

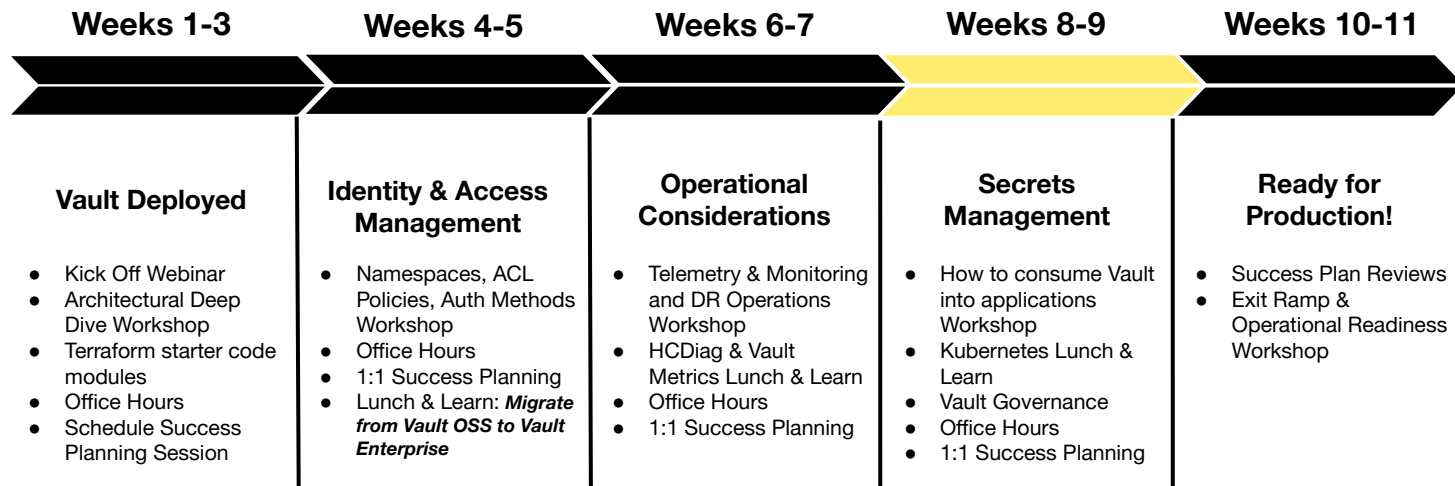


Vault Self-Managed Webinar - Vault Governance

January 2022

Copyright © 2021 HashiCorp

Vault Enterprise Path to Production





Agenda

- Sentinel
- Control Groups
- Quotas
- Next Steps
- Q & A

01

Sentinel

Overview



Sentinel is an embeddable ‘policy as code’ framework to enable fine-grained, logic-based policy decisions that can be extended to source external information to make decisions.

Sentinel policies are used in combination with your existing ACL policies and policy templates.

Working with Sentinel policies



- Vault injects data into the Sentinel runtime environment, including properties for: requests, replication, tokens, Identity secrets engine, MFA, and Control Groups
- Sentinel policies are written in a domain specific language
- Test policies offline with the Sentinel binary
- Manage policies via HTTP API, CLI, or web UI
- The root token or tokens with the **root** policy attached are **exempt** from Sentinel policies!



Sentinel policy structure

```
import "<library>"
<variable> = <value>

<name> = rule { <condition_to_evaluate> }
main = rule {
    <condition_to_evaluate>
}
```

CODE EDITOR

Example Sentinel policy



```
import "sockaddr"
import "strings"

# Only evaluated for update operations against transit/ path
precond = rule {
    request.operation in [ "update" ] and
    strings.has_prefix(request.path, "transit/")
}

# Requests must originate from our private IP range
cidrcheck = rule {
    sockaddr.is_contained(request.connection.remote_addr, "122.22.3.4/32")
}

# Check the precondition before executing the cidrcheck
main = rule when precond {
    cidrcheck
}
```


Policy types



- Sentinel allows you to write complex logic and use external information like client CIDR
- Two types **Endpoint Governing Policy** (EGP) & **Role Governing Policy** (RGP)
 - EGPs are applied to particular paths
 - RGPs are applied to tokens, Identity entities, or Identity groups
- [Enforcement levels](#): **advisory**, **soft-mandatory**, or **hard-mandatory**

Endpoint Governing Policies (EGPs)



<https://www.vaultproject.io/docs/enterprise/sentinel#policy-types>

- API: `/sys/policies/egp/`
- EGPs are tied to particular paths
- Access to as much information in the request as possible
- Can be tied to all authenticated and most unauthenticated paths
- Denote suffix with glob character (*) for example: `my-secret-path/*`
- Path of just * affects all authenticated and login requests



EGP example

**‘Break Glass’ policy
denies access when
token created prior
to specified time.**

```
import "time"

main = rule when not request.unauthenticated {
  time.load(token.creation_time).unix >
  time.load("2020-01-015T07:25:00Z").unix
}
```

Role Governing Policies (RGPs)



<https://www.vaultproject.io/docs/enterprise/sentinel#policy-types>

- API: `/sys/policies/rgp/`
- RGPs are tied to tokens, Identity entities, or Identity groups
- Access to a rich set of controls across many aspects of Vault

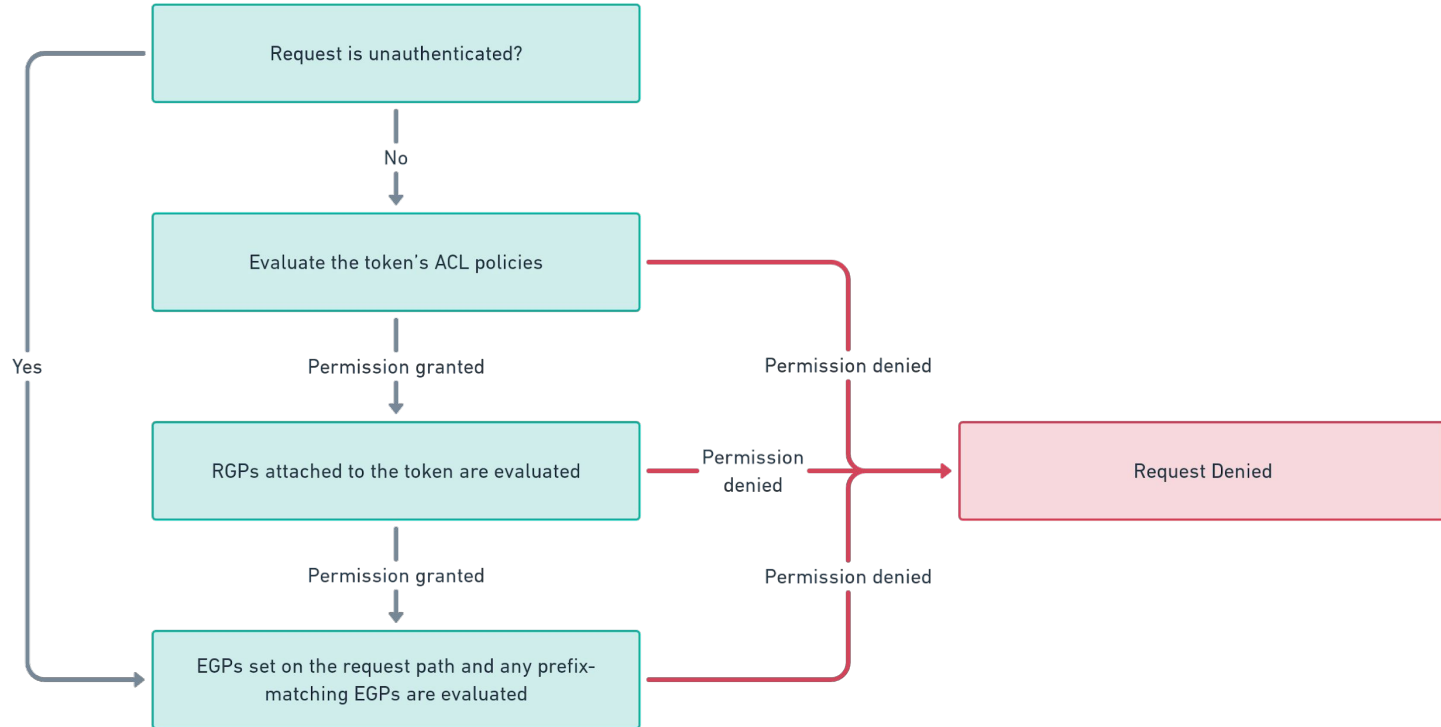


RGP example

Use available
Identity secrets
engine properties to
make decisions.

```
main = rule {  
    identity.entity.name is "vincent" or  
    identity.entity.id is  
    "fe2a5bfd-c483-9263-b0d4-f9d345c0ffee" or  
    "sysops" in identity.groups.names or  
    "c0ffee0a-5c07-4b97-81ec-0d423accb8e0" in  
    keys(identity.groups.by-id)  
}
```

Policy evaluation workflow



Test policies with Sentinel CLI



<https://docs.hashicorp.com/sentinel/downloads>

Sentinel

Intro

Docs

 Download

Download Sentinel

macOS

Windows

Linux

FreeBSD

NetBSD

OpenBSD

Solaris

MACOS BINARY DOWNLOAD

Sentinel 0.18.4

[64-bit](#)



Test cases 1/3

- Sentinel expects a `test/<policy_name>` folder with test case files in either HCL or JSON format.
- Test case files contain data to test the policy.

```
$ tree
```

```
├── cidr-check.sentinel
└── test
    ├── cidr-check
    │   ├── fail.hcl
    │   └── success.hcl
```




CODE EDITOR

```
global "my_global_variable" {  
  value = <test_data>  
},  
  
test {  
  rules = {  
    <expected_result>  
  }  
}
```

Test cases 2/3

Example test case

Specify the data to test the policy against.

Optional: Expected boolean value of the rules.

In absence of the 'test' block, all rules are expected to return true.

```
mock "time" {  
  data = {  
    now = {  
      weekday_name = "Monday"  
      hour          = 14  
    }  
  }  
}
```



Test cases 3/3

Use **mock** instead of **global** to inject static value directly into the policy's scope.

For example, if the `time` library is used in the policy, use **mock** to mock `time.now`.

```
import "sockaddr"
import "strings"

# Only evaluated for create, update, and delete operations against kv/ path
precond = rule {
    request.operation in ["create", "update", "delete"] and
    strings.has_prefix(request.path, "kv/")
}

# Requests must originate from our private IP range
cidrcheck = rule {
    sockaddr.is_contained(request.connection.remote_addr, "122.22.3.4/32")
}

# Check the precondition before executing the cidrcheck
main = rule when precond {
    cidrcheck
}
```

```
global "request" {  
  value = {  
    connection = {  
      remote_addr = "122.22.3.4"  
    }  
    operation = "create"  
    path = "kv/orders"  
  }  
}
```



Passing test

The file
`test/cidr-check/success.hcl` contains data for a
passing test.



Failing test

The file

`test/cidr-check/fail.hcl`

contains data for a failing test.

```
global "request" {
  value = {
    connection = {
      remote_addr = "122.22.3.10"
    }
    operation = "create"
    path = "kv/orders"
  }
}

test {
  rules = {
    main      = false
    precondition = true
  }
}
```

CODE EDITOR



Run tests

Use the `sentinel test` command to invoke the simulator and test policy.

TIP: Use `-verbose` flag to output additional traces and logs for failed tests.

```
$ sentinel test

PASS - cidr-check.sentinel
      PASS - test/cidr-check/success.hcl
      PASS - test/cidr-check/fail.hcl
```

A terminal window with a dark background and light gray text. The title bar at the top right says "TERMINAL". The command prompt shows the execution of `$ sentinel test`. The output consists of three lines of green text: `PASS - cidr-check.sentinel`, `PASS - test/cidr-check/success.hcl`, and `PASS - test/cidr-check/fail.hcl`.



Use CLI to deploy policy

Use `vault` CLI to write the policy.

When successfully written, Vault begins immediately enforcing the policy at the hard mandatory level.

TERMINAL

```
$ vault write sys/policies/egp/cidr-check \  
  policy=@cidr-check.sentinel \  
  enforcement_level="hard-mandatory" \  
  paths="kv/"
```



Success! Data written to: sys/policies/egp/cidr-check

02

Control Groups

Overview



Control groups allow for additional authorizations to be required for access to a path in Vault.

When a control group is defined, the following occurs:

- The requestor receives a wrapping token in return
- The authorizers required by the control group policy must approve the request
- Once all authorizations are satisfied, the requester can unwrap the secrets

Factors



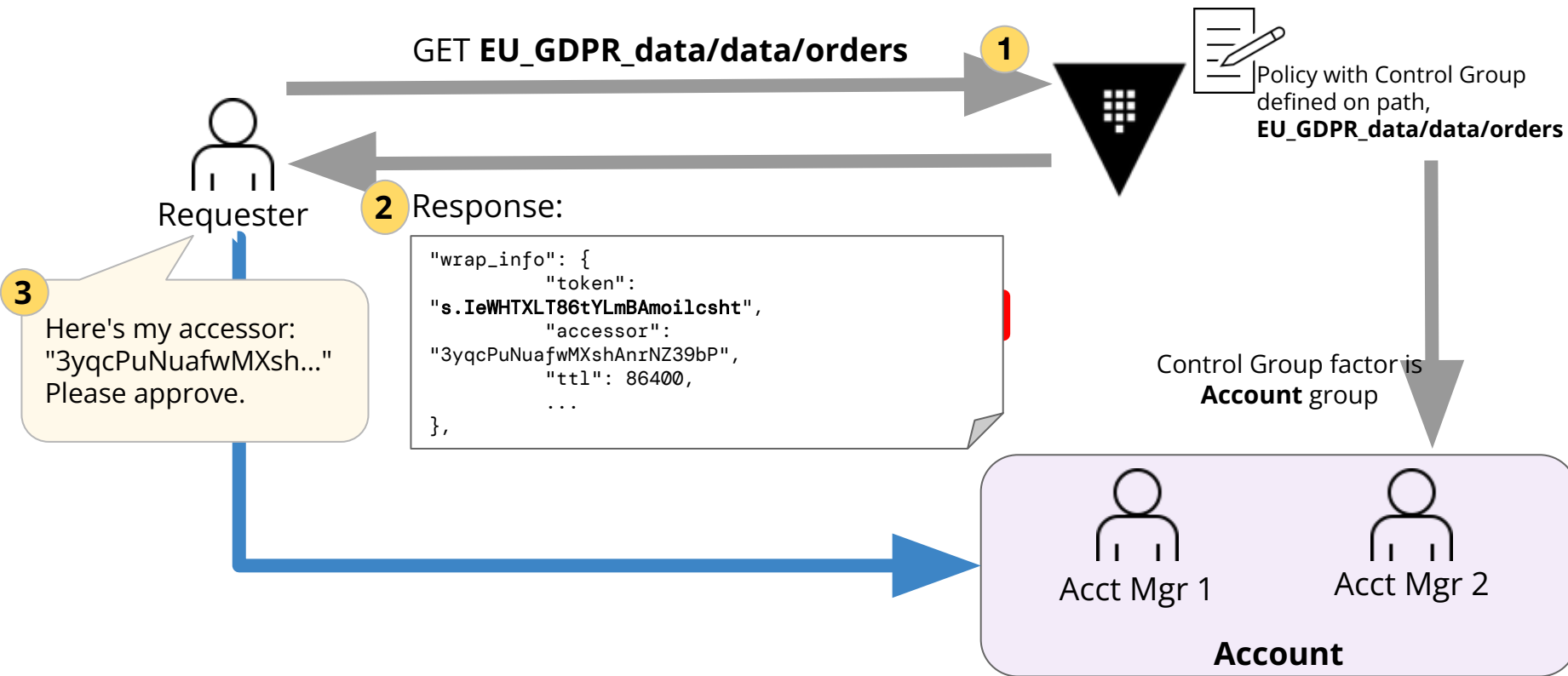
Control Group requirements on paths can be specified in:

- ACL policies
- Sentinel policies

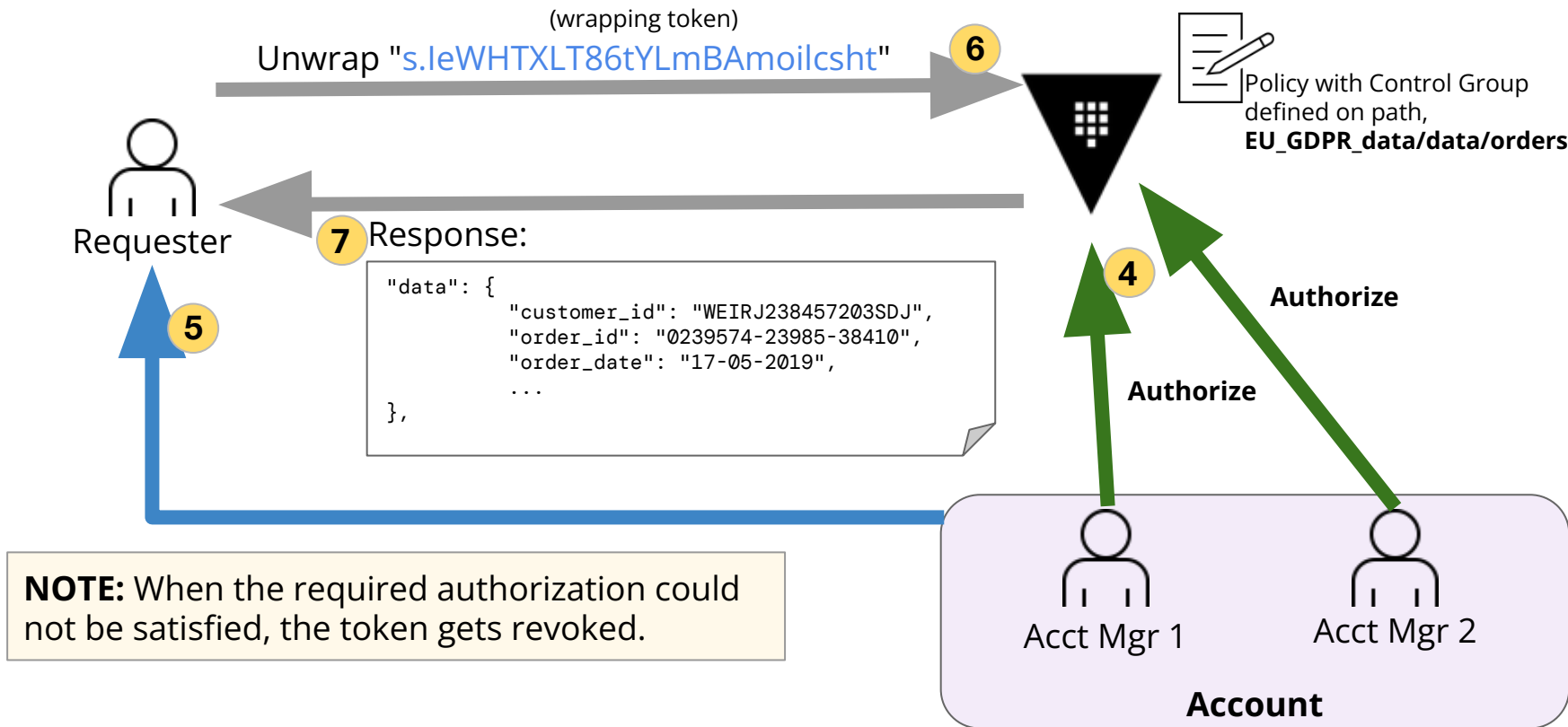
Currently supported Control Groups factor is Identity Groups

- An authorizer must belong to a specific Identity group (factor)

Control Groups Workflow (1/2)



Control Groups Workflow (2/2)





Control Groups in ACL policies



```
path "EU_GDPR_data/data/orders/*" {  
  capabilities = [ "read" ]  
  control_group = {  
    factor "acct_manager" {  
      identity {  
        group_names = [ "account" ]  
        approvals = 2  
      }  
    }  
  }  
}
```



Control Groups in Sentinel policies



```
import "controlgroup"
```

```
control_group = func() {  
    numAuthzs = 0  
    for controlgroup.authorizations as authz {  
        if "account" in authz.groups.by_name {  
            numAuthzs = numAuthzs + 1  
        }  
    }  
    if numAuthzs >= 2 {  
        return true  
    }  
    return false  
}
```

```
main = rule {  
    control_group()  
}
```

CODE EDITOR



Multiple factor Control Groups

```
CODE EDITOR

path "EU_GDPR_data/data/orders/*" {
  capabilities = [ "create", "read", "update" ]

  control_group = {
    factor "acct_manager" {
      ttl = "4h"
      identity {
        group_names = [ "account" ]
        approvals = 2
      }
    }
    factor "security" {
      identity {
        group_names = [ "eu-security" ]
        approvals = 1
      }
    }
  }
}
```



Control Groups workflow



Requester actions

```
> vault login -method=userpass username=bob password=training
```

Key	Value
Token	s.5VvDmKAFZyxZ3tfBAhVntanm
Token_policies	["default"]
Identity_policies	[" read-gdpr-order "]

```
> vault kv get EU_GDPR_data/orders/acct1
```

Key	Value
wrapping_token:	s.JwkWZ1sWICHfNaZSwDuMeF0A
wrapping_accessor:	HpsbV9ANBnxVlvig9Kvsot0B
wrapping_token_ttl:	24h
wrapping_token_creation_time:	2019-05-17 12:20:06 -0700 PDT
wrapping_token_creation_path:	EU_GDPR_data/data/orders/acct1

Authorizer actions



The screenshot shows the Kubernetes Authorizer web interface. At the top, there is a navigation bar with tabs for 'Secrets', 'Access', and 'Tools'. The 'Access' tab is selected. To the right of the tabs are three icons: a green status indicator with a dropdown arrow, a terminal icon with a dropdown arrow, and a user icon with a dropdown arrow. Below the navigation bar, the main content area is divided into two sections. On the left, under the heading 'ACCESS', there is a list of links: 'Leases' and 'Control Groups'. 'Control Groups' is highlighted with a blue underline. On the right, the 'Control Groups' section has a title 'Control Groups' and a descriptive paragraph: 'Control Groups add additional authorization factors to be required before satisfying a request. If you have a Control Group accessor, provide it here to view the lookup the authorization progress.' Below this text is a label 'Accessor' followed by a text input field containing the value 'HpsbV9ANBnxVlvig9Kvsot0B'. At the bottom of this section is a blue button labeled 'Lookup'.

Secrets Access Tools

Status

ACCESS

Leases

Control Groups

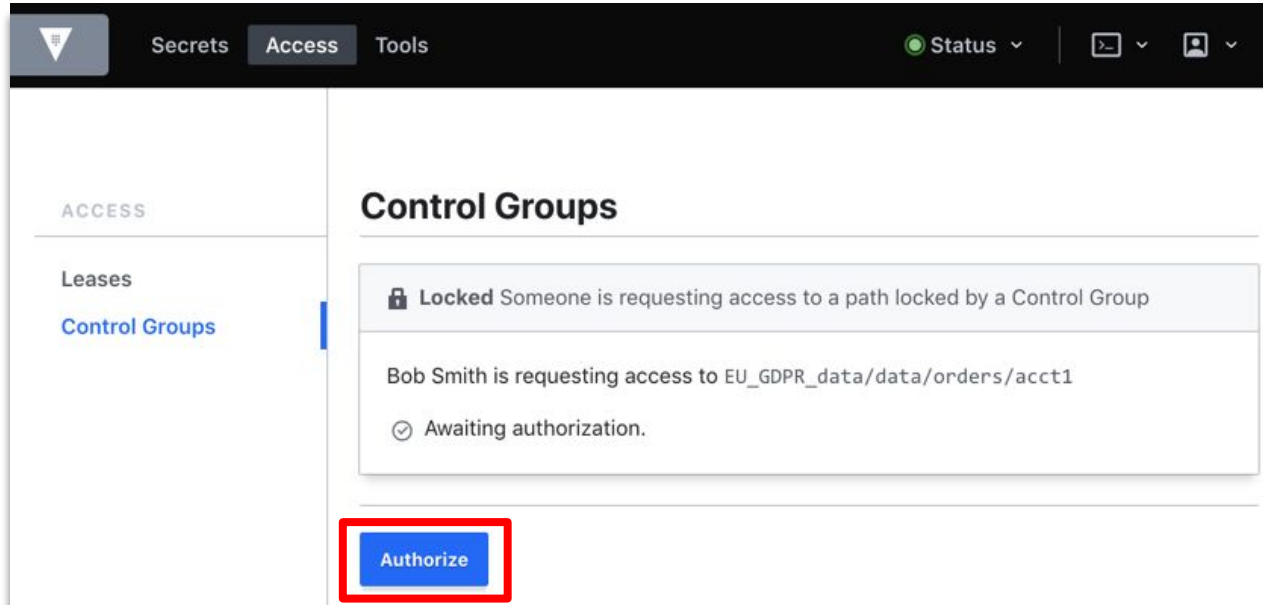
Control Groups

Control Groups add additional authorization factors to be required before satisfying a request. If you have a Control Group accessor, provide it here to view the lookup the authorization progress.

Accessor

Lookup

Authorizer actions



Authorizer actions



The screenshot shows a web interface for an authorizer. At the top, there is a dark navigation bar with a logo on the left, tabs for 'Secrets', 'Access' (which is active), and 'Tools' in the center, and a 'Status' indicator with a dropdown on the right. Below the navigation bar, the main content area is divided into two sections. On the left is a sidebar with the heading 'ACCESS' and two items: 'Leases' and 'Control Groups' (which is highlighted with a blue bar). The main section on the right is titled 'Control Groups'. It contains a green success message: '✓ Thanks! You have given authorization'. Below this, it states 'Bob Smith is authorized to access EU_GDPR_data/data/orders/acct1'. Underneath that, it says '✓ Already approved by Ellen Wright'. At the bottom of the main section, there is a '< Back' button.

Secrets Access Tools Status ▾

ACCESS

Leases

Control Groups

Control Groups

✓ Thanks! You have given authorization

Bob Smith is authorized to access EU_GDPR_data/data/orders/acct1

✓ Already approved by [Ellen Wright](#)

< Back

Unwrap the secret



The screenshot shows a web application interface for unwrapping data. At the top is a dark navigation bar with a logo on the left, tabs for 'Secrets', 'Access', and 'Tools', a 'Status' indicator with a dropdown arrow, and icons for a code editor and user profile. Below the navigation bar is a sidebar with a 'TOOLS' header and a list of options: 'Wrap', 'Lookup', 'Unwrap' (highlighted with a blue bar), 'Random', and 'Hash'. The main content area is titled 'Unwrap data' and contains a 'Wrapping token' label above a text input field containing the token 's.TQvZcyqQcRMxN8miKmSVZQbq'. Below the input field is a blue button labeled 'Unwrap data'.

Secrets Access Tools Status ▾

TOOLS

Wrap

Lookup

Unwrap

Random

Hash

Unwrap data

Wrapping token

s.TQvZcyqQcRMxN8miKmSVZQbq

Unwrap data

Unwrap the secret



The screenshot shows the 'Unwrap data' tool interface. The top navigation bar includes 'Secrets', 'Access', and 'Tools', along with a 'Status' indicator and icons for document and user. The left sidebar lists 'TOOLS' with options: 'Wrap', 'Lookup', 'Unwrap' (highlighted), 'Random', and 'Hash'. The main area is titled 'Unwrap data' and has two tabs: 'Data' (active) and 'Wrap Details'. The 'Data' tab displays a JSON object in a dark-themed editor with line numbers 1 through 12. Below the editor are 'Copy' and 'Back' buttons.

```
1 {  
2   "data": {  
3     "order_number": "12345678",  
4     "product_id": "987654321"  
5   },  
6   "metadata": {  
7     "created_time": "2019-05-17T19:15:29.719792Z",  
8     "deletion_time": "",  
9     "destroyed": false,  
10    "version": 2  
11  }  
12 }
```

Copy Back

Poll time



Do you foresee any secret requiring additional authorization prior to being read?

03

Quotas

Overview



Protect system stability and network, storage resource consumption from runaway application behavior and Distributed Denial of Service (DDoS)

- Rate Limit Quotas
Limit maximum amount of requests per second (RPS) to a system or mount to protect network bandwidth.
- Lease Count Quotas
Cap number of leases generated in a system or mount to protect system stability and storage performance at scale.





Configuring Resource Quotas

Creating Resource Quotas:

To set rate limit or lease quotas, set `/sys/quotas/<type>` with possible types being:

- `rate-limit`: Rate limit quota.
- `lease-count`: Maximum lease count (Enterprise only).

Configuring Resource Quotas

- Set/list specific types of quotas from `/sys/quotas/<type>/<name>`
- Path parameter can be set to a mount, mount in the namespace, or omitted for a systemwide quota.
- Different quotas have different parameters; see the documentation for more details.

Logging



If a request is rejected due to a 'Lease Count Quota' violation, Vault will record this in the audit log.

Violations of 'Rate Limit Quotas' are not logged to the audit log.

It is possible to enable traceability of 'Rate Limit Quotas' via the HTTP API endpoint: `sys/quotas/config`, changing the parameter:

`'enable_rate_limit_audit_logging'` to **true** (by default it will be false).

Enabling this can potentially impact performance if the request volume is large.



Rate Limit Quotas Example

Protect Vault from potential
DDoS attack

```
TERMINAL

# Create global quota rule and set rate at desired requests
per second
> vault write sys/quotas/rate-limit/global-rate rate=500

# Set rate limit on specific paths
> vault write sys/quotas/rate-limit/db-creds rate=30
    path="database"

# Set rate limit on specific namespace and path
> vault write sys/quotas/rate-limit/orders \
    path="us-west/kv-v2" \
    rate=16.67 \
    burst=100
```

04

Next Steps



Lease Count Quotas Example

Prevent the storage backend
to become the point of
failure

TERMINAL

```
> vault write sys/quotas/lease-count/global-count-limit \  
    max_leases=500  
  
> vault write sys/quotas/lease-count/db-creds \  
    max_leases=100  
    path="us-west/postgres"
```

<


>

HashiCorp | Discuss

Sign in





Q

≡



Information on HashiCorp Cloud Platform (HCP) use cases, Q&A, and best practices discussions. Please note that offerings may be in various release lifecycles. Categorize your question or comment under [HCP Consul](#) or [HCP Vault](#) subcategories. If your question or comment relates to networking, access control, or billing, post in this category. For support requests, please [open a ticket](#).

HashiCorp Cloud Platform (HCP) ▾ All ▾ Tags ▾ | Latest Top

Topic	Replies	Views	Activity
About the HashiCorp Cloud Platform (HCP) category			
<div>HashiCorp Cloud Platform (HCP)</div> <div>Information on HashiCorp Cloud Platform (HCP) use cases, Q&A, and best practices discussions. Please note that offerings may be in various release lifecycles. Categorize your question or comment under HCP Consul or HCP V... read more</div>	 1	387	May 24
<div>HCP Vault “per-client” pricing</div> <div>HCP Vault</div>	 0	39	9d
<div>Failing to use HCP Consul as my terraform backend</div> <div>HashiCorp Cloud Platform (HCP)</div>	 1	72	12d
<div>Does HCP support Automation APIs in AWS</div> <div>HashiCorp Cloud Platform (HCP) vault</div>	 0	73	27d



Discuss

Engage with the HashiCorp Cloud community including HashiCorp Architects and Engineers.

discuss.hashicorp.com

Learn

Step-by-step guides to accelerate deployment of Vault



The screenshot shows a web browser window displaying the HashiCorp Learn website. The page is titled "Deploy Cluster with Integrated Storage" and is part of a tutorial series. The left sidebar contains a navigation menu with sections: "GET STARTED" (including CLI Quick Start, HCP Vault, and UI Quick Start), "USE CASES" (including ADP, Data Encryption, and Secrets Management), and "CERTIFICATION PREP". The main content area features the tutorial title, a duration of 2 HR 15 MIN, and 12 TUTORIALS. A brief description states: "If you are responsible for setting up and maintaining a Vault cluster using integrated storage as a persistence layer, get started here." Below this, a card highlights a "15 MIN" tutorial titled "Vault with Integrated Storage Reference Architecture", which describes architectural best practices for implementing Vault using the Integrated Storage (Ref) storage backend. The top of the page includes the HashiCorp Learn logo, a search bar, and a "Sign in" button. The top right corner has links for "Docs" and "Forum".

HashiCorp Learn Browse tutorials ▾

Search

Sign in

Docs Forum

Vault

GET STARTED

- CLI Quick Start
- HCP Vault
- UI Quick Start

USE CASES

- ADP
- Data Encryption
- Secrets Management

CERTIFICATION PREP

Deploy Cluster with Integrated Storage

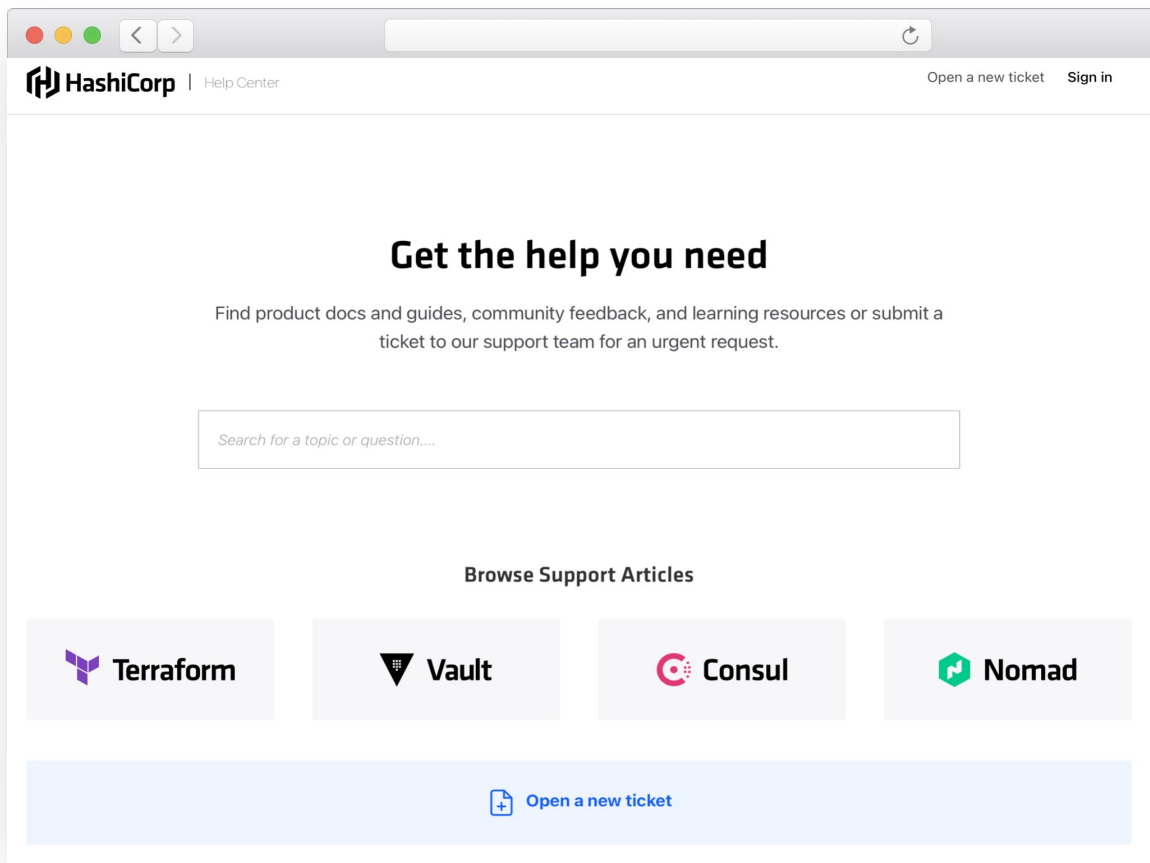
🕒 2 HR 15 MIN 📄 12 TUTORIALS

If you are responsible for setting up and maintaining a Vault cluster using integrated storage as a persistence layer, get started here.

15 MIN

Vault with Integrated Storage Reference Architecture

This guide describes architectural best practices for implementing Vault using the Integrated Storage (Ref) storage backend.



Support

<https://support.hashicorp.com>

Resources

Links mentioned during the webinar



- [Sentinel Policy Examples](#)
- [Standard Sentinel Imports](#)
- [List of Vault data available for Sentinel policies](#)
- [Sentinel Policy HashiCorp Learn Example](#)

05

Q & A



Thank You

customer.success@hashicorp.com

www.hashicorp.com