WILSON SONSINI

Wilson Sonsini Goodrich & Rosati
Professional Corporation
[redacted]

[redacted]

April 3, 2024

Via PDF email and FedEx

[redacted]

Re: HashiCorp BSL License Terms– Cease and Desist

Gentlemen:

We represent HashiCorp, Inc. As you know, HashiCorp is the maker of Terraform, an infrastructure as code software tool that allows users to manage environments with a configuration language called the HashiCorp Configuration Language (HCL) for human-readable, automated deployments.

Following our letters of August 14, August 25, and November 27, 2023, to specific OpenTofu sponsors (including Digger, Inc., Spacelift, Inc., and EnvZero CEO [redacted], in [redacted] capacity as an OpenTF Initiative member), this is a cease and desist demand to the supporters of the OpenTofu project. Specifically, OpenTofu has repeatedly taken code HashiCorp provided only under the Business Software License (BSL) and used it in a manner that violates those license terms and HashiCorp's intellectual property rights. In at least some instances, OpenTofu has incorrectly re-labeled HashiCorp's code to make it appear as if it was made available by HashiCorp originally under a different license.

HashiCorp has put OpenTofu and its sponsors on notice several times of various issues involving violation of HashiCorp's rights and misrepresenting or misapplying the terms of the

BSL, some of which have received no response. Therefore, we demand that OpenTofu provide a written response to our demands no later than ***Wednesday, April 10, 2024***, and cease and desist from further violations of HashiCorp's BSL license and infringement of HashiCorp's copyrights. If OpenTofu does not comply, we reserve all rights, including the right to send DMCA takedown notices to Github or any other third party hosting or source code repository provider, and the right to initiate litigation to stop further violations.

A. HashiCorp's Terraform Project BSL License.

HashiCorp's Terraform Project uses the Business Software License (BSL). The license consists of the standard BSL terms together with the HashiCorp specific Additional License Grant, and is available online at https://www.hashicorp.com/bsl.

The BSL provides, among other terms, that (1) all copies of the Licensed Work (as defined in BSL) and derivative works of the Licensed Work, are subject to the BSL, (2) the licensee must conspicuously display the BSL license on each original or modified copy of the Licensed Work, and (3) any use of the Licensed Work in violation of the license will automatically terminate the licensee's rights under the license for the current and all other versions of the Licensed Work. Further, the rights granted are limited to non-production use, except as set forth in the Additional License Grant, which provides a limited authorization to make production use of Terraform, so long as the use "does not include offering the Licensed Work to third parties on a hosted or embedded basis in order to compete with HashiCorp's paid version(s) of the Licensed Work". The license expressly states that any use that does not comply with the license requires the purchase of a commercial license from the rightsholder. The terms of the MPL, under which OpenTofu distributes its project, are wholly incompatible with the BSL.

In short, HashiCorp has promulgated Terraform under license terms such that use of the code other than in compliance with the terms, including production uses for competitive purposes is both a breach of contract as well as a violation of the condition of the license, rendering any such non-compliant use willfully infringing on HashiCorp's copyrights (registration of which is currently pending).

B. OpenTofu's Violations of HashiCorp's BSL License Terms.

Our investigation shows that OpenTofu has used HashiCorp code made available only under BSL license terms in violation of those terms. By way of example:

- On November 29, 2023, HashiCorp published a series of updates to its Terraform repository on Github (https://github.com/hashicorp/terraform), including the following files (the "Terraform Files"):
    - remove_statement.go (https://github.com/hashicorp/terraform/blob/main/internal/refactoring/remove_statement.go);

- removed.go (https://github.com/hashicorp/terraform/blob/main/internal/configs/removed.go);
- removed_test.go (https://github.com/hashicorp/terraform/blob/main/internal/configs/removed_test.go);
- remove_target.go (https://github.com/hashicorp/terraform/blob/main/internal/addrs/remove_target.go); and
- remove_target_test.go (https://github.com/hashicorp/terraform/blob/main/internal/addrs/remove_target_test.go).

- Each of the Terraform Files are comprised entirely of code never before published, contain the following header in the first two lines, identifying the file as being authored and copyrighted by HashiCorp, Inc., and subject to the BSL:

```
// Copyright © HashiCorp, Inc.
// SPDX-License-Identifier: BUSL-1.1
```

- On February 21st, the OpenTofu Project published a series of updates to its opentofu repository (https://github.com/opentofu/opentofu), including the following files (the "OpenTofu Files"):
  - remove_statement.go (https://github.com/opentofu/opentofu/blob/main/internal/refactoring/remove_statement.go)
  - removed.go (https://github.com/opentofu/opentofu/blob/main/internal/configs/removed.go);
  - removed_test.go (https://github.com/opentofu/opentofu/blob/main/internal/configs/removed_test.go);
  - remove_endpoint.go (https://github.com/opentofu/opentofu/blob/main/internal/configs/removed_test.go); and
  - remove_target_test.go (https://github.com/opentofu/opentofu/blob/main/internal/addrs/remove_endpoint_test.go).

Each of the above-referenced OpenTofu files was submitted and committed by [redacted], whose LinkedIn profile indicates [redacted] is presently a Core Contributor at OpenTofu ("Full-Time"), and a Full Stack Engineer ("Full-Time"), envZero Inc.: [redacted]

- The OpenTofu Files were reviewed and commented on prior to publication (including initiating a discussion about why HashiCorp copyright notices were being added) by [redacted], whose LinkedIn profile indicates [redacted] is presently a Core Contributor at OpenTofu ("Full-Time"), and a Software Engineer at Spacelift, Inc. ("Full Time"): [redacted]

- The OpenTofu Files were reviewed and commented on prior to publication by [redacted], whose LinkedIn profile indicates [redacted] is presently an [redacted] at OpenTofu, and a Software Engineering Team Lead at Spacelift, Inc. ("Full Time"): [redacted]

- The OpenTofu Files were also reviewed and commented on prior to publication by [redacted] and [redacted], both of whom are identified on Spacelift's website as Sponsored Dedicated Contributors to OpenTofu: [redacted]

Each of the OpenTofu Files contains the following header near the top, identifying the file as being authored and copyrighted by HashiCorp, Inc., but removing HashiCorp's license designation and replacing it with an incorrect reference to the Mozilla Public License (MPL 2.0
-):

```
// Copyrig©(c) 2023 HashiCorp, Inc.
// SPDX-License-Identifier: MPL-2.0
```

- In our review of the relevant files on March 25th, 2024, we observed that not only is the structure and substance of each of the OpenTofu Files substantially similar to the original corresponding Terraform Files, each contains substantial overlap with the code and comments contained in the original Terraform Files. While the Terraform Files include some new code, much of their contents comprise HashiCorp BSL code in its original unmodified form, as well as sections where OpenTofu has made minor modifications to the original HashiCorp code such as altering the names of packages and variables, and revising explanatory comments. Comparisons files (attached), clearly show the portions of the original copyrighted HashiCorp code that either remain unchanged or have been modified in immaterial ways. As derivative works of the Terraform Files, the OpenTofu Files are required to be governed by the BSL.

These are mere examples reflecting instances of HashiCorp code licensed only under BSL incorporated in the OpenTofu with little or no modification, and purportedly re-licensed under the MPL. While there are certainly some differences between the Terraform Files and the OpenTofu Files, there is no question about whether they contain portions copied from

HashiCorp's original files. Indeed, OpenTofu's own versions of the files include attribution acknowledging authorship and copyright ownership by HashiCorp.

Removal of HashiCorp's proprietary notices, and purporting to grant broader rights than granted under the BSL, violates the terms of the license. Pursuant to the terms of the BSL, any such violation "automatically terminate[s] your rights under the[e] License for the current and all other versions of the Licensed Work." Use, modification, reproduction, or distribution in violation of the license constitutes a breach of contract and given our repeated notices of the importance of respecting HashiCorp's intellectual property rights and compliance with our license terms, a willful infringement. By promulgating code which contains unlicensed and infringing copies of HashiCorp code, OpenTofu is also creating breach and infringement exposure for all downstream recipients of the offending OpenTofu files. OpenTofu is acting unlawfully by distributing HashiCorp code in flagrant violation of our license terms and intellectual property rights, free-riding on HashiCorp's effort and investment in innovation, and disrupting the legitimate market for HashiCorp's commercial licenses.

In light of these instances of infringing reproductions and distributions of HashiCorp copyrighted code, we are concurrently sending DMCA takedown notices to Github to ensure the offending materials are removed, and any repeat infringers' accounts are disabled.

C. HashiCorp's Demands

As is clear from the foregoing, the OpenTofu project — through its various sponsors –has breached and facilitated breaches HashiCorp's BSL license terms and as such, has engaged in and facilitated unauthorized and infringing use of HashiCorp code. It has been on notice for several months as to the importance of compliance with, and seriousness of violating, our license terms, and yet continues to ignore the issue and our communications. To the extent OpenTofu's sponsors have failed to obtain legal advice, or ignored out prior letters, it is time to act.

Notwithstanding any takedowns that may be effected as a result of our submission of DMCA takedown notices to Github, the OpenTofu project must immediately cease and desist from any further use of any and all HashiCorp BSL code, including the files identified in this letter, and it must cease and desist from any future license violations.

We demand that OpenTofu respond to the following demands in writing, no later than ***Wednesday, April 10, 2024***:

- Confirm that the OpenTofu project has taken down from all repos (public or private), and is not using, any HashiCorp code that is subject only to the BSL license, including those instances identified in this letter, or that OpenTofu has otherwise used;

- Confirm that the OpenTofu project will not use HashiCorp code subject to the BSL license in many matter that breaches those terms in the future;

- Confirm that the OpenTofu project will reject any pending or future pull requests containing BSL licensed code, and make a clear and unambiguous statement to its

community that BSL licensed code cannot and will not be accepted into OpenTofu's MPL licensed repositories;

- Confirm that the OpenTofu project has appointed a monitor or liaison to review code submissions to the project for any indication that such submissions may include BSL licensed code originating from HashiCorp, and reject any such submissions;

- Properly educate contributors about the importance of respecting third party intellectual property rights and license terms, and the impermissibility of incorporating BSL: licensed code into MPL licensed codebases;

- Confirm that the OpenTofu project will not falsely mislabel HashiCorp code subject to the BSL license as being subject to any other license terms; and

- Confirm that the OpenTofu project and its agents and supporters will preserve, and not delete, any code, documents, communications, and other materials reasonably responsive to the subject matter of this dispute.

HashiCorp reserves all rights and will not hesitate to initiate litigation if necessary to preserve its rights. The OpenTofu project and its sponsors and others associated with it must take steps to preserve, and not delete or destroy, all documents and materials that could bear in any fashion on these issues.

We anticipate your cooperation and look forward to your prompt response.

Sincerely,

[redacted]

cc: [redacted]

```
// Copyright (c) The OpenTofu Authors
// SPDX-License-Identifier: MPL-2.0
// Copyright (c) 2023 HashiCorp, Inc.
// SPDX-License-Identifier: BUSL-1.1MPL-2.0

package refactoring

import (
	"fmt"

	"github.com/hashicorp/hcl/v2"
	"github.com/hashicorp/terraformopentofu/opentofu/internal/addrs"
	"github.com/hashicorp/terraformopentofu/opentofu/internal/configs"
	"github.com/hashicorp/terraformopentofu/opentofu/internal/tfdiags"
)

// RemoveStatement is the fully-specified form of addrs.Remove
type RemoveStatement struct {
	// From is the absolute address of the configuration object being removed.
	From      addrs.ConfigMoveableConfigRemovable
	// Destroy indicates that the resource should be destroyed, not just removed
	// from state.
	Destroy   bool
	DeclRange tfdiags.SourceRange
}

// FindRemoveStatementsGetEndpointsToRemove recurses through the modules of the given configuration
// and returns a setan array of all "removed" blocks definedaddresses within after deduplication, in a
// on the From addressdeterministic but undefined order.
//
// Error diagnostics are returned if any resource or module targeted by a remove
// block is still defined inWe also validate that the removed modules/resources configuration blocks were removed.
//
// A "removed" block in a parent module overrides a removed block in a child
// module when both target the same configuration object.
func FindRemoveStatementsGetEndpointsToRemove(rootCfg *configs.Config) ([]addrs.Map[addrs.ConfigMoveable,
RemoveStatement]ConfigRemovable, tfdiags.Diagnostics) {
	stmtsrm := findRemoveStatements(rootCfg, addrs.MakeMap[addrs.ConfigMoveable, RemoveStatement](nil))
	diags := validateRemoveStatements(rootCfg, stmtsrm)
	removedAddresses := make([]addrs.ConfigRemovable, len(rm))
	for i, rs := range rm {
		removedAddresses[i] = rs.From
	}
	return stmtsremovedAddresses, diags
}

func findRemoveStatements(cfg *configs.Config, into []*RemoveStatement) []*RemoveStatement {
	modAddr := cfg.Path
```

```
		for _, rc := range cfg.Module.Removed {
			var removedEndpoint *RemoveStatement
			switch FromAddress := rc.From.RelSubject.(type) {
			case addrs.ConfigResource:
				// Get the absolute address of the resource by appending the module config address
				// to the resource's relative address
				absModule := make(addrs.Module, 0, len(modAddr)+len(FromAddress.Module))
				absModule = append(absModule, modAddr...)
				absModule = append(absModule, FromAddress.Module...)

				var absConfigResource addrs.ConfigRemovable = addrs.ConfigResource{
					Resource: FromAddress.Resource,
					Module:   absModule,
				}

				removedEndpoint = &RemoveStatement{From: absConfigResource, DeclRange:
tfdiags.SourceRangeFromHCL(rc.DeclRange)}

			case addrs.Module:
				// Get the absolute address of the module by appending the module config address
				// to the module itself
				var absModule = make(addrs.Module, 0, len(modAddr)+len(FromAddress))
				absModule = append(absModule, modAddr...)
				absModule = append(absModule, FromAddress...)
				removedEndpoint = &RemoveStatement{From: absModule, DeclRange:
tfdiags.SourceRangeFromHCL(rc.DeclRange)}

			default:
				panic(fmt.Sprintf("unhandled address type %T", FromAddress))
			}

			into = append(into, removedEndpoint)

		}

		for _, childCfg := range cfg.Children {
			into = findRemoveStatements(childCfg, into)
		}

		return into
}

// validateRemoveStatements validates that the removed modules/resources configuration blocks were removed.
func validateRemoveStatements(cfg *configs.Config, stmts addrs.Map[addrs.ConfigMoveable, removeStatements
[]*RemoveStatement]) (diags tfdiags.Diagnostics) {
		var diags tfdiags.Diagnostics

		for _, rstrs := range stmts.Keys()removeStatements {
			fromAddr := rs.From
```

```
		if fromAddr == nil {
			// Invalid value should've been caught during original
			// configuration decoding, in the configs package.
			panic(fmt.Sprintf("incompatible Remove endpoint in %s", rs.DeclRange.ToHCL()))
		}

		// validate that a resource/module with this address doesn't exist in the config
		switch rstfromAddr := rstfromAddr.(type) {
		case addrs.ConfigResource:
			mmoduleConfig := cfg.Descendent(rstfromAddr.Module)
			if m == nil {
				break
			}
			if r :moduleConfig != mnil && moduleConfig.Module.ResourceByAddr(rstfromAddr.Resource); r != nil {
				diags = diags.Append(&hcl.Diagnostic{
					Severity: hcl.DiagError,
					Summary:  "Removed resource block still exists",
					Detail: fmt.Sprintf(
					Detail:   fmt.Sprintf(     "This statement declares that %s was removed, but it is still declared ina removal of the resource %s, but this resource block still exists in the configuration. Please remove the resource block.", rst),
						fromAddr,
					),
					Subject: rrs.DeclRange.ToHCL().Ptr(),
				})
			}
		case addrs.Module:
			if m := cfg.Descendent(rstfromAddr); m != nil {
				diags = diags.Append(&hcl.Diagnostic{
					Severity: hcl.DiagError,
					Summary:  "Removed module block still exists",
					Detail: fmt.Sprintf(
					Detail:   fmt.Sprintf(     "This statement declares thata removal of the module %s was removed, but it is still declared inthis module block still exists in the configuration. Please remove the module block.", rst),
						fromAddr,
					),
					Subject: mrs.CallRangeDeclRange.ToHCL().Ptr(),
				})
			}
		}
	}
	return diags
}
func findRemoveStatements(cfg *configs.Config, into addrs.Map[addrs.ConfigMoveable, RemoveStatement]) addrs.Map[addrs.ConfigMoveable, RemoveStatement] {
	for _, mc := range cfg.Module.Removed {
		switch mc.From.ObjectKind() {
		case addrs.RemoveTargetResource:
			// First, stitch together the module path and the RelSubject to form
```

```
                        // the absolute address of the config object being removed.
                        res := mc.From.RelSubject.(addrs.ConfigResource)
                fromAddr := addrs.ConfigResource{
                                Module:   append(cfg.Path, res.Module...),
                                Resource: res.Resource,
                        }
                        // If we already have a remove statement for this ConfigResource, it
                        // must have come from a parent module, because duplicate removed
                        // blocks in the same module are ignored during parsing.
                        // The removed block in the parent module overrides the block in the
                        // child module.
                        existingStatement, ok := into.GetOk(fromAddr)
                        if ok {
                                if existingResource, ok := existingStatement.From.(addrs.ConfigResource); ok &&
                                        existingResource.Equal(fromAddr) {
                                        continue
                                }
                        }
                        into.Put(fromAddr, RemoveStatement{
                                From:      fromAddr,
                                Destroy:   mc.Destroy,
DeclRange: tfdiags.SourceRangeFromHCL(mc.DeclRange),
                        })
                case addrs.RemoveTargetModule:
                        // First, stitch together the module path and the RelSubject to form
                        // the absolute address of the config object being removed.
                        mod := mc.From.RelSubject.(addrs.Module)
                        absMod := append(cfg.Path, mod...)
                        // If there is already a statement for this Module, it must
                        // have come from a parent module, because duplicate removed blocks
                        // in the same module are ignored during parsing.
                        // The removed block in the parent module overrides the block in the
                        // child module.
                        existingStatement, ok := into.GetOk(mc.From.RelSubject)
                        if ok {
                                if existingModule, ok := existingStatement.From.(addrs.Module); ok &&
                                        existingModule.Equal(absMod) {
                                        continue
                                }
                        }
                        into.Put(absMod, RemoveStatement{
                                From:      absMod,
                                Destroy:   mc.Destroy,
DeclRange: tfdiags.SourceRangeFromHCL(mc.DeclRange),
                        })
                default:
                        panic("Unsupported remove target kind")fmt.Sprintf("incompatible Remove endpoint address type in
%s", rs.DeclRange.ToHCL()))
                }
        }
```

```
		for _, childCfg := range cfg.Children {
			into = findRemoveStatements(childCfg, into)
		}
		return intodiags
}
```

Summary report: Litera Compare for Word 11.5.0.74 Document comparison done on 3/28/2024 10:32:44 AM	
Style name: Default Style	
Intelligent Table Comparison: Active	
Original filename: TERRAFORM - remove statement.go.docx	
Modified filename: OPENTOFU - remove statement.go.docx	
Changes:	
Add	104
~~Delete~~	120
~~Move From~~	19
Move To	19
Table Insert	0
~~Table Delete~~	0
Table moves to	0
~~Table moves from~~	0
Embedded Graphics (Visio, ChemDraw, Images etc.)	0
Embedded Excel	0
Format changes	0
Total Changes:	262

```
// Copyright (c) The OpenTofu Authors
// SPDX-License-Identifier: MPL-2.0
// Copyright (c) 2023 HashiCorp, Inc.
// SPDX-License-Identifier: BUSL-1.1MPL-2.0

package addrs

import (
	"github.com/hashicorp/hcl/v2"
	"github.com/hashicorp/terraformopentofu/opentofu/internal/tfdiags"
)

// RemoveEndpoint is to ConfigRemovable what Target is to Targetable:
// Like MoveEndpoint, RemoveTarget is a wrapping struct that captures the result
//  of decoding an HCL
// traversal representing a relative path from the current module to
// module to a removeablea removable object. It is very similar to MoveEndpoint.
//
// Its purpose is to represent the "from" address in a "removed" block
// Remove targets are somewhat simpler than move endpoints, in that they deal
// only with resources and modules defined in configuration, not instances of
// those objects as recorded in state. We are therefore able to determine the
// ConfigMoveable up front, since specifying any resource or module instance key
// in a removed block is invalidthe configuration.
//
// An interesting quirk of RemoveTarget is that RelSubject denotes aTo obtain a full address from a RemoveEndpoint we
need to combine it
// with any ancestor modules in the configuration object that, if the removed block is valid, should no longer
// exist in configuration. This "last known address" is used to locate and delete
// the appropriate state objects, or, in the case in which the user has forgotten
// to remove the object from configuration, to report the address of that block
// in an error diagnostic.
type RemoveTargetRemoveEndpoint struct {
	// SourceRange is the location of the targetphysical endpoint address in configuration.
	// in configuration, if this RemoveEndpoint was decoded from a
	// configuration expression.
	SourceRange tfdiags.SourceRange

	// RelSubject, like MoveEndpoint's relSubject, abuses an absolute address
	// type to represent athe representation of our relative address. as a ConfigRemovable
	RelSubject ConfigMoveableConfigRemovable
}

// ParseRemoveEndpoint attempts to interpret the given traversal as a
// "remove endpoint" address, which is a relative path from the module containing
// the traversal to a removable object in either the same module or in some
func (t *RemoveTarget) ObjectKind() RemoveTargetKind {
	return removeTargetKind(t.RelSubject)
```

```
}
// child module.
func (t *RemoveTarget) String() string {
	if t.ObjectKind() == RemoveTargetModule {
		return t.RelSubject.(Module).String()
	} else if t.ObjectKind() == RemoveTargetResource {
		return t.RelSubject.(ConfigResource).String()
	}
//No other valid address types
	panic("Usupported remove target kind")
}
func (t *RemoveTarget) Equal(other *RemoveTarget) bool {
	switch {
	case (t == nil) != (other == nil):
		return false
	case t == nil:
		return true
	default:
// We can safely compare string representations, since the Subject is aThis deals only with the syntactic element of a
remove endpoint expression
// in configuration. Before the result will be useful you'll need to combine
// it with the address of the module where it was declared in order to get
// simple module or resourcean absolute address relative to the root module.
		return t.String() == other.String() && t.SourceRange == other.SourceRange
	}
}
func ParseRemoveTargetParseRemoveEndpoint(traversal hcl.Traversal) (*RemoveTargetRemoveEndpoint, tfdiags.Diagnostics) {
	path, remain, diags := parseModulePrefix(traversal)
	if diags.HasErrors() {
		return nil, diags
	}

	rng := tfdiags.SourceRangeFromHCL(traversal.SourceRange())

	if len(remain) == 0 {
		return &RemoveTargetRemoveEndpoint{
			RelSubject:  path,
			SourceRange: rng,
		}, diags
	}

	rAddrriAddr, moreDiags := parseConfigResourceUnderModuleparseResourceUnderModule(path, remain)
	diags = diags.Append(moreDiags)
	if diags.HasErrors() {
		return nil, diags
	}

	if rAddrriAddr.Resource.Mode == DataResourceMode {
		diags = diags.Append(&hcl.Diagnostic{
			Severity: hcl.DiagError,
```

```
                        Summary:  "Data source address is not allowed",
                        Detail:   "Data sources are nevercannot be destroyed, so they are not valid targets of removed blocksand therefore, 'removed' blocks are not allowed to target them. To remove data sources from the state, you should remove the data source from state, remove the data source block from the configuration.",
                        Subject:  rng.ToHCLtraversal.SourceRange().Ptr(),
                })

                return nil, diags
        }

        return &RemoveTargetRemoveEndpoint{
                RelSubject:  rAddrriAddr,
                SourceRange: rng,
        }, diags
}
```

Summary report: Litera Compare for Word 11.5.0.74 Document comparison done on 3/28/2024 10:40:41 AM	
Style name: Default Style	
Intelligent Table Comparison: Active	
Original filename: TERRAFORM - remove target.go.docx	
Modified filename: OPENTOFU - remove endpoint.go.docx	
Changes:	
Add	54
~~Delete~~	67
~~Move From~~	7
Move To	7
Table Insert	0
~~Table Delete~~	0
Table moves to	0
~~Table moves from~~	0
Embedded Graphics (Visio, ChemDraw, Images etc.)	0
Embedded Excel	0
Format changes	0
Total Changes:	135

```
// Copyright (c) The OpenTofu Authors
// SPDX-License-Identifier: MPL-2.0
// Copyright (c) 2023 HashiCorp, Inc.
// SPDX-License-Identifier: ~~BUSL-1.1~~MPL-2.0

package addrs

import (
	"testing"

	"github.com/google/go-cmp/cmp"
	"github.com/hashicorp/hcl/v2"
	"github.com/hashicorp/hcl/v2/hclsyntax"
)

func ~~TestParseRemoveTarget~~TestParseRemoveEndpoint(t *testing.T) {
	tests := []struct {
		Input   string
		~~Want      ConfigMoveable~~WantRel ConfigRemovable
		WantErr string
	}{
		{
			`~~test_instance~~foo.bar`,
			ConfigResource{
				Module: RootModule,
				Resource: Resource{

					Mode: ManagedResourceMode,
					Type: "~~test_instance~~foo",
					Name: "bar",
				},
			},
			``,
		},
		{
			`module.boop`,
			Module{"boop"},
			``,
		},
		{
			`module.boop.foo.~~test_instance.~~bar`,
			ConfigResource{
				Module: ~~[]string~~Module{"~~foo~~boop"},
				Resource: Resource{
					Mode: ManagedResourceMode,
					Type: "~~test_instance~~foo",
					Name: "bar",
				},
```

```
		},
		`,
	},
	{
		`module.foo.module.bar`,
		Module{"foo", "bar"},
		``,
	},
	{
		`module.fooboop.module.bazbip.test_instancefoo.bar`,
		ConfigResource{
			Module: []stringModule{"fooboop", "bazbip"},
			Resource: Resource{
				Mode: ManagedResourceMode,
				Type: "test_instancefoo",
				Name: "bar",
			},
		},
		`,
	},
	{
		`foo.bar[0]`,
		nil,
		`Resource instance address with keys is not allowed: Resource address cannot be a resource instance (e.g. "null_resource.a[0]"), it must be a resource instead (e.g. "null_resource.a").`,
	},
	{
		`foo.bar["a"]`,
		nil,
		`Resource instance address with keys is not allowed: Resource address cannot be a resource instance (e.g. "null_resource.a[0]"), it must be a resource instead (e.g. "null_resource.a").`,
	},
	{
		`data.test_ds.moomodule.boop.foo.bar[0]`,
		nil,
		`Data sourceResource instance address with keys is not allowed: Data sources are never destroyed, so they are not valid targets of removed blocks. To remove the data source from state, remove the data source block from configurationResource address cannot be a resource instance (e.g. "null_resource.a[0]"), it must be a resource instead (e.g. "null_resource.a").`,
	},
	{
		`module.boop.foo.data.test_ds.noobar["a"]`,
		nil,

		`Resource instance address with keys is not allowed: Resource address cannot be a resource instance (e.g. "null_resource.a[0]"), it must be a resource instead (e.g. "null_resource.a").`,
	},
	{
		`data.foo.bar`,
		nil,
```

```
                `Data source address is not allowed: Data sources are nevercannot be destroyed, so they are not valid targets of removed blocksand therefore, 'removed' blocks are not allowed to target them. To remove data sources from the state, you should remove the data source from state, remove the data source block from the configuration.`,
            },
            {
                `test_instancedata.foo.bar[0]`,
                nil,
                `Resource instance address with keys is not allowed: Resource address must be a resource (e.g. "test_instance.foo"), not acannot be a resource instance (e.g. "test_instance.foo[1]null resource.a[0]"), it must be a resource instead (e.g. "null_resource.a").`,
            },
            {
                `data.foo.bar["a"]`,
                nil,
                `Resource instance address with keys is not allowed: Resource address cannot be a resource instance (e.g. "null_resource.a[0]"), it must be a resource instead (e.g. "null_resource.a").`,
            },
            {
                `module.boop.data.foo.bar[0].test_instance.bar`,
                nil,
                `Resource instance address with keys is not allowed: Resource address cannot be a resource instance (e.g. "null_resource.a[0]"), it must be a resource instead (e.g. "null_resource.a").`,
            },
            {
                `module.boop.data.foo.bar["a"]`,
                nil,
                `Resource instance address with keys is not allowed: Resource address cannot be a resource instance (e.g. "null_resource.a[0]"), it must be a resource instead (e.g. "null_resource.a").`,
            },
            {
                `module.foo[0]`,
                nil,
                `Module instance address with keys is not allowed: Module address must be a module (e.g. "module.foo"), not acannot be a module instance (e.g. "module.fooa[10]"), it must be a module instead (e.g. "module.a").`,
            },
            {
                `module.foo["a"]`,
                nil,
                `Module instance address with keys is not allowed: Module address cannot be a module instance (e.g. "module.a[0]"), it must be a module instead (e.g. "module.a").`,
            },
            {
                `module.foo[1].module.bar`,
                nil,
                `Module instance address with keys is not allowed: Module address cannot be a module instance (e.g. "module.a[0]"), it must be a module instead (e.g. "module.a").`,
            },
            {
                `module.foo.test_instancemodule.bar[01]`,
```

```
                nil,
                `ResourceModule instance address with keys is not allowed: ResourceModule address must be a resource (e.g. "test_instance.foo"), not a resourcecannot be a module instance (e.g. "test_instance.foo[1]module.a[0]"), it must be a module instead (e.g. "module.a").`,
            },
            {
                `module.foo[0].module.bar[1]`,
                nil,
                `Module instance address with keys is not allowed: Module address cannot be a module instance (e.g. "module.a[0]"), it must be a module instead (e.g. "module.a").`,
            },
            {
                `module`,
                nil,
                `Invalid address operator: Prefix "module." must be followed by a module name.`,
            },
            {
                `module[0]`,
                nil,
                `Invalid address operator: Prefix "module." must be followed by a module name.`,
            },
            {
                `module.foo.data`,
                nil,
                `Invalid address: Resource specification must include a resource type and name.`,
            },
            {
                `module.foo.data.bar`,
                nil,
                `Invalid address: Resource specification must include a resource type and name.`,
            },
            {
                `module.foo.data[0]`,
                nil,
                `Invalid address: Resource specification must include a resource type and name.`,
            },
            {
                `module.foo.data.bar[0]`,
                nil,
                `Invalid address: A resource name is required.`,
            },
            {
                `module.foo.bar`,
                nil,
                `Invalid address: Resource specification must include a resource type and name.`,
            },
            {
                `module.foo.bar[0]`,
                nil,
                `Invalid address: A resource name is required.`,
```

```
		},
	}

	for _, test := range tests {
		t.Run(test.Input, func(t *testing.T) {
			traversal, hclDiags := hclsyntax.ParseTraversalAbs([]byte(test.Input), "", hcl.InitialPos)
			if hclDiags.HasErrors() {
				// We're not trying to test the HCL parser here, so any
				// failures at this point are likely to be bugs in the
				// test case itself.
				t.Fatalf("syntax error: %s", hclDiags.Error())
			}

			remTmoveEp, diags := ParseRemoveTargetParseRemoveEndpoint(traversal)

			switch {
			case test.WantErr != "":
				if !diags.HasErrors() {
					t.Fatalf("unexpected success\nwant error: %s", test.WantErr)
				}
				gotErr := diags.Err().Error()
				if gotErr != test.WantErr {
					t.Fatalf("wrong error\ngot:  %s\nwant: %s", gotErr, test.WantErr)
				}
			default:
				if diags.HasErrors() {
					t.Fatalf("unexpected error: %s", diags.Err().Error())
				}
				if diff := cmp.Diff(test.Want, remTWantRel, moveEp.RelSubject); diff != "" {
					t.Errorf("wrong result\n%s", diff)
				}
			}
		})
	}
}
```

Summary report: **Litera Compare for Word 11.5.0.74 Document comparison done on 3/28/2024 10:41:58 AM**	
Style name: Default Style	
Intelligent Table Comparison: Active	
Original filename: TERRAFORM - remove target test.go.docx	
Modified filename: OPENTOFU - remove endpoint test.go.docx	
Changes:	
Add	156
~~Delete~~	43
~~Move From~~	5
Move To	5
Table Insert	0
~~Table Delete~~	0
Table moves to	0
~~Table moves from~~	0
Embedded Graphics (Visio, ChemDraw, Images etc.)	0
Embedded Excel	0
Format changes	0
Total Changes:	209

```
// Copyright (c) The OpenTofu Authors
// SPDX-License-Identifier: MPL-2.0
// Copyright (c) 2023 HashiCorp, Inc.
// SPDX-License-Identifier: BUSL-1.1MPL-2.0

package configs

import (
	"github.com/hashicorp/terraform/internal/addrs"
	"github.com/hashicorp/hcl/v2"
	"github.com/hashicorp/hcl/v2/gohclopentofu/opentofu/internal/addrs"
)

// Removed describes the contents of a "represents a removed" block in the configuration.
type Removed struct {
	// From is the address of the configuration object being removed.
	From *addrs.RemoveTargetRemoveEndpoint

	// Destroy indicates that the resource should be destroyed, not just removed
	// from state. Defaults to true.
	Destroy bool
	DeclRange hcl.Range
}

func decodeRemovedBlock(block *hcl.Block) (*Removed, hcl.Diagnostics) {
	var diags hcl.Diagnostics
	removed := &Removed{
		DeclRange: block.DefRange,
	}

	content, moreDiags := block.Body.Content(removedBlockSchema)
	diags = append(diags, moreDiags...)

	if attr, exists := content.Attributes["from"]; exists {
		from, traversalDiags := hcl.AbsTraversalForExpr(attr.Expr)
		diags = append(diags, traversalDiags...)
		if !traversalDiags.HasErrors() {
			from, fromDiags := addrs.ParseRemoveTargetParseRemoveEndpoint(from)
			diags = append(diags, fromDiags.ToHCL()...)
			removed.From = from
		}
	}

	removed.Destroy = true
	for _, block := range content.Blocks {
		switch block.Type {
		case "lifecycle":
			lcContent, lcDiags := block.Body.Content(removedLifecycleBlockSchema)
```

```
                        diags = append(diags, lcDiags...)
                        if attr, exists := lcContent.Attributes["destroy"]; exists {
                                valDiags := gohcl.DecodeExpression(attr.Expr, nil, &removed.Destroy)
                                diags = append(diags, valDiags...)
                        }
                }
        }
        return removed, diags
}

var removedBlockSchema = &hcl.BodySchema{
        Attributes: []hcl.AttributeSchema{
                {
                        Name:     "from",
                        Required: true,
                },
        },
        Blocks: []hcl.BlockHeaderSchema{
                {
                        Type: "lifecycle",
                },
        },
}

var removedLifecycleBlockSchema = &hcl.BodySchema{
        Attributes: []hcl.AttributeSchema{
                {
                        Name: "destroy",
                },
        },
}
```

Summary report: Litera Compare for Word 11.5.0.74 Document comparison done on 3/28/2024 10:42:51 AM	
Style name: Default Style	
Intelligent Table Comparison: Active	
Original filename: TERRAFORM - removed.go.docx	
Modified filename: OPENTOFU - removed.go.docx	
Changes:	
Add	10
~~Delete~~	37
~~Move From~~	1
Move To	1
Table Insert	0
~~Table Delete~~	0
Table moves to	0
~~Table moves from~~	0
Embedded Graphics (Visio, ChemDraw, Images etc.)	0
Embedded Excel	0
Format changes	0
Total Changes:	49

TERRAFORM - removed test.go - As of March 28th, 2024

```
// Copyright (c) HashiCorp, Inc.
// SPDX-License-Identifier: BUSL-1.1MPL-2.0

package configs

import (
	"testing"

	"github.com/hashicorp/hclgoogle/go-cmp/v2cmp"
	"github.com/hashicorp/hcl/v2/hcltest"
	"github.com/hashicorp/terraform/internal/addrshcl/v2/hcltest"
	"github.com/google/go-cmp/cmp"
	"github.com/zclconf/go-cty/ctyopentofu/opentofu/internal/addrs"
)

func TestRemovedBlock_decode(t *testing.T) {
	blockRange := hcl.Range{
		Filename: "mock.tf",
		Start:    hcl.Pos{Line: 3, Column: 12, Byte: 27},
		End:      hcl.Pos{Line: 3, Column: 19, Byte: 34},
	}

	foo_expr := hcltest.MockExprTraversalSrc("test_instance.foo")
	mod_foo_expr := hcltest.MockExprTraversalSrc("module.foo")
	foo_index_expr := hcltest.MockExprTraversalSrc("test_instance.foo[1]")
	mod_foo_exprmod_boop_index_foo_expr := hcltest.MockExprTraversalSrc("module.boop[1].test_instance.foo")
	mod_foo_index_exprdata_foo_expr := hcltest.MockExprTraversalSrc("moduledata.test_instance.foo[1]")

	tests := map[string]struct {
		input *hcl.Block
		want  *Removed
		err   string
	}{
		"destroy truesuccess": {
			&hcl.Block{
				Type: "removed",
				Body: hcltest.MockBody(&hcl.BodyContent{
					Attributes: hcl.Attributes{
						"from": {
							Name: "from",
							Expr: foo_expr,
						},
					},
					Blocks: hcl.Blocks{
						&hcl.Block{
							Type: "lifecycle",
							Body: hcltest.MockBody(&hcl.BodyContent{
								Attributes: hcl.Attributes{
```

```
                                                "destroy": {
                                                        Name: "destroy",
                                                        Expr: hcltest.MockExprLiteral(cty.BoolVal(true)),
                                                },
                                        },
                                }),
                        },
                },
                        }),
                        DefRange: blockRange,
                },
                &Removed{
                        From:      mustRemoveEndpointFromExpr(foo_expr),
                        Destroy:   true,
                        DeclRange: blockRange,
                },
                `,
        },
        "destroy false": {
                &hcl.Block{
                        Type: "removed",
                        Body: hcltest.MockBody(&hcl.BodyContent{
                                Attributes: hcl.Attributes{
                                        "from": {
                                                Name: "from",
                                                Expr: foo_expr,
                                        },
                                },
                                Blocks: hcl.Blocks{
                                        &hcl.Block{
                                                Type: "lifecycle",
                                                Body: hcltest.MockBody(&hcl.BodyContent{
                                                        Attributes: hcl.Attributes{
                                                                "destroy": {
                                                                        Name: "destroy",
                                                                        Expr: hcltest.MockExprLiteral(cty.BoolVal(false)),
                                                                },
                                                        },
                                                }),
                                        },
                                },
                        }),
                        DefRange: blockRange,
                },
                &Removed{
                        From:      mustRemoveEndpointFromExpr(foo_expr),
                        Destroy:   false,
                        DeclRange: blockRange,
                },
                `,
```

```
},
"modules": {
	&hcl.Block{
		Type: "removed",
		Body: hcltest.MockBody(&hcl.BodyContent{
			Attributes: hcl.Attributes{
				"from": {
					Name: "from",
					Expr: mod_foo_expr,
				},
			},
			Blocks: hcl.Blocks{
				&hcl.Block{
					Type: "lifecycle",
					Body: hcltest.MockBody(&hcl.BodyContent{
						Attributes: hcl.Attributes{
							"destroy": {
								Name: "destroy",
								Expr: hcltest.MockExprLiteral(cty.BoolVal(true)),
							},
						},
					}),
				},
			},
		}),
		DefRange: blockRange,
	},
	&Removed{
		From:      mustRemoveEndpointFromExpr(mod_foo_expr),
		Destroy:   true,
		DeclRange: blockRange,
	},
	`,
},
// KEM Unspecified behaviour
"no lifecycle blockerror: missing argument": {
	&hcl.Block{
		Type: "removed",
		Body: hcltest.MockBody(&hcl.BodyContent{
			Attributes: hcl.Attributes{},
				"from": {
					Name: "from",
					Expr: foo_expr,
				},
			},
		}),
		DefRange: blockRange,
	},
	&Removed{
		From:      mustRemoveEndpointFromExpr(foo_expr),
```

```
                Destroy:   true,
                DeclRange: blockRange,
        },
        ``"Missing required argument",
},
"error: missing argumentindexed resources": {
        &hcl.Block{
                Type: "removed",
                Body: hcltest.MockBody(&hcl.BodyContent{
                        Blocks: hcl.Blocks{
                                &hcl.Block{
                                        Type: "lifecycle",
                                        Body: hcltest.MockBody(&hcl.BodyContent{
                        Attributes: hcl.Attributes{
                                "destroyfrom": {
                                        Name: "destroyfrom",
                                                        Expr: hcltest.MockExprLiteral(cty.BoolVal(true)),
                                                },
                                        },
                                        })Expr: foo_index_expr,
                                },
                        },
                }),
                DefRange: blockRange,
        },
        &Removed{
                Destroy:   true,
                DeclRange: blockRange,
        },
        "Missing required argumentResource instance address with keys is not allowed",
},
"error: indexed resource instancemodules": {
        &hcl.Block{
                Type: "removed",
                Body: hcltest.MockBody(&hcl.BodyContent{
                        Attributes: hcl.Attributes{
                                "from": {
                                        Name: "from",
                                        Expr: foo_index_exprmod_boop_index_foo_expr,
                                },
                        },
                        Blocks: hcl.Blocks{
                                &hcl.Block{
                                        Type: "lifecycle",
                                        Body: hcltest.MockBody(&hcl.BodyContent{
                                                Attributes: hcl.Attributes{
                                                        "destroy": {
                                                                Name: "destroy",
                                                                Expr: hcltest.MockExprLiteral(cty.BoolVal(true)),
                                                        },
```

```
                                        },
                                    }),
                                },
                            },
                        }),
                        DefRange: blockRange,
                    },
                    &Removed{
                        From:      nil,
                        Destroy:   true,
                        DeclRange: blockRange,
                    },
                    `Resource"Module instance address with keys is not allowed`",
                },
                "error: indexed module instancedata address": {
                    &hcl.Block{
                        Type: "removedmoved",
                        Body: hcltest.MockBody(&hcl.BodyContent{
                            Attributes: hcl.Attributes{
                                "from": {
                                    Name: "from",
                                    Expr: mod_foo_index_exprdata foo expr,
                                },
                            },
                            Blocks: hcl.Blocks{
                                &hcl.Block{
                                    Type: "lifecycle",
                                    Body: hcltest.MockBody(&hcl.BodyContent{
                                        Attributes: hcl.Attributes{
                                            "destroy": {
                                                Name: "destroy",
                                                Expr: hcltest.MockExprLiteral(cty.BoolVal(true)),
                                            },
                                        },
                                    }),
                                },
                            },
                        }),
                        DefRange: blockRange,
                    },
                    &Removed{
                        From:      nil,
                        Destroy:   true,
                        DeclRange: blockRange,
                    },
                    `Module instance keys"Data source address is not allowed`",
                },
            }

            for name, test := range tests {
```

```
		t.Run(name, func(t *testing.T) {
			got, diags := decodeRemovedBlock(test.input)

			if diags.HasErrors() {
				if test.err == "" {
					t.Fatalf("unexpected error: %s", diags.Errs())
				}
				if gotErr := diags[0].Summary; gotErr != test.err {
					t.Errorf("wrong error, got %q, want %q", gotErr, test.err)
				}
			} else if test.err != "" {
				t.Fatal("expected error")
			}

			if !cmp.Equal(got, test.want, cmp.AllowUnexported(addrs.MoveEndpoint{})) {
				t.Fatalf("wrong result: %s", cmp.Diff(got, test.want))
			}
		})
	}
}

func TestRemovedBlock_inModule(t *testing.T) {
	parser := NewParser(nil)
	mod, diags := parser.LoadConfigDir("testdata/valid-modules/removed-blocks")
	if diags.HasErrors() {
		t.Errorf("unexpected error: %s", diags.Error())
	}

	var got []string
	for _, mc := range mod.Removed {
		got = append(got, mc.From.RelSubject.String())
	}
	want := []string{
		`test.foo`,
		`test.foo`,
		`module.a`,
		`module.a`,
		`test.foo`,
		`test.boop`,
	}
	if diff := cmp.Diff(want, got); diff != "" {
		t.Errorf("wrong addresses\n%s", diff)
	}
}

func mustRemoveEndpointFromExpr(expr hcl.Expression) *addrs.RemoveTargetRemoveEndpoint {
	traversal, hcldiags := hcl.AbsTraversalForExpr(expr)
	if hcldiags.HasErrors() {
		panic(hcldiags.Errs())
	}
```

```
	ep, diags := addrs.ParseRemoveTargetParseRemoveEndpoint(traversal)
	if diags.HasErrors() {
		panic(diags.Err())
	}

	return ep
}
```

Summary report: Litera Compare for Word 11.5.0.74 Document comparison done on 3/28/2024 10:43:18 AM	
Style name: Default Style	
Intelligent Table Comparison: Active	
Original filename: TERRAFORM - removed test.go.docx	
Modified filename: OPENTOFU - removed test.go.docx	
Changes:	
Add	54
~~Delete~~	136
~~Move From~~	6
Move To	6
Table Insert	0
~~Table Delete~~	0
Table moves to	0
~~Table moves from~~	0
Embedded Graphics (Visio, ChemDraw, Images etc.)	0
Embedded Excel	0
Format changes	0
Total Changes:	202