```
/tensorflow/python/distribute:collective_all_reduce_strategy)"
//tensorflow:tf_python_api_gen_v2
//tensorflow:create_tensorflow.python_api_tf_python_api_gen_v2
//tensorflow/python:no_contrib
//tensorflow/python/distribute:combinations
//tensorflow/python/distribute:collective_all_reduce_strategy
```

```
(hm) wxf@seir19:~/tf2/tensorflow$ grep -wnr "create_tensorflow" # 找不到
experiments/test_new_python_file/README.md:62://tensorflow:create_tensorflow.python_api_tf_python_api_gen_v2
experiments/test_new_python_file/README.md:159://tensorflow:create_tensorflow.python_api_tf_python_api_gen_v2
(hm) wxf@seir19:~/tf2/tensorflow$ grep -wnr "create_" # 找到了
tensorflow/python/tools/api/generator/api_gen.bzl:99:    api_gen_binary_target = ("create_" + primary_package + "_api_%s") % name
```

```
def gen_api_init_files(
        name,
        output_files = TENSORFLOW_API_INIT_FILES,
        root_init_template = None,
        srcs = [],
        api_name = "tensorflow",
        api_version = 2,
        compat_api_versions = [],
        compat_init_templates = [],
        packages = [
            "tensorflow.python",
            "tensorflow.lite.python.analyzer",
            "tensorflow.lite.python.lite",
            "tensorflow.lite.python.authoring.authoring",
            "tensorflow.python.modules_with_exports",
        ],
        package_deps = [
            "//tensorflow/python:no_contrib",
            "//tensorflow/python:modules_with_exports",
        ],
        output_package = "tensorflow",
        output_dir = "",
        root_file_name = "__init__.py"):
    """Creates API directory structure and __init__.py files.

    Creates a genrule that generates a directory structure with __init__.py
    files that import all exported modules (i.e. modules with tf_export
    decorators).

    Args:
      name: name of genrule to create.
      output_files: List of __init__.py files that should be generated.
        This list should include file name for every module exported using
        tf_export. For e.g. if an op is decorated with
        @tf_export('module1.module2', 'module3'). Then, output_files should
        include module1/module2/__init__.py and module3/__init__.py.
      root_init_template: Python init file that should be used as template for
        root __init__.py file. "# API IMPORTS PLACEHOLDER" comment inside this
        template will be replaced with root imports collected by this genrule.
      srcs: genrule sources. If passing root_init_template, the template file
        must be included in sources.
      api_name: Name of the project that you want to generate API files for
        (e.g. "tensorflow" or "estimator").
      api_version: TensorFlow API version to generate. Must be either 1 or 2.
      compat_api_versions: Older TensorFlow API versions to generate under
        compat/ directory.
      compat_init_templates: Python init file that should be used as template
        for top level __init__.py files under compat/vN directories.
        "# API IMPORTS PLACEHOLDER" comment inside this
        template will be replaced with root imports collected by this genrule.
      packages: Python packages containing the @tf_export decorators you want to
        process
      package_deps: Python library target containing your packages.
      output_package: Package where generated API will be added to.
      output_dir: Subdirectory to output API to.
        If non-empty, must end with '/'.
      root_file_name: Name of the root file with all the root imports.
    """
    root_init_template_flag = ""
    if root_init_template:
        root_init_template_flag = "--root_init_template=" + root_init_template

    primary_package = packages[0]
    api_gen_binary_target = ("create_" + primary_package + "_api_%s") % name
    native.py_binary(
        name = api_gen_binary_target,
        srcs = ["//tensorflow/python/tools/api/generator:create_python_api.py"],
        main = "//tensorflow/python/tools/api/generator:create_python_api.py",
        python_version = "PY3",
        srcs_version = "PY3",
        visibility = ["//visibility:public"],
        deps = package_deps + [
            "//tensorflow/python:util",
            "//tensorflow/python/tools/api/generator:doc_srcs",
        ],
    )

    # Replace name of root file with root_file_name.
    output_files = [
        root_file_name if f == "__init__.py" else f
        for f in output_files
    ]
    all_output_files = ["%s%s" % (output_dir, f) for f in output_files]
    compat_api_version_flags = ""
    for compat_api_version in compat_api_versions:
        compat_api_version_flags += " --compat_apiversion=%d" % compat_api_version

    compat_init_template_flags = ""
    for compat_init_template in compat_init_templates:
        compat_init_template_flags += (
            " --compat_init_template=$(location %s)" % compat_init_template
        )

    # copybara:uncomment_begin(configurable API loading)
    # native.vardef("TF_API_INIT_LOADING", "default")
    # loading_flag = " --loading=$(TF_API_INIT_LOADING)"
    # copybara:uncomment_end_and_comment_begin
    loading_flag = " --loading=default"
    # copybara:comment_end

    native.genrule(
        name = name,
        outs = all_output_files,
        cmd = (
            "$(location :" + api_gen_binary_target + ") " +
            root_init_template_flag + " --apidir=$(@D)" + output_dir +
            " --apiname=" + api_name + " --apiversion=" + str(api_version) +
            compat_api_version_flags + " " + compat_init_template_flags +
            loading_flag + " --packages=" + ",".join(packages) +
            " --output_package=" + output_package +
            " --use_relative_imports=True $(OUTS)"
        ),
        srcs = srcs,
        tools = [":" + api_gen_binary_target],
        visibility = [
            "//tensorflow:__pkg__",
            "//tensorflow/tools/api/tests:__pkg__",
        ],
    )
```

```
py_library(
    name = "tensorflow_py_no_contrib",
    srcs = select({
        "api_version_2": [":tf_python_api_gen_v2"],
        "//conditions:default": [":tf_python_api_gen_v1"],
    }) + [
        ":root_init_gen",
        "//tensorflow/python/keras/api:keras_python_api_gen",
        "//tensorflow/python/keras/api:keras_python_api_gen_compat_v1",
        "//tensorflow/python/keras/api:keras_python_api_gen_compat_v2",
    ],
    srcs_version = "PY3",
    visibility = ["//visibility:public"],
    deps = ["//tensorflow/python:no_contrib"],
)
# copybara:comment_end
```

```
py_library(
    name = "no_contrib",
    srcs = ["__init__.py"],
    srcs_version = "PY3",
    visibility = [
        "//tensorflow:__pkg__",
        "//tensorflow/python/estimator:__subpackages__",
        "//tensorflow/python/keras:__subpackages__",
        "//tensorflow/python/tools:__pkg__",
        "//tensorflow/python/tools/api/generator:__pkg__",
        "//tensorflow/tools/api/tests:__pkg__",
        "//tensorflow/tools/compatibility/update:__pkg__",
        "//third_party/py/keras:__subpackages__",
        "//third_party/py/tensorflow_core:__subpackages__",
    ],
    deps = [
        ":_pywrap_py_exception_registry",
        ":_pywrap_quantize_training",
        ":_pywrap_utils",
        ":array_ops",
        ":audio_ops_gen",
        ":bincount_ops",
        ":bitwise_ops",
        ":boosted_trees_ops",
        ":check_ops",
        ":client_testlib",
        ":clustering_ops",
        ":collective_ops",
        ":composite_tensor_ops",
        ":cond_v2",
        ":confusion_matrix",
        ":control_flow_ops",
        ":cudnn_rnn_ops_gen",
        ":distributed_framework_test_lib",
        ":functional_ops",
        ":gradient_checker",
        ":gradient_checker_v2",
        ":histogram_ops",
        ":image_ops",
        ":initializers_ns",
        ":io_ops",
        ":keras_lib",
        ":lib",
        ":list_ops",
        ":manip_ops",
        ":map_fn",
        ":map_ops",
        ":math_ops",
        ":metrics",
        ":nccl_ops",
        ":nn",
        ":ops",
        ":platform",
        ":proto_ops",
        ":pywrap_tensorflow",
        ":pywrap_tfe",
        ":rnn_ops_gen",
        ":script_ops",
        ":sendrecv_ops_gen",
        ":session_ops",
        ":sets",
        ":sparse_ops",
        ":standard_ops",
        ":state_ops",
        ":string_ops",
        ":tensor_array_ops",
        ":training",
        ":weights_broadcast_ops",
        ":while_v2",
        "//tensorflow/core:protos_all_py",
        "//tensorflow/lite/python:analyzer",
        "//tensorflow/lite/python:lite",
        "//tensorflow/lite/python/authoring",
        "//tensorflow/python/client",
        "//tensorflow/python/client:_pywrap_events_writer",
        "//tensorflow/python/client:pywrap_tf_session",
        "//tensorflow/python/compat",
        "//tensorflow/python/compat:v2_compat",
        "//tensorflow/python/compiler",
        "//tensorflow/python/data",
        "//tensorflow/python/debug:debug_py",
        "//tensorflow/python/distribute",
        "//tensorflow/python/distribute:combinations",  # For tf.__internal__ API.
        "//tensorflow/python/distribute:distribute_config",
        "//tensorflow/python/distribute:estimator_training",
        "//tensorflow/python/distribute:strategy_combinations",  # For tf.__internal__,
        "//tensorflow/python/distribute/experimental/rpc:rpc_ops",
        "//tensorflow/python/dlpack",
        "//tensorflow/python/eager:def_function",
        "//tensorflow/python/eager:monitoring",
        "//tensorflow/python/eager:profiler",
        "//tensorflow/python/eager:profiler_client",
        "//tensorflow/python/eager:remote",
        "//tensorflow/python/framework",
        "//tensorflow/python/framework:_pywrap_python_op_gen",
        "//tensorflow/python/framework:combinations",
        "//tensorflow/python/framework:config",
        "//tensorflow/python/framework:errors",
        "//tensorflow/python/framework:extension_type",
        "//tensorflow/python/framework:for_generated_wrappers",
        "//tensorflow/python/framework:graph_util",
        "//tensorflow/python/framework:kernels",
        "//tensorflow/python/framework:subscribe",
        "//tensorflow/python/framework:test_ops",  # TODO(b/183988750): Break testing code out into separate rule.
        "//tensorflow/python/grappler:tf_cluster",
        "//tensorflow/python/grappler:tf_item",
        "//tensorflow/python/grappler:tf_optimizer",
        "//tensorflow/python/module",
        "//tensorflow/python/ops/distributions",
        "//tensorflow/python/ops/linalg",
        "//tensorflow/python/ops/linalg/sparse",
        "//tensorflow/python/ops/losses",
        "//tensorflow/python/ops/numpy_ops:numpy",
        "//tensorflow/python/ops/parallel_for",
        "//tensorflow/python/ops/ragged",
        "//tensorflow/python/ops/signal",
        "//tensorflow/python/platform:_pywrap_stacktrace_handler",
        "//tensorflow/python/profiler",
        "//tensorflow/python/profiler:profiler_client",
        "//tensorflow/python/profiler:profiler_v2",
        "//tensorflow/python/profiler:trace",
        "//tensorflow/python/saved_model",
        "//tensorflow/python/summary",
        "//tensorflow/python/summary/writer",
        "//tensorflow/python/tools:module_util",
        "//tensorflow/python/tools/api/generator:create_python_api",
        "//tensorflow/python/tpu:tpu_noestimator",
        "//tensorflow/python/training:saver_test_utils",
        "//tensorflow/python/types",
        "//tensorflow/python/util",
        "//tensorflow/python/util:_pywrap_checkpoint_reader",
        "//tensorflow/python/util:_pywrap_kernel_registry",
        "//tensorflow/python/util:_pywrap_nest",
        "//tensorflow/python/util:_pywrap_stat_summarizer",
        "//tensorflow/python/util:_pywrap_tfprof",
        "//tensorflow/python/util:_pywrap_transform_graph",
        "//tensorflow/python/util:_pywrap_util_port",
        "//third_party/py/numpy",
    ],
)
```

```
py_library(
    name = "combinations",
    srcs = ["combinations.py"],
    srcs_version = "PY3",
    visibility = [
        "//tensorflow:internal",
        "//tensorflow_models:__subpackages__",
        "//third_party/py/keras:__subpackages__",
    ],
    deps = [
        ":collective_all_reduce_strategy",
        ":distribute_lib",
        ":multi_process_runner",
        ":multi_worker_test_base",
        "//tensorflow/python:framework_combinations",
        "//tensorflow/python:framework_ops",
        "//tensorflow/python:framework_test_combinations_lib",
        "//tensorflow/python:framework_test_lib",
        "//tensorflow/python:platform",
        "//tensorflow/python:session",
        "//tensorflow/python:tf_decorator",
        "//tensorflow/python/eager:context",
        "//tensorflow/python/eager:def_function",
        "//tensorflow/python/util:tf_export",
        "@six_archive//:six",
    ],
)
```

```
py_library(
    name = "collective_all_reduce_strategy",
    srcs = ["collective_all_reduce_strategy.py"],
    srcs_version = "PY3",
    visibility = ["//tensorflow:internal"],
    deps = [
        ":collective_util",
        ":cross_device_ops",
        ":cross_device_utils",
        ":device_util",
        ":distribute_lib",
        ":distribute_utils",
        ":input_lib",
        ":mirrored_strategy",
        ":multi_worker_util",
        ":numpy_dataset",
        ":reduce_util",
        ":values",
        "//tensorflow/core:protos_all_py",
        "//tensorflow/python:array_ops",
        "//tensorflow/python:collective_ops",
        "//tensorflow/python:errors",
        "//tensorflow/python:framework_ops",
        "//tensorflow/python:platform",
        "//tensorflow/python:training",
        "//tensorflow/python:util",
        "//tensorflow/python/distribute/cluster_resolver:cluster_resolver_lib",
        "//tensorflow/python/eager:context",
        "//tensorflow/python/util:tf_export",
    ],
)
```