ZABBIX 20 YEARS

Securing your Zabbix database credentials with an external secret vault

Aleksandrs Petrovs-Gavrilovs

Technical training administrator

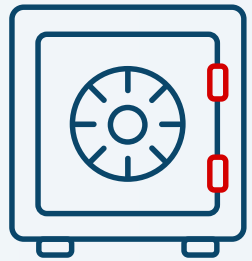
Zabbix database and vault

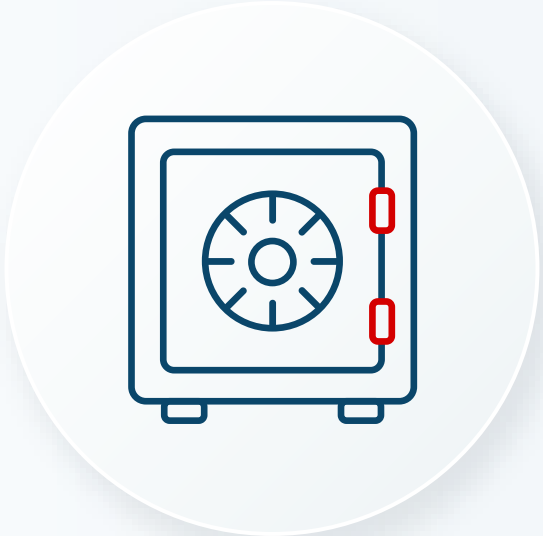
Zabbix database

- Is a critically important part of Zabbix installation
- Stores all the collected data
  - History, trends, events, alerts, etc.
- Stores all the created configuration
  - Templates, hosts, items, triggers, dashboards, etc.
- Including user macro and secret macro values
  - Links, usernames, passwords, tokens, etc.

External Vault

- Securely stores sensitive information
  - Including credentials used for database access
- Token and security policy-based access
  - Zabbix components can have different access levels
  - I.e. Zabbix server to macro values, Zabbix proxy to DB credentials
- Built-in support for secret revocation
  - Assists in key rolling and locking down in the case of an intrusion.





Vault support and communications

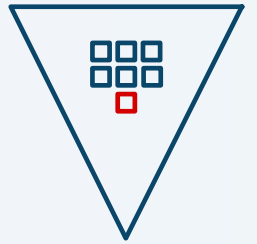
External vault support

Zabbix 7.0 supports two external vaults:

- HashiCorp vault
  - By default, the vault listens on TCP port 8200
  - Use the HTTPS protocol to communicate with the vault

- Zabbix uses HashiCorp KV secret engine v2
  - Includes secret versioning and metadata
  - Managed using the GUI interface or CLI tools

- Macro values use the following path
  - <secret_engine>/<secret_path>/macro:value



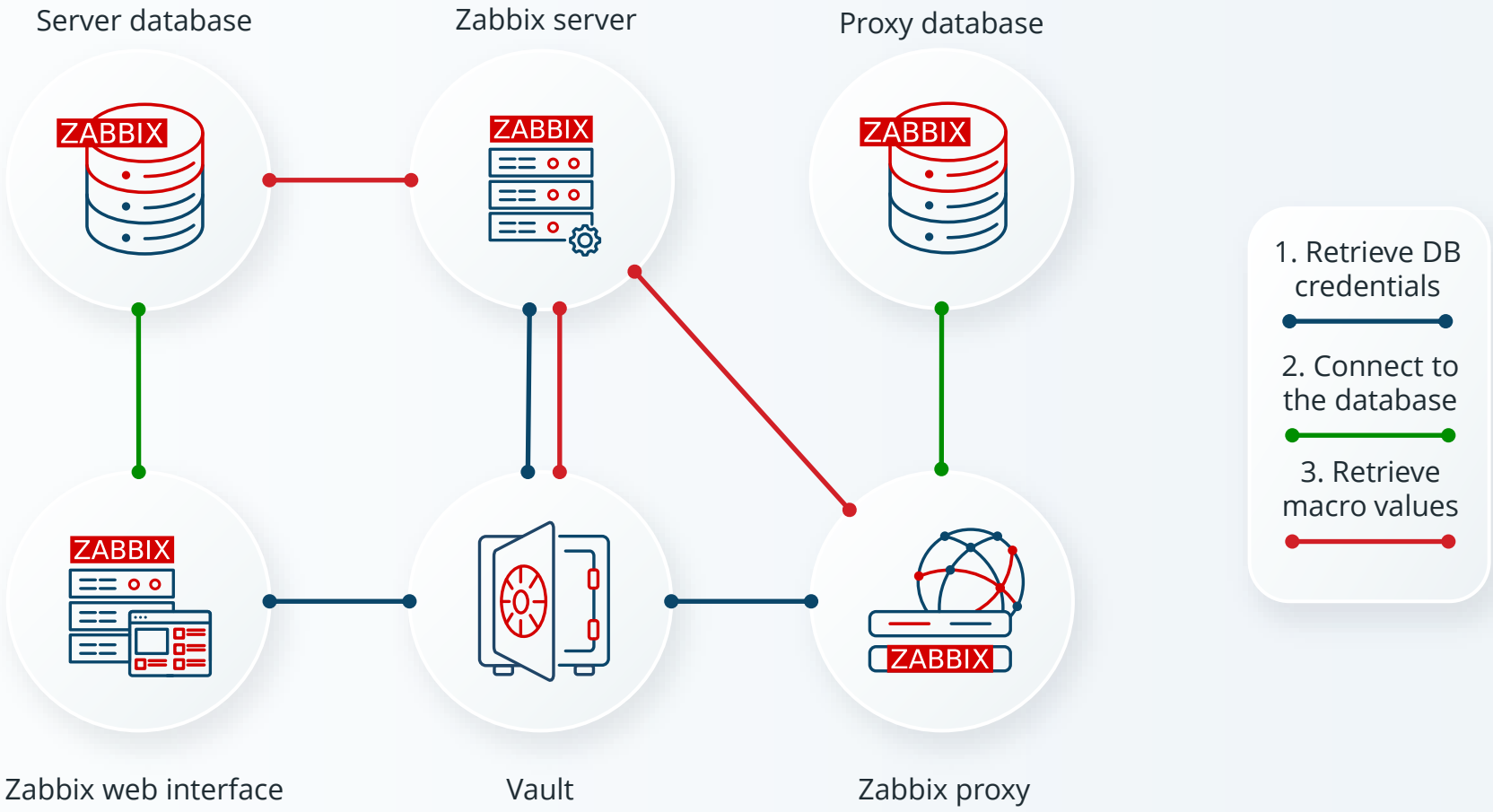
External vault support

Zabbix 7.0 supports two external vaults:

- CyberArk Vault
  - By default, the vault listens on TCP port 1858
  - Use the HTTPS protocol to communicate with the vault
- Zabbix uses CyberArk Vault CV12
  - Secure access to sensitive credentials through APIs or SDKs
  - Secrets are managed using CyberArk's PVWA (Password Vault Web Access) or CLI tools
- Macro values use the following path
  - AppID>&Safe=<SafeName>&Object=<ObjectName>

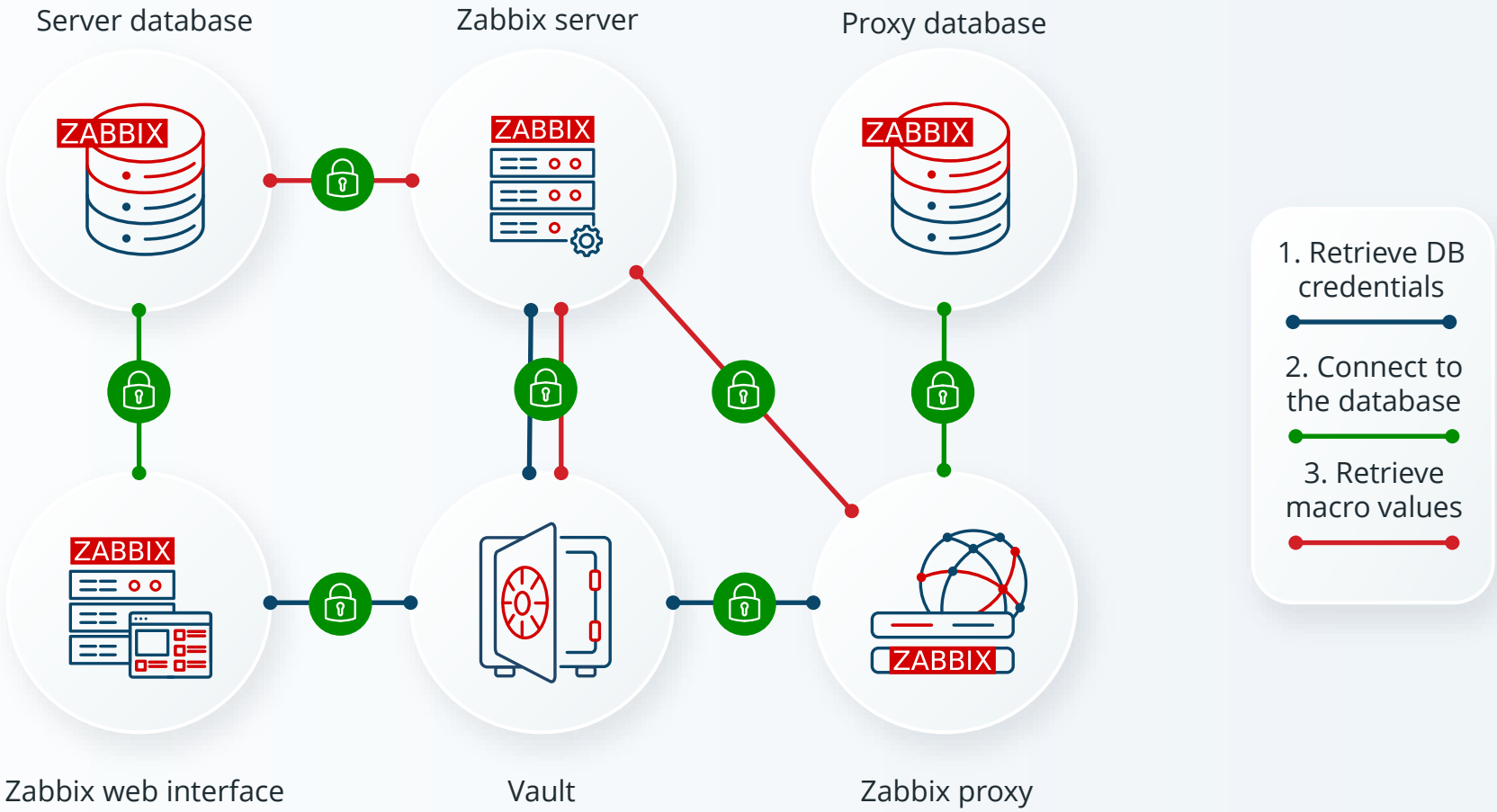
Communications with the vaults

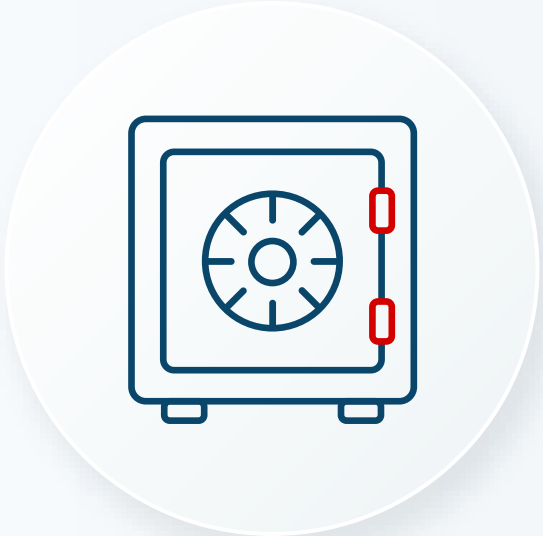
- Zabbix retrieves database access credentials for Zabbix server, proxies, and frontend
- User macro values for Zabbix server



Communications with the vaults

- Zabbix provides read-only access to the secrets in a vault
  - Secrets/credentials are exchanged between some of the components
  - Make sure to encrypt communications as well





Vault installation

Installation options

- Vault can be installed on the same machine as Zabbix components or separate
  - Docker images can be used as well
- Installation on a separate machine may be a more secure option
  - In case DB/Zabbix will get compromised, secret will remain protected
- Separate Zabbix components may use separate vaults

Vault installation steps

ZABBIX

- Install the HashiCorp repo and then vault

```
# dnf config-manager --add-repo https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo
# dnf -y install vault
```

- Start the vault

```
# systemctl enable vault --now
```

- Check vault status

```
# systemctl status
```

```
WARNING! VAULT_ADDR and -address unset. Defaulting to https://127.0.0.1:8200.
Error checking seal status: Get "https://127.0.0.1:8200/v1/sys/seal-status": tls: failed to verify
certificate: x509: cannot validate certificate for 127.0.0.1 because it doesn't contain any IP SANs
```

- The server expects TLS enabled by default. You must then also configure tls_cert_file and tls_key_file and set a path to a valid TLS certificate and key file respectively.

Vault installation steps

- Use openssl to generate self-signed certificate (or use certificates from a trusted CA) to encrypt all communications, and thus exchanged data, with vault
  - /opt/vault/tls/ is the default vault directory for certs, confirm that DNS and SAN are correct

```
# openssl req -x509 -newkey rsa:4096 -sha256 -days 365 \
-nodes -keyout /opt/vault/tls/vault-key.pem -out /opt/vault/tls/vault-cert.pem \
-subj "/CN=vault.zabbix.meetup" \
-addext "subjectAltName=DNS:vault.zabbix.meetup,IP:127.0.0.1"
```

- Change the owner to vault

```
# chown -R vault:vault /opt/vault/tls/
```

- Edit vault configuration file

```
# vi /etc/vault.d/vault.hcl
```

```
# HTTPS listener
listener "tcp" {
  address       = "vault.zabbix.meetup:8200"
  tls_cert_file = "/opt/vault/tls/vault-cert.pem"
  tls_key_file  = "/opt/vault/tls/vault-key.pem"
}
```

Vault installation steps

- Restart the vault and check the status again

```
# systemctl restart vault
# vault status
```

```
Error checking seal status: Get "https://127.0.0.1:8200/v1/sys/seal-status": dial tcp 127.0.0.1:8200:
connect: connection refused
```

- Export the VAULT_ADDR environment variable to set correct value for "vault" variable and send requests to correct vault server address
  - Check status once again

```
# export VAULT_ADDR='https://vault.zabbix.meetup:8200'
# vault status
```

```
Error checking seal status: Get "https://vault.zabbix.meetup:8200/v1/sys/seal-status": tls: failed to
verify certificate: x509: certificate signed by unknown authority
```

- Vault uses strict verification of all TLS certificates by default. You may disable this or

```
# export VAULT_SKIP_VERIFY=true
```

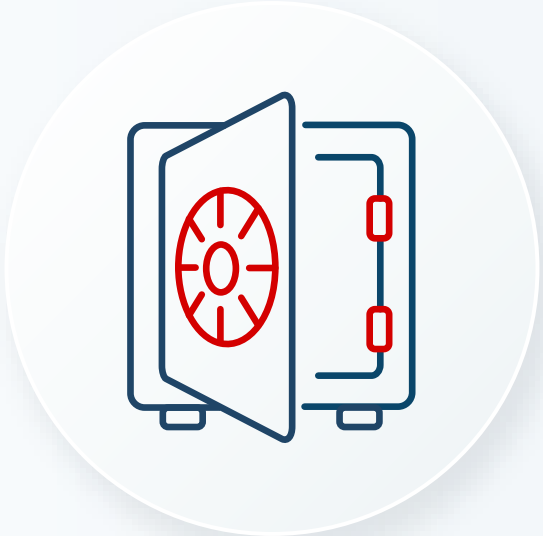
Vault installation steps

- Check the Vault status one more time

```
# vault status
```

```
Key                Value
---                -----
Seal Type          shamir
Initialized        false
Sealed             true
Total Shares       0
Threshold          0
Unseal Progress    0/0
Unseal Nonce       n/a
Version            1.19.0
Build Date         2025-03-04T12:36:40Z
Storage Type       file
HA Enabled         false
```

- Installation complete!



Vault initialization and unseal

Vault initialization

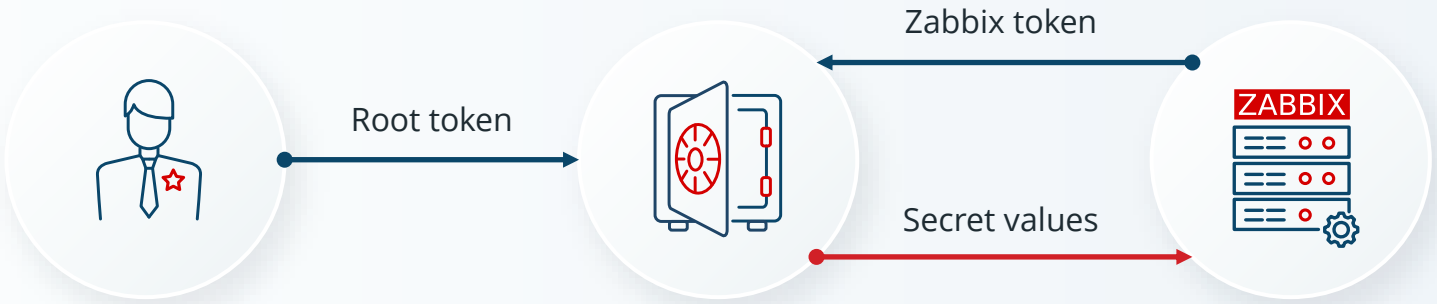
After installation is finished, the vault needs to be initialized

- Initialization is a process by which Vault's is prepared to receive data.
- The default Vault configuration uses Shamir's Secret Sharing to split the root key into a configured number of shards (referred as key shares, or unseal keys)
  - Number of unseal keys and amount to reconstruct root key can be specified
  - -key-shares (int: 5) - Number of "unseal keys" to generate.
  - -key-threshold (int: 3) - Number of key shares to reconstruct the root key.
  - Key threshold must be less than or equal to -key-shares.
  - Without unsealing, data stays encrypted even for the vault itself

Vault unsealing

When the vault is unsealed, a token is used to access the secrets:

- A root token will be generated when the vault is initialized
- Tokens are renewed automatically by Zabbix until the end of TTL
  - Root token does not have a TTL and can be only revoked
- Other tokens can be generated to provide diferrent access permissions



Vault initialization

- To initialize the vault, 3 unseal keys with 2 required to reconstruct root key will be used

```
# vault operator init -key-shares=3 -key-threshold=2
```

```
vault operator init -key-shares=3 -key-threshold=2

Unseal Key 1: QzYBo2PNo8ykMKCEww+drEJ1xiUnD9+L8gbI+3o5xvF4
Unseal Key 2: gMNBu9KOlH7p2ZkzyHJeduRWjaFa7WCrRue4k7vPmcxz
Unseal Key 3: 1Ha1FVcpRKh7BnwQCKcKvct/uZaP8tDRNqsLjo0pVO2z

Initial Root Token: hvs.qiQ3jfF4HTadgs5JsuBRDRld

Vault initialized with 3 key shares and a key threshold of 2. Please securely
distribute the key shares printed above. When the Vault is re-sealed,
restarted, or stopped, you must supply at least 2 of these keys to unseal it
before it can start servicing requests.

Vault does not store the generated root key. Without at least 2 keys to
reconstruct the root key, Vault will remain permanently sealed!

It is possible to generate new unseal keys, provided you have a quorum of
existing unseal keys shares. See "vault operator rekey" for more information.
```

- Make sure to store those and store securily!

Vault unsealing

- To unseal the vault, execute twice, using diferrent unseal keys

```
# vault operator unseal
```

```
Unseal Key (will be hidden):
```

- The result should be

```
Key             Value
---             -----
Seal Type       shamir
Initialized     true
Sealed          false
Total Shares    3
Threshold       2
Version         1.19.0
Build Date      2025-03-04T12:36:40Z
Storage Type    file
Cluster Name    vault-cluster-2df03388
Cluster ID      f2900e42-ad9f-c7e1-5f13-29a08483213f
HA Enabled      false
```

- Vault is now unsealed!

Storing Zabbix database credentials in the Vault

Policies and roles vault

Secure access to Zabbix secrets is based on Vault policies and roles

- After initializing the supported kv2 engine to store the secrets policies can applied
- Policies give the ability to configure granular control over access to secrets
- Roles simplify configuration of an auth method or secrets engine

Typically, three types of policies/roles are used with Zabbix

- Zabbix server — access to all secrets
- Zabbix frontend — access only to the primary database credentials
- Zabbix proxy — access only to the proxy database credentials

Zabbix policies

Secure access to Zabbix secrets is based on Vault policies and roles

- read: Allows for data to be read at a given path.
- list: Allows for values to be listed at the given path.
  Different from read, shows you a list of available secrets at a path but not their contents.

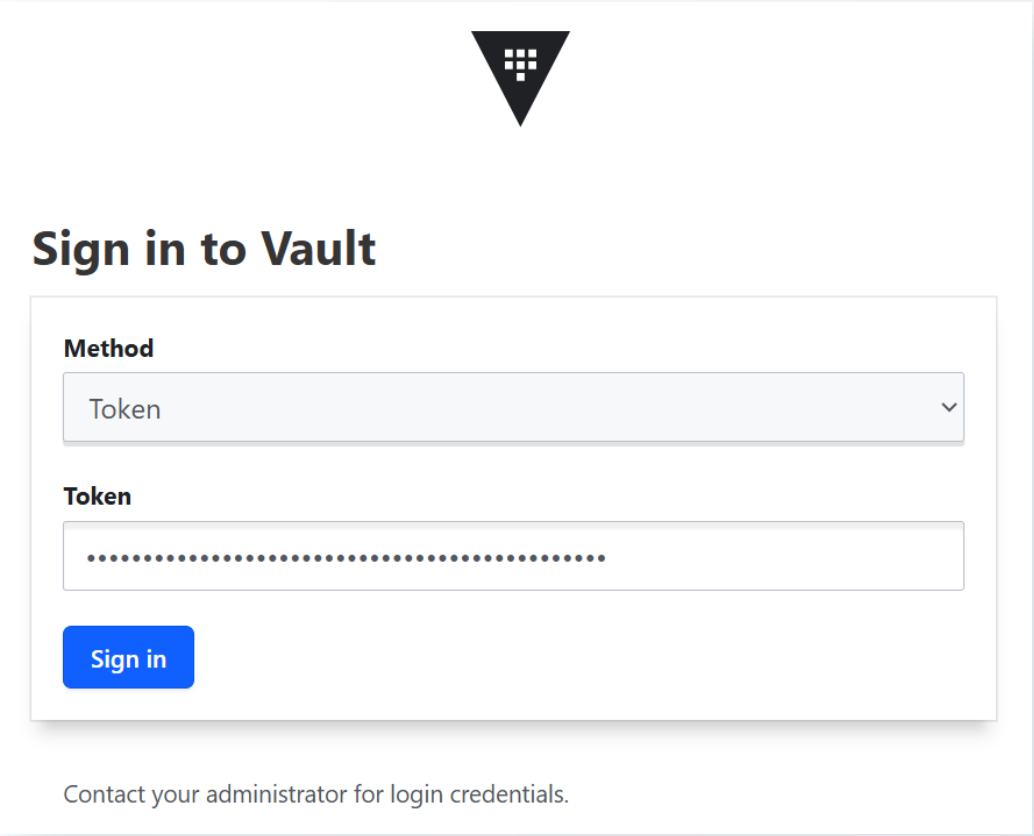

```
path "zabbix/data/zabbix_db"
{
  capabilities = ["list","read"]
}
path "zabbix/data/macros/*"
{
  capabilities = ["list","read"]
}
```


```
path "zabbix/data/frontend"
{
  capabilities = ["list","read"]
}
```

Configuring Zabbix credentials through vault

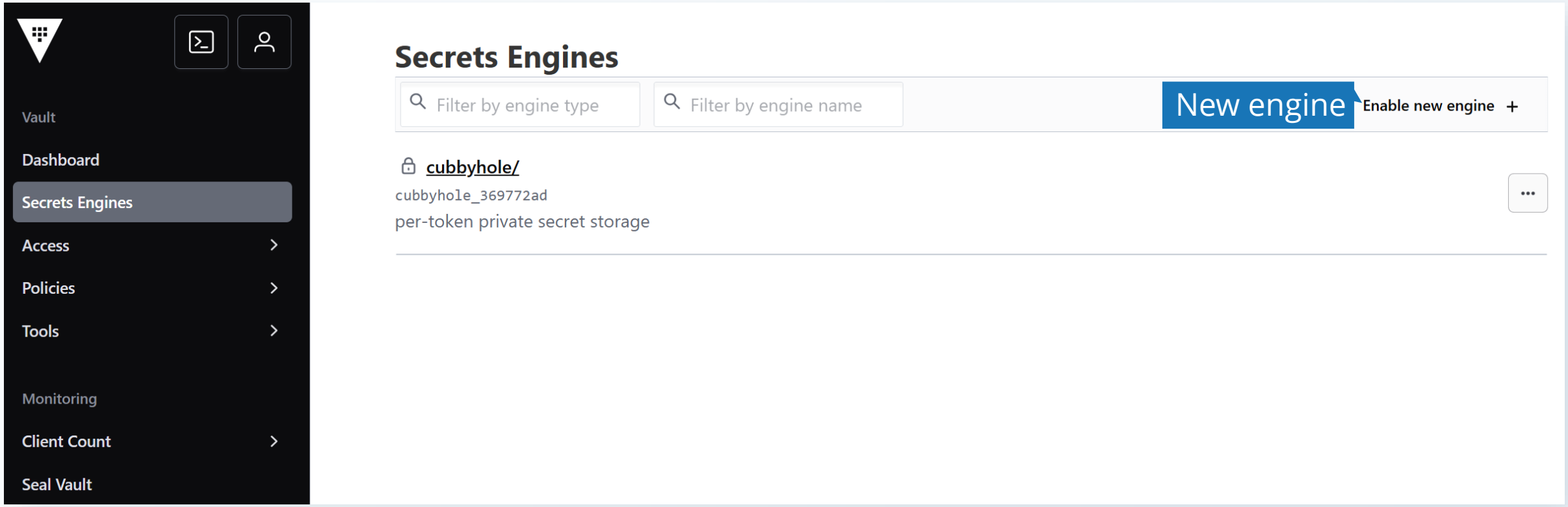
Initialize kv2 secrets engine

Sign in to the vault using root token



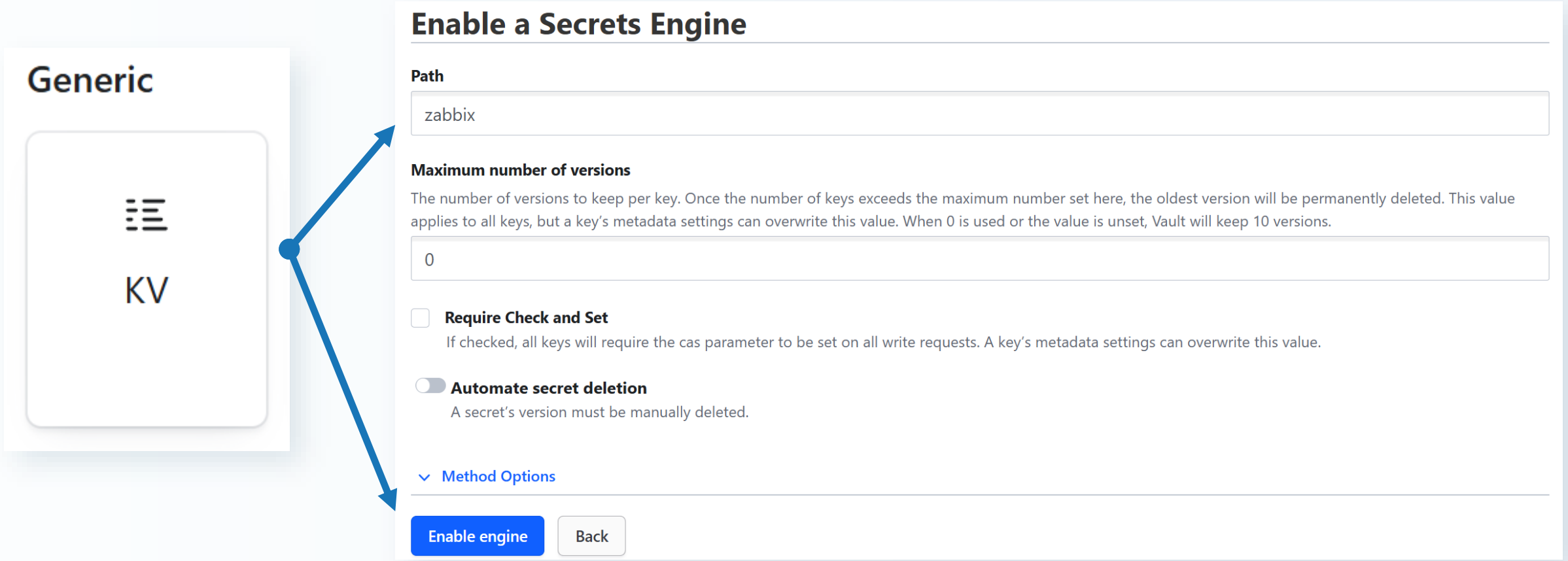
Initialize kv2 secrets engine

Click on "Secrets Engines" in the left menu and then "Enable new engine +"



Initialize kv2 secrets engine

Click on KV and set path to "zabbix", press "Enable engine"



Initialize kv2 secrets engine

ZABBIX

Alternatively, the same can be done from the command line

- Export vault root token (specify your vault root token)

```
# read -s -p "Enter vault token:" VAULT_TOKEN
```

```
Enter vault token: <VAULT ROOT TOKEN>
```

- Export the token

```
# export VAULT_TOKEN
```

- Initialize the kv2 secrets engine

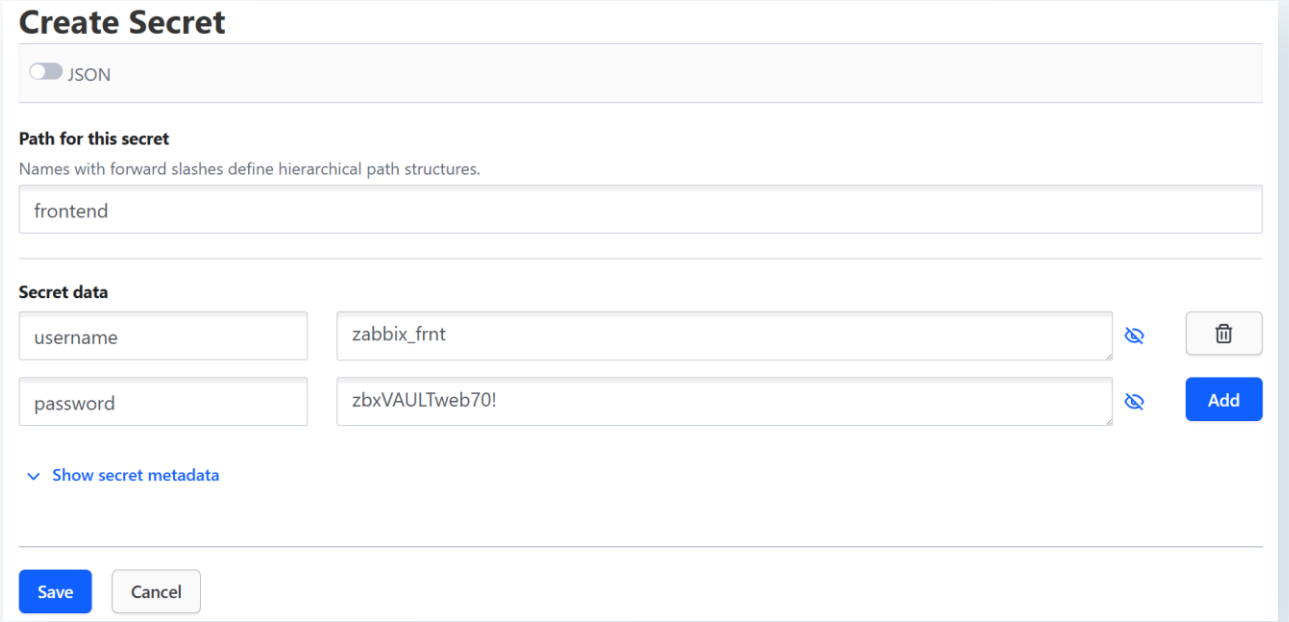
```
# vault secrets enable -path=zabbix kv-v2
```

```
Success! Enabled the kv-v2 secrets engine at: zabbix/
```

Having secrets

In "Secrets Engines", select "zabbix" engine, click "Create secret +" on the right

- Path for this secret frontend
- Secret data
  - username <your MySQL login>
  - password <your MySQL pass>
- Press Save



- Alternatively, this can be done using a CLI one-liner (replace with your DB credentials)

```
# vault kv put zabbix/frontend username=zabbix_frnt password=zbxVAULTweb70!
```

Creating role based access

Secrets are now created (Zabbix web interface database credentials), secure access tokens can be created based on roles

- A role is a set of parameters that you group together to simplify configuration.
- Authentication requests only need to pass the role name to Vault.
- Vault will read the role configuration and issue a token based on the settings of that role.
- Roles can be created using Vault web interface based or regular CLI.

Creating role based access

In Vault GUI click on terminal icon (top left)

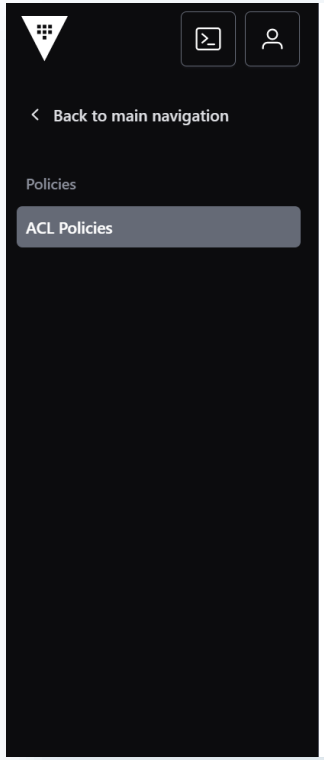
- Create new token role with 30-day renewal period

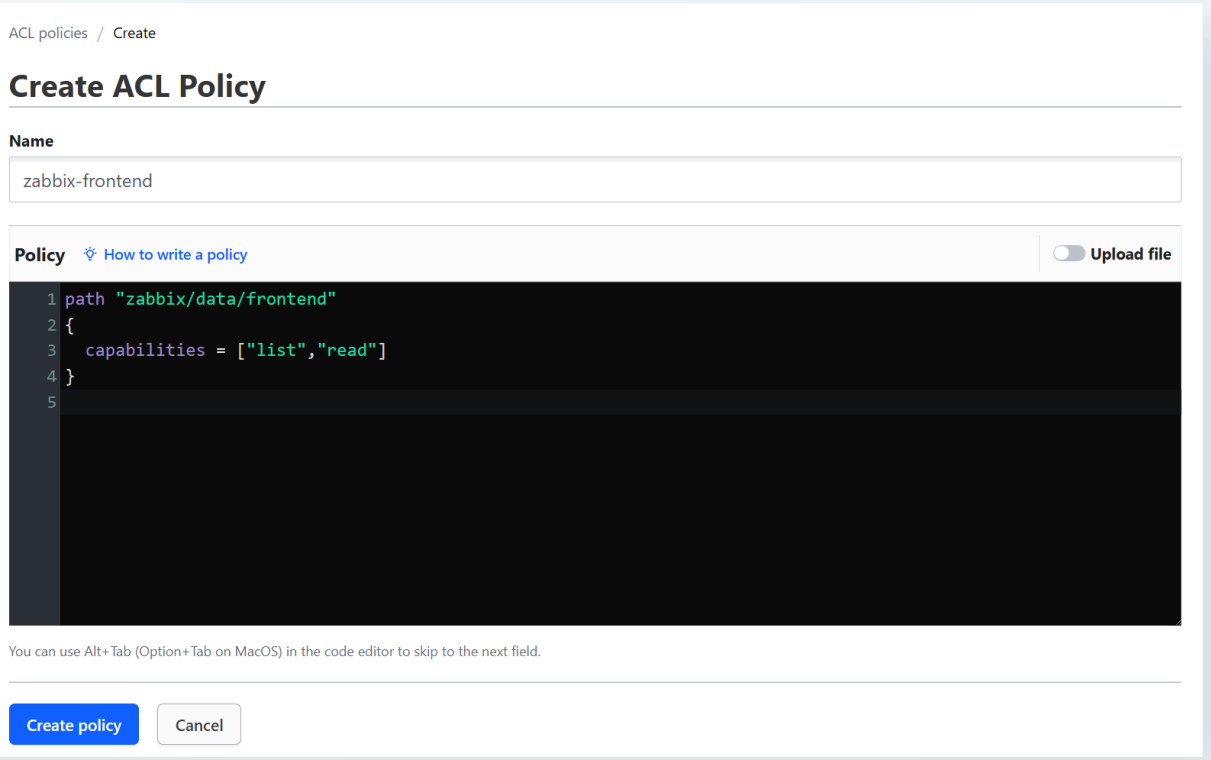
```
# vault write auth/token/roles/zabbix allowed_policies="zabbix-frontend,zabbix-server" period="720h"
Success! Data written to: auth/token/roles/zabbix
```

Close the CLI interface and click on Policies > Create ACL Policy

- Name zabbix-frontend
- Policy

```
path "zabbix/data/frontend"
{
  capabilities = ["list","read"]
}
```





- Press Create policy

Creating role based access

Tokens, hovewer, can be generated only using CLI on the server side

- Create new access token role for web interface with 30-day renewal period

```
# vault token create -policy=zabbix-frontend -role=zabbix
```

```
Key                  Value
---                  -----
token                hvs.CAESIErmwSIWKI_dBvEjsoX8pxeGiVuvygLsYm4JftibOI_vGh4KHGh2cy5FT21abnVhRklYU2MzbEp4UElYcktadFo
token_accessor       ogJNvsJRQ2MkmPUbBhkvg5Tz
token_duration       720h
token_renewable      true
token_policies       ["default" "zabbix-frontend"]
identity_policies    []
policies             ["default" "zabbix-frontend"]
```

Testing role based access

Confirm that web interface secrets can be retrieved using created token

```
# VAULT_TOKEN=<PUT VAULT FRONTEND TOKEN HERE> vault kv get zabbix/frontend
```

```
==== Secret Path ====
zabbix/data/frontend

======= Metadata =======
Key                Value
---                -----
.....
====== Data ======
Key         Value
---         -----
password    zbxVAULTweb70!
username    zabbix_frnt
```

Testing role based access

We should also confirm communication from the Zabbix frontend machine

```
# curl -H "X-Vault-Token: <VAULT FRONTEND TOKEN>" \
-X GET https://vault.zabbix.meetup:8200/v1/zabbix/data/frontend
```

```
{
..."data": {
    "data": {
      "password": "zbxVAULTweb70!",
      "username": "zabbix_frnt"
    },
...
}
```

- If using self-signed certificates, those must be added to trust on the system level

```
# cp /opt/vault/tls/*.pem /etc/pki/ca-trust/source/anchors/
# update-ca-trust extract
```

Configuring Zabbix frontend

All that it left is to configure Zabbix frontend, to use vault.

```
# vi /etc/zabbix/web/zabbix.conf.php
```

```
$DB['USER']                     = 'zabbix_frnt'
$DB['PASSWORD']                 = 'zbxVAULTweb70!'
// Vault configuration. Used if database credentials are stored in Vault secrets manager.
$DB['VAULT']                    = 'HashiCorp';
$DB['VAULT_URL']                = 'https://vault.zabbix.meetup:8200';
$DB['VAULT_DB_PATH']            = 'zabbix/frontend';
$DB['VAULT_TOKEN']              = '<VAULT FRONTEND TOKEN>';
```

Confirm that web interface is working and no errors are seen.

Configuring Zabbix server

- This time, using CLI only, create another secret for Zabbix server on vault machine

```
# vault kv put zabbix/server username=zabbix_bknd password=zbxVAULTsrv70!
```

```
=== Secret Path ===
zabbix/data/server
======= Metadata =======
Key                Value
---                -----
created_time       2025-03-24T12:01:48.035102296Z
```

- Create a policy for Zabbix server

```
# vi /etc/vault.d/zabbix-server-policy.hcl
```

```
path "zabbix/data/server"
{
  capabilities = ["list","read"]
}
```

Configuring Zabbix server

- Write new policy to vault

```
# vault policy write zabbix-server /etc/vault.d/zabbix-server-policy.hcl
```

- Create new token for this policy

```
# vault token create -policy=zabbix-server -role=zabbix
```

```
Key                  Value
---                  -----
token                hvs.CAESIHcGu2OfWTmBFzcSBp4hU95rtPlYeUk4RUiTrxXRBgj_Gh4KHGh2cy5WSUFFc3pyaEEweWlVVUFlUUxZUnZXZW4
token_accessor       juCIoN0trhdh1A1onWuyNR1l
token_duration       720h
token_renewable      true
token_policies       ["default" "zabbix-server"]
identity_policies    []
policies             ["default" "zabbix-server"]
```

- Test the token

```
# VAULT_TOKEN=<PUT VAULT SERVER TOKEN HERE> vault kv get zabbix/server
```

Configuring Zabbix server

- Edit Zabbix server configuration file (make sure DBUser and DBPassword are removed)

```
# vi /etc/zabbix/zabbix_server.conf
```

```
DBUser=zabbix_bknd
DBPassword=zbxVAULTsrv70!

### Option: Vault
#       Specifies vault:
Vault=HashiCorp

### Option: VaultToken
#  Vault authentication token that should have been generated exclusively for Zabbix server with read only permission
#  to paths specified in Vault macros and read only permission to path specified in optional VaultDBPath configuration parameter.
VaultToken=<ZABBIX SERVER VAULT TOKEN>

### Option: VaultURL
#       Vault server HTTP[S] URL. System-wide CA certificates directory will be used if SSLCALocation is not specified.
VaultURL=https://vault.zabbix.meetup:8200

### Option: VaultDBPath
#       Vault path from where credentials for database will be retrieved by keys 'password' and 'username'.
VaultDBPath=zabbix/server
```

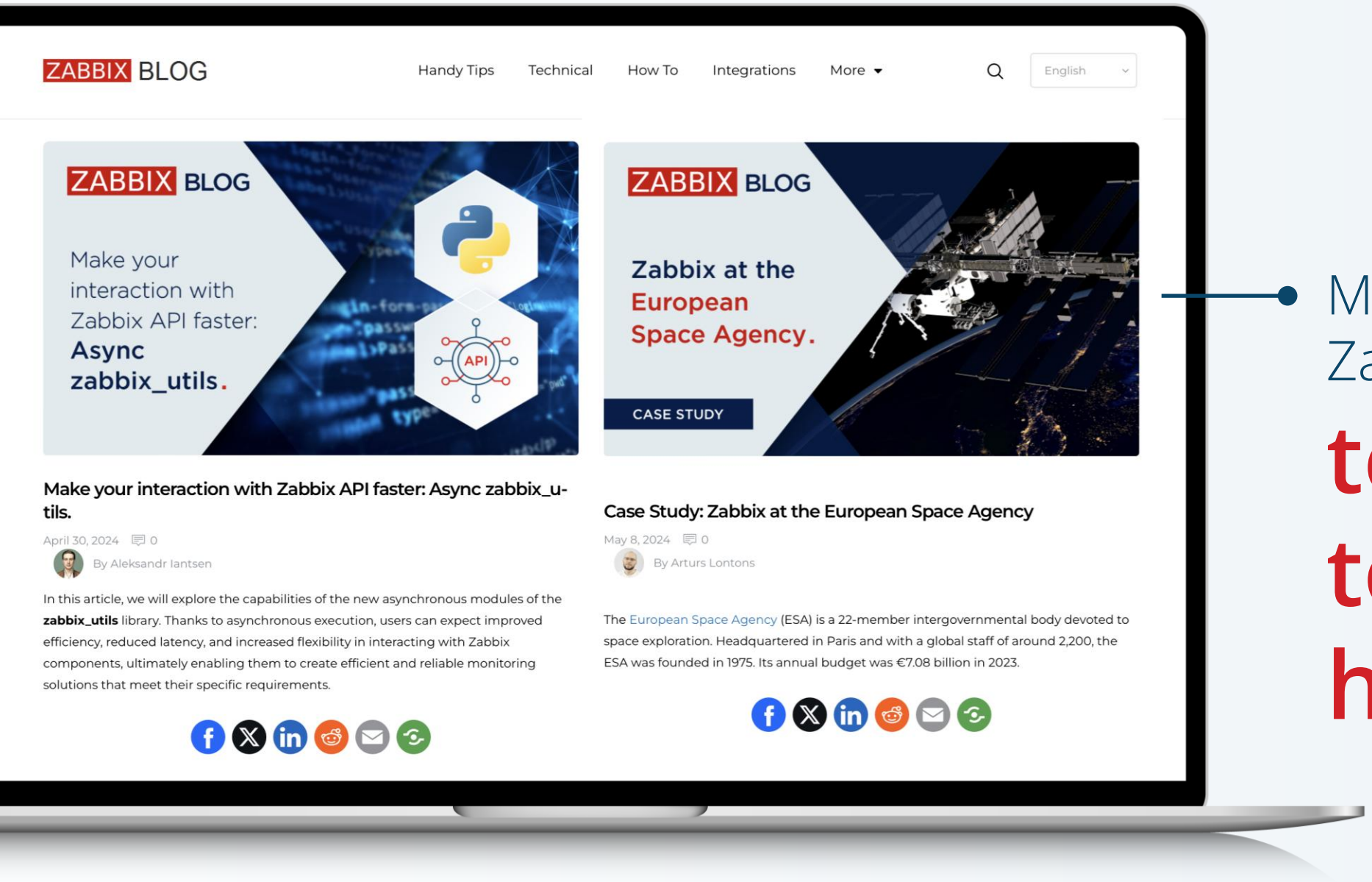
- Restart Zabbix server to apply the changes!

Extra notes

Extra notes to simplify configuration.

- Initial configuration can be done using CLI only, to speed up the process
- For proper security use verified CA to generate certs (Self-signed or Trusted)
- If using self signed CA or certs, install those on each Zabbix component machine
- Instead of storing tokens in config files, store them as environment variables
- Tokens are renewed automatically by Zabbix until TTL!
- Periodic token can be used to ensure non-expiry as long as those are renewed.
- Zabbix can monitor vault out of the box
- Learn more Vault secrets on Zabbix Certifies Expert training


ZABBIX
20 YEARS
Thank you!



Find out more

Monitor the latest Zabbix news,

technical topics and how-tos

Meet ZABBIX around the world!

Always trying to be closer to our users, we actively take part in various IT expos, conferences and meetups all over the world

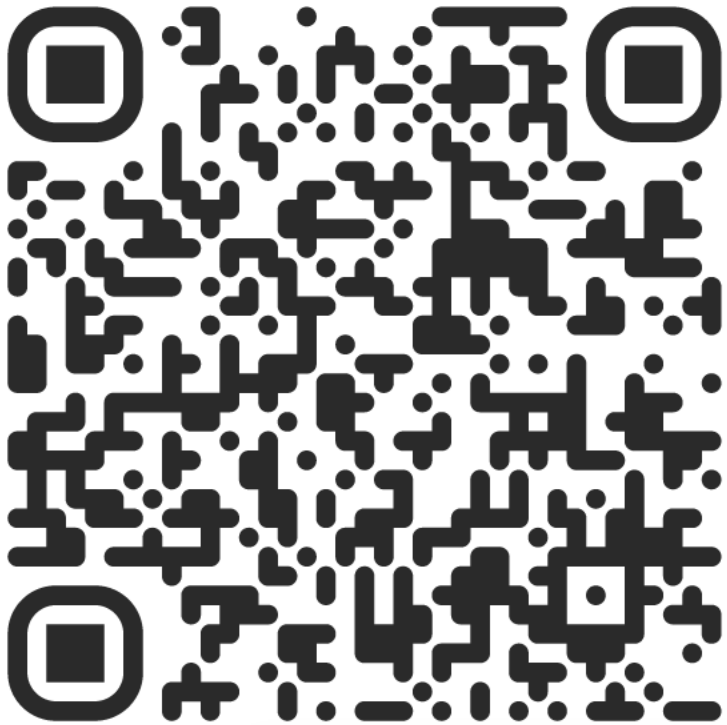
Explore all events


ZABBIX '25
CONFERENCE
GERMANY
Berlin
May 14 - 15 • 2025
Find out more

ZABBIX SUMMIT

Riga

October 8 - 10 ▪ 2025



Save the date!

Contact us

USA

Phone +1 877-4-ZABBIX
+1 877-4-922249 (Toll-free)

Email sales@zabbix.com

EUROPE

Phone +371 6778-4742

Email sales@zabbix.com

JAPAN

Phone +81 3-4405-7338

Email sales@zabbix.co.jp

CHINA

Phone +86 021-6978-6188

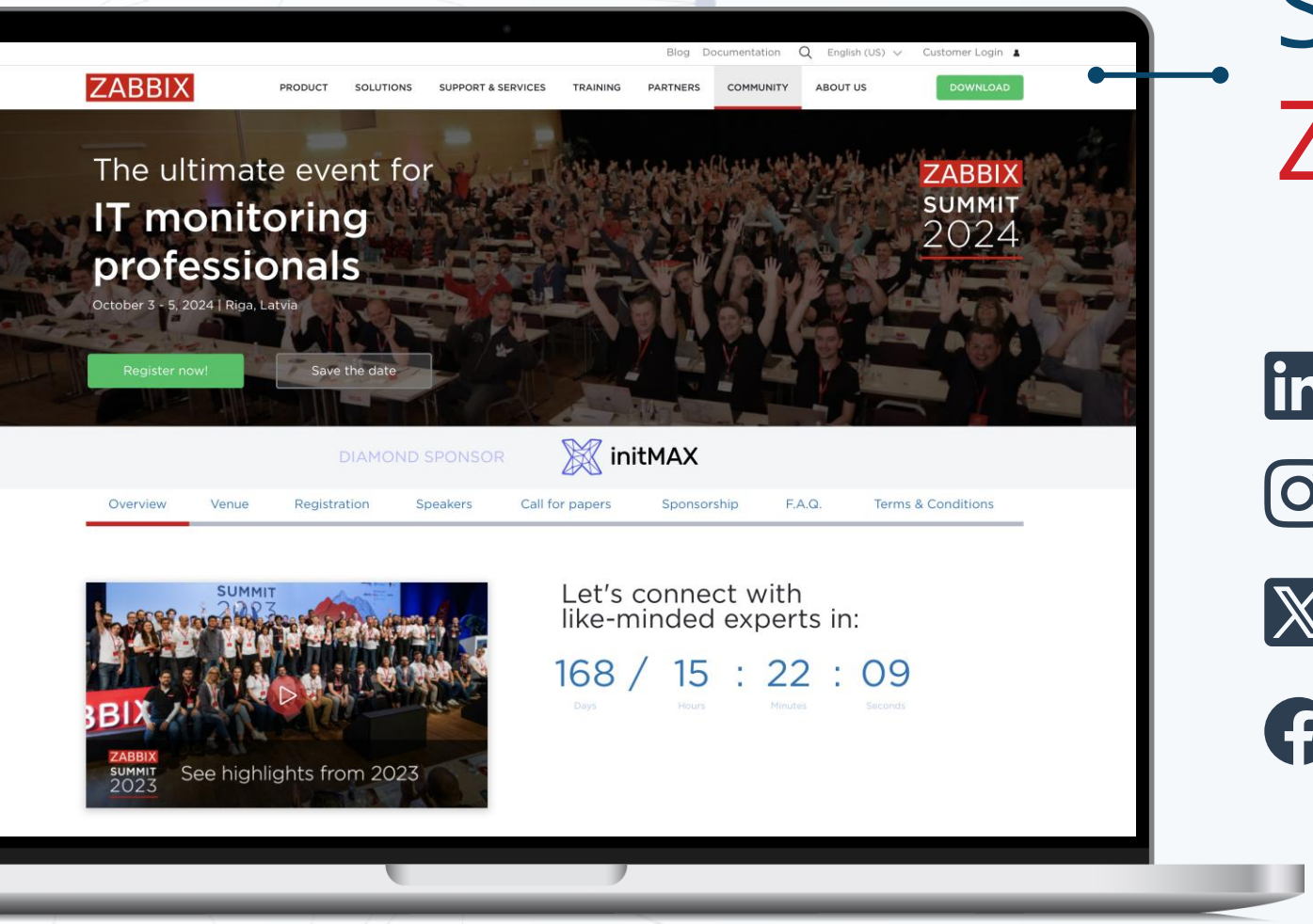
Email china@zabbix.co.jp

LATIN AMERICA

Phone	Argentina | Buenos Aires	+54 113989-4060
	Brazil | San Paulo	+55 11 4210-5104
	Chile |National	+56 44 890-9410
	Colombia | Bogota	+57 1 3819310
	Mexico | Mexico city	+52 55 8526-2606

Email sales.latam@zabbix.com

ZABBIX



Stay updated on Zabbix news:

- Zabbix
- zabbix_official
- @zabbix
- Zabbix