

tf.get_variable

```
get_variable(  
    name,  
    shape=None,  
    dtype=None,  
    initializer=None,  
    regularizer=None,  
    trainable=True,  
    collections=None,  
    caching_device=None,  
    partitioner=None,  
    validate_shape=True,  
    use_resource=None,  
    custom_getter=None,  
    constraint=None  
)
```

Defined in [tensorflow/python/ops/variable_scope.py](#).

See the guide: [Variables > Sharing Variables](#)

Gets an existing variable with these parameters or create a new one.

This function prefixes the name with the current variable scope and performs reuse checks. See the [Variable Scope How To](#) for an extensive description of how reusing works. Here is a basic example:

```
def foo():  
    with tf.variable_scope("foo", reuse=tf.AUTO_REUSE):  
        v = tf.get_variable("v", [1])  
    return v  
  
v1 = foo() # Creates v.  
v2 = foo() # Gets the same, existing v.  
assert v1 == v2
```

If `initializer` is `None` (the default), the default initializer passed in the variable scope will be used. If that one is `None` too, a `glorot_uniform_initializer` will be used. The initializer can also be a Tensor, in which case the variable is initialized to this value and shape.

Similarly, if the `regularizer` is `None` (the default), the default regularizer passed in the variable scope will be used (if that is `None` too, then by default no regularization is performed).

If a partitioner is provided, a `PartitionedVariable` is returned. Accessing this object as a `Tensor` returns the shards concatenated along the partition axis.

Some useful partitioners are available. See, e.g., `variable_axis_size_partitioner` and `min_max_variable_partitioner`.

Args:

- `name`: The name of the new or existing variable.
- `shape`: Shape of the new or existing variable.
- `dtype`: Type of the new or existing variable (defaults to `DT_FLOAT`).

- `initializer` : Initializer for the variable if one is created.
- `regularizer` : A (Tensor -> Tensor or None) function; the result of applying it on a newly created variable will be added to the collection `tf.GraphKeys.REGULARIZATION_LOSSES` and can be used for regularization.
- `trainable` : If `True` also add the variable to the graph collection `GraphKeys.TRAINABLE_VARIABLES` (see `tf.Variable`).
- `collections` : List of graph collections keys to add the Variable to. Defaults to `[GraphKeys.GLOBAL_VARIABLES]` (see `tf.Variable`).
- `caching_device` : Optional device string or function describing where the Variable should be cached for reading. Defaults to the Variable's device. If not `None`, caches on another device. Typical use is to cache on the device where the Ops using the Variable reside, to deduplicate copying through `Switch` and other conditional statements.
- `partitioner` : Optional callable that accepts a fully defined `TensorShape` and `dtype` of the Variable to be created, and returns a list of partitions for each axis (currently only one axis can be partitioned).
- `validate_shape` : If False, allows the variable to be initialized with a value of unknown shape. If True, the default, the shape of `initial_value` must be known.
- `use_resource` : If False, creates a regular Variable. If true, creates an experimental ResourceVariable instead with well-defined semantics. Defaults to False (will later change to True). In Eager mode, this argument is always forced to be True.
- `custom_getter` : Callable that takes as a first argument the true getter, and allows overwriting the internal `get_variable` method. The signature of `custom_getter` should match that of this method, but the most future-proof version will allow for changes: `def custom_getter(getter, *args, **kwargs)`. Direct access to all `get_variable` parameters is also allowed: `def custom_getter(getter, name, *args, **kwargs)`. A simple identity custom getter that simply creates variables with modified names is: `python def custom_getter(getter, name, *args, **kwargs): return getter(name + '_suffix', *args, **kwargs)`

Returns:

The created or existing `Variable` (or `PartitionedVariable`, if a partitioner was used).

Raises:

- `ValueError` : when creating a new variable and shape is not declared, when violating reuse during variable creation, or when `initializer` dtype and `dtype` don't match. Reuse is set inside `variable_scope`.

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.

Last updated November 2, 2017.

Stay Connected

Blog
 GitHub
 Twitter

Support

Issue Tracker
 Release Notes

English

[Terms](#) | [Privacy](#)