors

Anchors in regexes are used to check whether the matching symbol is the starting or ending symbol of the input string. There are two types of anchors: `^` (which checks whether the matching character is the start character of the input string) and `$` (which checks whether the matching character is the end character).

Caret

The caret `^` is used to check whether a matching character is the first character of the input string. If the regex `^a` (if a is the starting symbol) is used to match the string `abc`, it matches `a`.
However, if the regex `^b` is used, it does not match anything, because "b" in the string `abc` is not the start character.
The regex `^(T|t)he` means that the uppercase character `T` or the lowercase letter `t` is the starting symbol of the input string, followed by the letter `h`, followed by the lowercase letter `e`.

```
"(T|t)he" => The car is parked in the garage.
```

```
"^(T|t)he" => The car is parked in the garage.
```

Dollar Sign

The dollar sign `$` is used to check whether a matching character is the last character of the input string. For example, the regex `(at\\.)$` means: the lowercase letter `a`, followed by the lowercase letter `t`, followed by the character `.`, and the matcher must be the end of the string.

```
"(at\\.)" => The fat cat. sat. on the mat.
```

```
"(at\\.)$" => The fat cat sat on the mat.
```

hand Character Sets

There are shorthands for commonly used character sets and regexes. The shorthand character sets are as follows:

Shorthand	Description
.	Matches any character except a line break
\\w	Matches alphanumeric characters: `[a-zA-Z0-9_]`
\\W	Matches non-alphanumeric characters: `[^\\w]`
\\d	Matches digits: `[0-9]`
\\D	Matches non-digits: `[^\\d]`
\\s	Matches whitespace character: `[\\t\\n\\f\\r\\p{Z}]`
\\S	Matches non-whitespace character: `[^\\s]`

Lookaround

Lookbehind and lookahead (also called lookaround) are specific types of ***non-capturing groups*** (used to match the pattern but not included in the matching list). Lookarounds are used when there is the condition that this pattern is preceded or followed by another certain pattern.

For example, to get all the numbers and the `.` character that are preceded by the character `$` in the input string `$4.44 and $10.88`, the regex `(?<=\\$)[0-9\\.]*` can be used.

Below are the lookarounds used in regexes:

Symbol	Description
?=	Positive lookahead
?!	Negative lookahead
?<=	Positive lookbehind
?<!	Negative lookbehind

Positive Lookahead

A positive lookahead asserts that the first part of the expression must be followed by the lookahead expression. The returned match only contains the text that is matched by the first part of the expression.
To define a positive lookahead, parentheses are used. Within those parentheses, a question mark with equal sign is denoted as `(?=...)`. The lookahead expression is written after the equal sign inside parentheses.
For example, the regex `(T|t)he(?=\\sfat)` means: the uppercase letter `T` or lowercase letter `t`, followed by the letter `h`, followed by the lowercase letter `e` or `c`.
In parentheses, the positive lookahead is defined, which tells the regex engine to match `The` or `the` which is followed by the word `fat`.

```
"(at\\.)$" => The fat cat sat on the mat.
```

Negative Lookahead

A negative lookahead is used to get the content that does not match the expression from the input string and is defined in the same way as positive lookahead.
The only difference lies in that a negative lookahead uses the negation symbol `!` instead of the equal sign `=`, such as `(?!...)`.
For example, the regex `(T|t)he(?!\\sfat)` means: get all the words `The` or `the` and add a whitespace character before the unmatched `fat` word from the input string.

```
"(T|t)he(?!\\sfat)" => The fat cat sat on the mat.
```

Positive Lookbehind

A positive lookbehind is used to get all the matches that are preceded by a specific pattern and is denoted as `(?<=...)`. For example, the regex `(?<=(T|t)he\\s)(fat|mat)` means: get all the words `fat` and `mat` after the word `The` or `the` from the input string.

```
"(?<=(T|t)he\\s)(fat|mat)" => The fat cat sat on the mat.
```

Negative Lookbehind

A negative lookbehind is used to get all the matches that are not preceded by a specific pattern and is denoted as `(?<!...)`. For example, the regex `(?<!(T|t)he\\s)(cat)` means: get all the `cat` words that are not after the word `The` or `the` from the input string.

```
"(?<!(T|t)he\\s)(cat)" => The cat sat on cat.
```

Flags

Flags are also called modifiers as they modify the output of regexes. They can be used in any order or combination and are an integral part of a regex.

Flag	Description
i	Case-insensitive: Sets matching to be case-insensitive.
g	Global search: Searches for all the matches throughout the input string.
m	Multiline match: Matches every line of the input string.

Case Insensitivity

The modifier `i` is used to perform a case-insensitive match. For example, the regex `/The/gi` means: the uppercase letter `T`, followed by the lowercase letter `h`, followed by the lowercase letter `e`.
At the end of the regex, the flag `i` tells the regex to ignore the case. As can be seen, the flag `g` is also used so as to search for matches in the whole input string.

```
"The" => The fat cat sat on the mat.
```

```
"/The/gi" => The fat cat sat on the mat.
```

Global Search

The modifier `g` is used to perform a global match (find all matches rather than stopping after the first match).
For example, the regex `/.(at)/g` means: any character except a line break, followed by the lowercase letter `a`, followed by the lowercase letter `t`.
As the flag `g` is used at the end of the regex, it will find all matches in the input string.

```
".(at)" => The fat cat sat on the mat.
```

```
"/.(at)/g" => The fat cat sat on the mat.
```

Multiline

The modifier `m` is used to perform multiline matching. As discussed earlier, anchors `(^, $)` are used to check whether the matched character is the beginning or end of the input string. To have anchors work on each line, the flag `m` should be used.

For example, the regex `/at(.)?$/gm` means: the lowercase letter `a`, followed by the lowercase letter `t`, and optionally zero or one repetition of any character except line break. Because the modifier `m` is at the end of the regex, the regex engine matches pattern at the end of each line in a string.

```
"/.at(.)?$/" => The fat
              cat sat

       on the mat.
```

```
"/.at(.)?$/gm" => The fat
            cat sat
            on the mat.
```

Common Regexes

Type	Expression
Positive integer	^-\\d+$
Negative integer	^-\\d+$
Phone number	^+?[\\d\\s]{3,}$
Phone code	^+?[\\d\\s]+(?[\\d\\s]{10,}$
Integer	^-?\\d+$
Username	^[\\w\\d_.]{4,16}$
Alphanumeric character	^[a-zA-Z0-9]*$
Alphanumeric character with whitespace	^[a-zA-Z0-9]*$
Password	^(?=^.{6,}$)((?=.*[A-Za-z0-9])(?=.*[A-Z])(?=.*[a-z]))^.*$

Email	^([a-zA-Z0-9._%-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4})*$
IPv4 address	^((?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?))*$`
Lowercase letters	^([a-z])*$
Uppercase letter	^([A-Z])*$
Username	^[\\w\\d_.]{4,16}$
URL	^(((http|https|ftp):\\/\\/)?([[a-zA-Z0-9]\\-\\.])+(\\.)([[a-zA-Z0-9]]){2,4}([[a-zA-Z0-9]\\/+=%&_\\.~?\\-]*))*$
Visa credit card number	^(4[0-9]{12}(?:[0-9]{3})?)*$
Date (MM/DD/YYYY)	^(0?[1-9]|1[012])[- /.](0?[1-9]|[12][0-9]|3[01])[- /.](19|20)?[0-9]{2}$
Date (YYYY/MM/DD)	^(19|20)?[0-9]{2}[- /.](0?[1-9]|1[012])[- /.](0?[1-9]|[12][0-9]|3[01])$
Mastercard credit card number	^(5[1-5][0-9]{14})*$