Installing HashiCorp Boundary

HashiCorp Boundary is an open-source tool for secure access management. This guide provides step-by-step instructions for installing Boundary on a system.

1. Prerequisites

Before you begin, ensure you have the following:

- A supported operating system (Linux, macOS, or Windows).
- Access to a terminal or command prompt.
- Administrative privileges to install software.
- Service Princial Name (WP09_PackerPrincipal)

You can create SPN with command

```
az ad sp create-for-rbac -n "WP09_PackerPrincipal" --role="Contributor" --scopes="/subscriptions/<your-subscription>"
```

2. Provision new Azure VM

Follow the Azure portal instructions to create a new virtual machine. Ensure that the VM meets the system requirements for running Boundary.

3. Install and Configure PostgreSQL

3.1 Install PostgreSQL

Run the following command in your terminal:

```
sudo apt install postgresql -y
```

```
Setting up libtypes-serialiser-perl (1.01-1) ...
Setting up libjson-perl (4.04000-1) ...
Setting up sysstat (12.5.2-2ubuntu0.2) ...

Creating config file /etc/default/sysstat with new version
update-alternatives: using /usr/bin/sar.sysstat to provide /usr/bin/sar (sar) in auto mode
Created symlink /etc/systemd/system/sysstat.service.wants/sysstat-collect.timer → /lib/systemd/system/sysstat-collect.timer.
Created symlink /etc/systemd/system/sysstat.service.wants/sysstat-summary.timer → /lib/systemd/system/sysstat-summary.timer.
Created symlink /etc/systemd/system/multi-user.target.wants/sysstat.service → /lib/systemd/system/sysstat.service.
Setting up libjson-xs-perl (4.030-1build3) ...
Setting up postgresql-common (238) ...
Adding user postgres to group ssl-cert

Creating config file /etc/postgresql-common/createcluster.conf with new version
Building PostgreSQL dictionaries from installed myspell/hunspell packages...
Removing obsolete dictionary files:
Created symlink /etc/systemd/system/multi-user.target.wants/postgresql.service → /lib/systemd/system/postgresql.service.
Setting up postgresql-14 (14.13-0ubuntu0.22.04.1) ...
Creating new PostgreSQL cluster 14/main ...
/usr/lib/postgresql/14/bin/initdb -D /var/lib/postgresql/14/main --auth-local peer --auth-host scram-sha-256 --no-instructions
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "C.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgresql/14/main ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok
update-alternatives: using /usr/share/postgresql/14/man/man1/postmaster.1.gz to provide /usr/share/man/man1/postmaster.1.gz (postmaster.1.gz) in auto mode
Setting up postgresql (14+238) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

3.2 Configure PostgreSQL

Open the PostgreSQL command line:

```
sudo –u postgres psql
```

Create a new database and user:

```
CREATE DATABASE boundary;
CREATE USER boundary WITH ENCRYPTED PASSWORD 'hidden_password';
GRANT ALL PRIVILEGES ON DATABASE boundary TO boundary;
```

Quit the postgres `\q`

```
postgres=# CREATE DATABASE boundary;
CREATE DATABASE
postgres=# CREATE USER boundary WITH ENCRYPTED PASSWORD '*************';
CREATE ROLE
postgres=# GRANT ALL PRIVILEGES ON DATABASE boundary TO boundary;
GRANT
postgres=# \q
root@hcl–boundary–02:~#
```

3.3 Enable Password Authentication

Edit the PostgreSQL configuration file to allow password authentication:

```
echo "host    all             all             127.0.0.1/32
password" | sudo tee -a /etc/postgresql/14/main/pg_hba.conf
```

Then `Restart` the PostgreSQL service:

```
sudo systemctl restart postgresql
```

4. Install and Configure Apache2

4.1 Install Apache2

Run the following commands:

```
sudo apt install apache2 -y
sudo a2enmod ssl proxy proxy_http
sudo systemctl start apache2
```

```
root@hcl-boundary-02:~# apt install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.3-0 mailcap mime-support
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser ufw bzip2-doc
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils bzip2 libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.3-0 mailcap mime-support
0 upgraded, 12 newly installed, 0 to remove and 4 not upgraded.
Need to get 2124 kB of archives.
After this operation, 8458 kB of additional disk space will be used.
Get:1 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libapr1 amd64 1.7.0-8ubuntu0.22.04.1 [108 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1 amd64 1.6.1-5ubuntu4.22.04.2 [92.8 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1-dbd-sqlite3 amd64 1.6.1-5ubuntu4.22.04.2 [11.3 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1-ldap amd64 1.6.1-5ubuntu4.22.04.2 [9170 B]
Get:5 http://azure.archive.ubuntu.com/ubuntu jammy/main amd64 liblua5.3-0 amd64 5.3.6-1build1 [140 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 apache2-bin amd64 2.4.52-1ubuntu4.12 [1348 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 apache2-data all 2.4.52-1ubuntu4.12 [165 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 apache2-utils amd64 2.4.52-1ubuntu4.12 [89.1 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu jammy/main amd64 mailcap all 3.70+nmu1ubuntu1 [23.8 kB]
Get:10 http://azure.archive.ubuntu.com/ubuntu jammy/main amd64 mime-support all 3.66 [3696 B]
Get:11 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 apache2 amd64 2.4.52-1ubuntu4.12 [97.9 kB]
Get:12 http://azure.archive.ubuntu.com/ubuntu jammy/main amd64 bzip2 amd64 1.0.8-5build1 [34.8 kB]
Fetched 2124 kB in 5s (458 kB/s)
Selecting previously unselected package libapr1:amd64.
(Reading database ... 65465 files and directories currently installed.)
Preparing to unpack .../00-libapr1_1.7.0-8ubuntu0.22.04.1_amd64.deb ...
Unpacking libapr1:amd64 (1.7.0-8ubuntu0.22.04.1) ...
Selecting previously unselected package libaprutil1:amd64.
Preparing to unpack .../01-libaprutil1_1.6.1-5ubuntu4.22.04.2_amd64.deb ...
Unpacking libaprutil1:amd64 (1.6.1-5ubuntu4.22.04.2) ...
Selecting previously unselected package libaprutil1-dbd-sqlite3:amd64.
Preparing to unpack .../02-libaprutil1-dbd-sqlite3_1.6.1-5ubuntu4.22.04.2_amd64.deb ...
Unpacking libaprutil1-dbd-sqlite3:amd64 (1.6.1-5ubuntu4.22.04.2) ...
Selecting previously unselected package libaprutil1-ldap:amd64.
Preparing to unpack .../03-libaprutil1-ldap_1.6.1-5ubuntu4.22.04.2_amd64.deb ...
Unpacking libaprutil1-ldap:amd64 (1.6.1-5ubuntu4.22.04.2) ...
Selecting previously unselected package liblua5.3-0:amd64.
Preparing to unpack .../04-liblua5.3-0_5.3.6-1build1_amd64.deb ...
```

```
root@hcl-boundary-02:~# sudo a2enmod ssl proxy
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
Enabling module proxy.
To activate the new configuration, you need to run:
  systemctl restart apache2
root@hcl-boundary-02:~#
```

4.2 Create a Self-Signed Certificate

Create a directory for SSL certificates and generate a self-signed certificate:

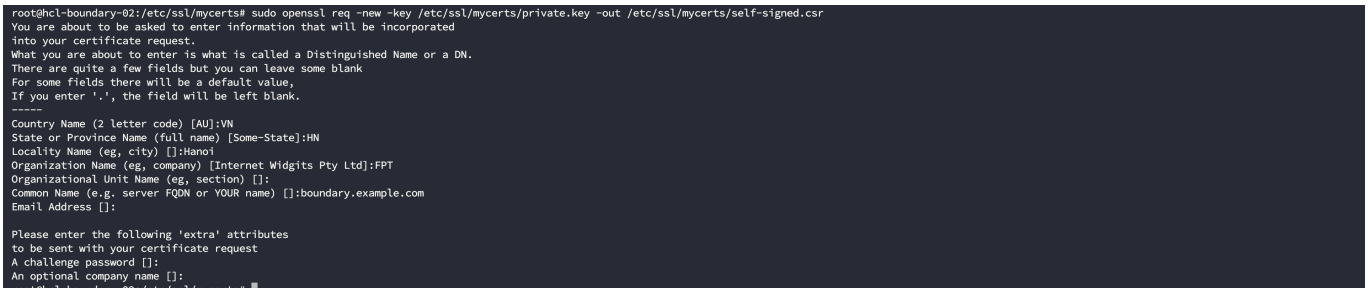
```
sudo mkdir /etc/ssl/mycerts
```

Generate a private key with the following command:

```
sudo openssl genrsa -out /etc/ssl/mycerts/private.key 2048
```

Run this command to create a CSR. You'll be prompted to enter some information.

```
sudo openssl req -new -key /etc/ssl/mycerts/private.key -out
/etc/ssl/mycerts/self-signed.csr
```



Use the following command to create the self-signed certificate:

```
sudo openssl x509 -req -days 365 -in /etc/ssl/mycerts/self-signed.csr -
signkey /etc/ssl/mycerts/private.key -out /etc/ssl/mycerts/self-signed.crt
```

Output

```
root@hcl-boundary-02:/etc/ssl/mycerts# sudo openssl x509 -req -days 365 -
in /etc/ssl/mycerts/self-signed.csr -signkey /etc/ssl/mycerts/private.key
-out /etc/ssl/mycerts/self-signed.crt
Certificate request self-signature ok
subject=C = VN, ST = HN, L = Hanoi, O = FPT, CN = boundary.example.com
```

4.3 Create a Dummy SSL Site

Edit your Apache configuration file `/etc/apache2/sites-available/default-ssl.conf` to include the following lines:

```
<IfModule mod_ssl.c>
        <VirtualHost 127.0.0.1:443>
                ServerAdmin webmaster@localhost
                ServerName test.example.com
                DocumentRoot /var/www/html

                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined

                SSLEngine on
                SSLProxyEngine on
                SSLProxyVerify none
                SSLProxyCheckPeerCN off
                SSLProxyCheckPeerName off
                SSLCertificateFile      /etc/apache2/ssl/cert.pem
                SSLCertificateKeyFile /etc/apache2/ssl/key.pem
```

```
                <FilesMatch "\.(cgi|shtml|phtml|php)$">
                                SSLOptions +StdEnvVars
                </FilesMatch>
                <Directory /usr/lib/cgi-bin>
                                SSLOptions +StdEnvVars
                </Directory>


        </VirtualHost>
</IfModule>
```

Enable default ssl website

```
a2ensite default-ssl
```

Reload service

```
systemctl reload apache2
```

Test the dummy site:

```
curl -I -k https://127.0.0.1
```

4.4 Update /etc/hosts

Add the following lines to your /etc/hosts file:

```
echo "127.0.0.1 localhost test.example.com controller.example.com
worker.example.com oundary.example.com" | sudo tee -a /etc/hosts
```

```
root@hcl-boundary-02:/etc/apache2/sites-available# echo "127.0.0.1 localhost test.example.com controller.example.com worker.example.com boundary.example.com" | sudo tee -a /etc/hosts
127.0.0.1 localhost test.example.com controller.example.com worker.example.com boundary.example.com
root@hcl-boundary-02:/etc/apache2/sites-available# cat /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
127.0.0.1 localhost test.example.com controller.example.com worker.example.com boundary.example.com
root@hcl-boundary-02:/etc/apache2/sites-available#
```

5. Install and Configure Boundary

5.1 Install Boundary

Add the HashiCorp GPG Key

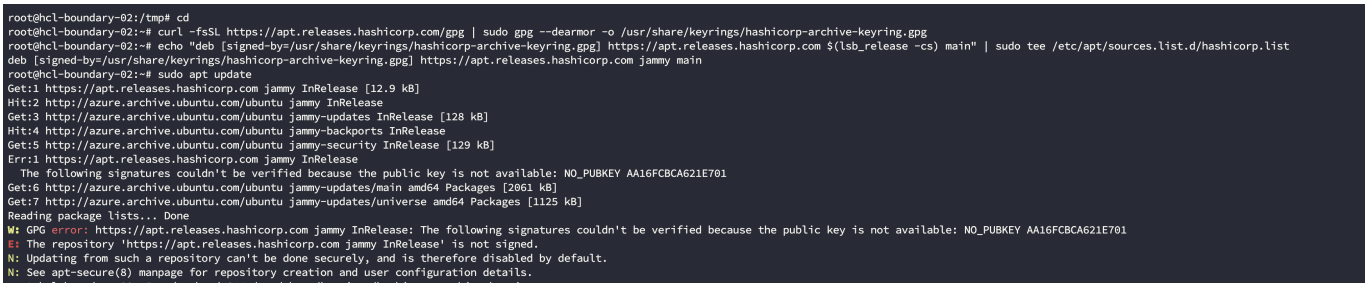
```
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

Add the HashiCorp Repository

```
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
```

Install Boundary

```
sudo apt update
```



Fix GPG Key Errors (if any)

```
sudo chmod 644 /usr/share/keyrings/hashicorp-archive-keyring.gpg
```

```
root@hcl-boundary-02:~# curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
root@hcl-boundary-02:~# echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com jammy main
root@hcl-boundary-02:~# sudo apt update
Get:1 https://apt.releases.hashicorp.com jammy InRelease [12.9 kB]
Hit:2 http://azure.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://azure.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:4 http://azure.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 http://azure.archive.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Err:1 https://apt.releases.hashicorp.com jammy InRelease
  The following signatures couldn't be verified because the public key is not available: NO_PUBKEY AA16FCBCA621E701
Get:6 http://azure.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2061 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1125 kB]
Reading package lists... Done
W: GPG error: https://apt.releases.hashicorp.com jammy InRelease: The following signatures couldn't be verified because the public key is not available: NO_PUBKEY AA16FCBCA621E701
E: The repository 'https://apt.releases.hashicorp.com jammy InRelease' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
root@hcl-boundary-02:~# sudo chmod 644 /usr/share/keyrings/hashicorp-archive-keyring.gpg
root@hcl-boundary-02:~# sudo apt update
Get:1 https://apt.releases.hashicorp.com jammy InRelease [12.9 kB]
Hit:2 http://azure.archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://azure.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://azure.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:5 http://azure.archive.ubuntu.com/ubuntu jammy-security InRelease
Get:6 https://apt.releases.hashicorp.com jammy/main amd64 Packages [150 kB]
Fetched 163 kB in 1s (213 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
4 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@hcl-boundary-02:~#
```

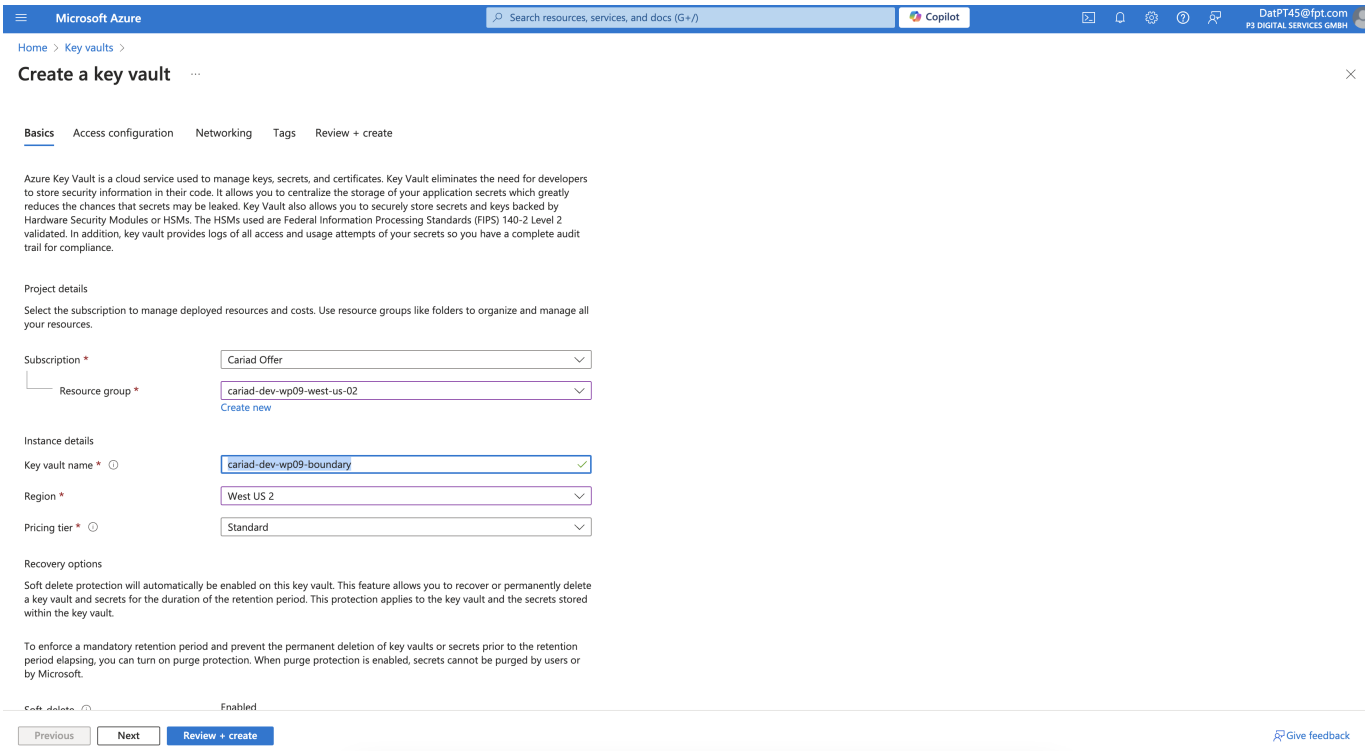
Run command to install `boundary`

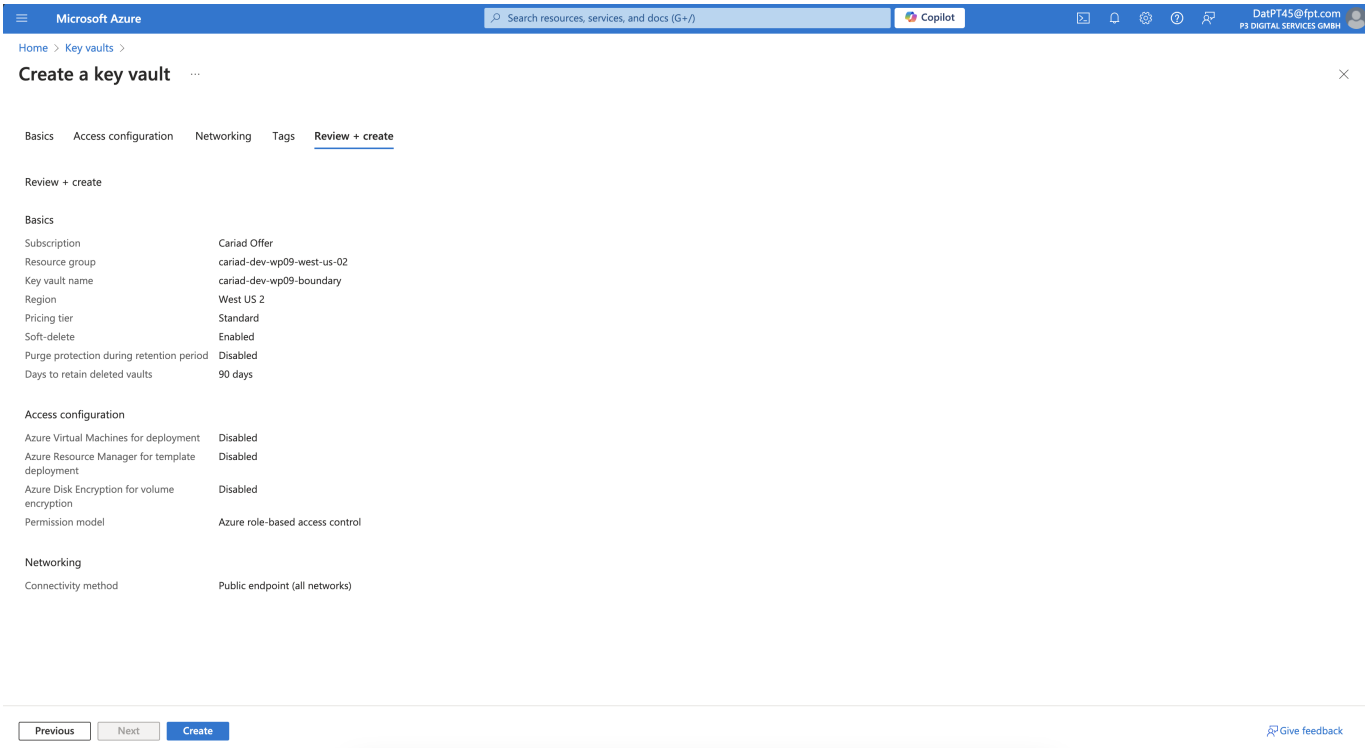
```
sudo apt install boundary -y
```

```
root@hcl-boundary-02:~# sudo apt install boundary -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  boundary
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 145 MB of archives.
After this operation, 201 MB of additional disk space will be used.
Get:1 https://apt.releases.hashicorp.com jammy/main amd64 boundary amd64 0.17.1-1 [145 MB]
Fetched 145 MB in 2s (59.1 MB/s)
Selecting previously unselected package boundary.
(Reading database ... 66225 files and directories currently installed.)
Preparing to unpack .../boundary_0.17.1-1_amd64.deb ...
Unpacking boundary (0.17.1-1) ...
Setting up boundary (0.17.1-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@hcl-boundary-02:~#
```

5.3 Create KMS Keys

Go to Auzre Portal and create new keyvault named `cariad-dev-wp09-boundary`

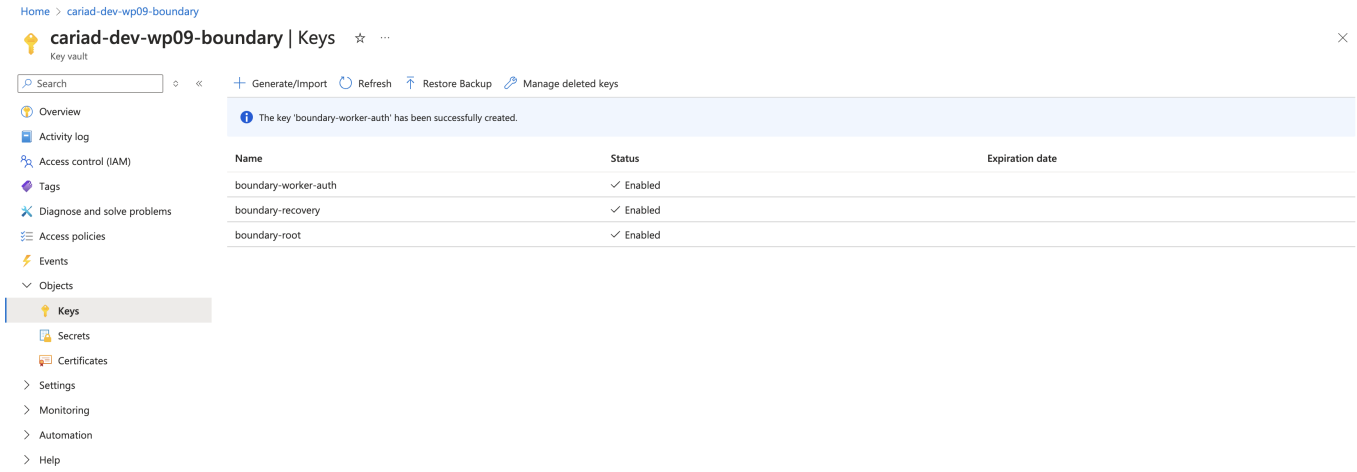




Grant access for the service principal (SPN) with role `Key Vault Crypto Officer`

Then create three keys

- Root key
- Recovery key
- Worker-auth key



5.2 Configure Boundary Controller

Prepare TLS certificates: Run following command

Create a directory for SSL certificates and generate a self-signed certificate:

```
sudo mkdir /etc/ssl/boundary
```

Generate a private key with the following command:

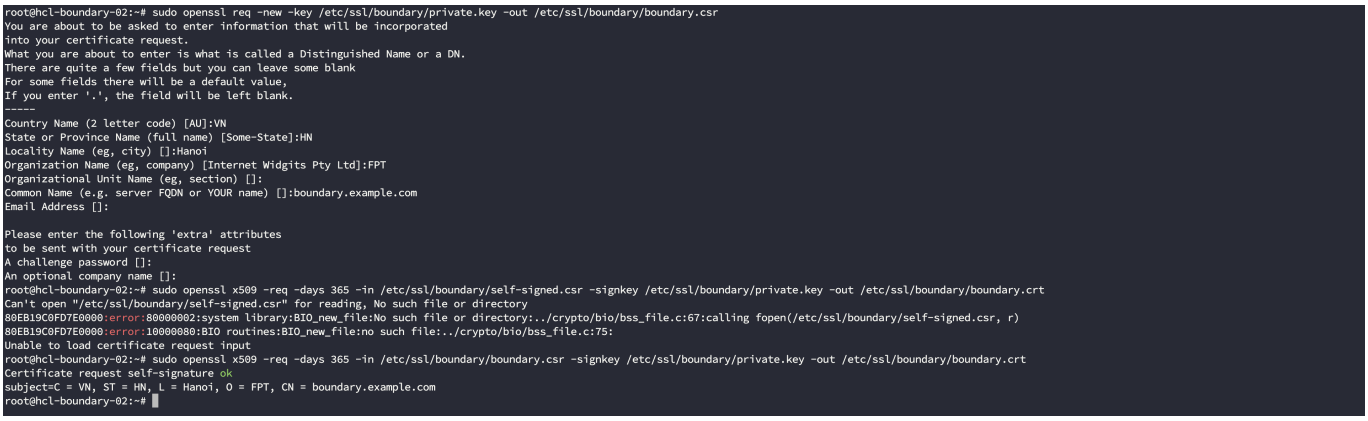
```
sudo openssl genrsa -out /etc/ssl/boundary/private.key 2048
```

Run this command to create a CSR. You'll be prompted to enter some information.

```
sudo openssl req -new -key /etc/ssl/boundary/private.key -out
/etc/ssl/boundary/boundary.csr
```

Use the following command to create the self-signed certificate:

```
sudo openssl x509 -req -days 365 -in /etc/ssl/boundary/boundary.csr -
signkey /etc/ssl/boundary/private.key -out /etc/ssl/boundary/boundary.crt
```



Output

```
root@hcl-boundary-02:~# sudo openssl x509 -req -days 365 -in
/etc/ssl/boundary/boundary.csr -signkey /etc/ssl/boundary/private.key -out
/etc/ssl/boundary/boundary.crt
Certificate request self-signature ok
subject=C = VN, ST = HN, L = Hanoi, O = FPT, CN = boundary.example.com
```

Edit Configuration File `/etc/boundary.d/boundary.env` :

```
POSTGRESQL_CONNECTION_STRING=postgresql://boundary:********@127.0.0.1:5432
/boundary
```

Reload OS environment

```
source /etc/boundary.d/boundary.env
```

Base Controller Configuration Create the boundary.hcl file as per requirements.

```
# disable memory from being swapped to disk
disable_mlock = true

# API listener configuration block
listener "tcp" {
  # Should be the address of the NIC that the controller server will be
reached on
  # Use 0.0.0.0 to listen on all interfaces
  address = "127.0.0.1:9200"
  # The purpose of this listener block
  purpose = "api"

  # TLS Configuration
  tls_disable   = false
  tls_cert_file = "/etc/ssl/boundary/boundary.crt"
```

```
  tls_key_file  = "/etc/ssl/boundary/private.key"

  # Uncomment to enable CORS for the Admin UI. Be sure to set the allowed
origin(s)
  # to appropriate values.
  #cors_enabled = true
  #cors_allowed_origins = ["https://yourcorp.yourdomain.com",
"serve://boundary"]
}

# Data-plane listener configuration block (used for worker coordination)
listener "tcp" {
  # Should be the IP of the NIC that the worker will connect on
  address = "127.0.0.1:9201"
  # The purpose of this listener
  purpose = "cluster"
}

# Ops listener for operations like health checks for load balancers
listener "tcp" {
  # Should be the address of the interface where your external systems'
  # (eg: Load-Balancer and metrics collectors) will connect on.
  address = "127.0.0.1:9203"
  # The purpose of this listener block
  purpose = "ops"

  tls_disable   = false
  tls_cert_file = "/etc/ssl/boundary/boundary.crt"
  tls_key_file  = "/etc/ssl/boundary/private.key"
}

# Controller configuration block
controller {
  # This name attr must be unique across all controller instances if
running in HA mode
  name = "boundary-controller-1"
  description = "Boundary controller number one"

  # This is the public hostname or IP where the workers can reach the
  # controller. This should typically be a load balancer address
  public_cluster_addr = "controller.example.com"

  # Enterprise license file, can also be the raw value or env:// value
  #license = "file:///path/to/license/file.hclic"

  # After receiving a shutdown signal, Boundary will wait 10s before
initiating the shutdown process.
  graceful_shutdown_wait_duration = "10s"

  # Database URL for postgres. This is set in boundary.env and
  #consumed via the “env://” notation.
  database {
     url = "env://POSTGRESQL_CONNECTION_STRING"
  }
```

```
}

# Events (logging) configuration. This
# configures logging for ALL events to both
# stderr and a file at /var/log/boundary/controller.log
events {
  audit_enabled       = true
  sysevents_enabled   = true
  observations_enable = true
  sink "stderr" {
    name = "all-events"
    description = "All events sent to stderr"
    event_types = ["*"]
    format = "cloudevents-json"
  }
  sink {
    name = "file-sink"
    description = "All events sent to a file"
    event_types = ["*"]
    format = "cloudevents-json"
    file {
      path = "/var/log/boundary"
      file_name = "controller.log"
    }
    audit_config {
      audit_filter_overrides {
        sensitive = "redact"
        secret    = "redact"
      }
    }
  }
}

# Root KMS Key (managed by AWS KMS in this example)
# Keep in mind that sensitive values are provided via ENV VARS
# in this example, such as access_key and secret_key
#
kms "azurekeyvault" {
  purpose        = "root"
  tenant_id      = "tenant-id"
  client_id      = "client_id"
  client_secret  = "client_secret"
  vault_name     = "cariad-kms"
  key_name       = "vault1"
}
kms "azurekeyvault" {
  purpose        = "recovery"
  tenant_id      = "tenant-id"
  client_id      = "client_id"
  client_secret  = "client_secret"
  vault_name     = "cariad-kms"
  key_name       = "vault2"
}
kms "azurekeyvault" {
```

```
    purpose         = "worker-auth"
    tenant_id       = "tenant-id"
    client_id       = "client_id"
    client_secret   = "client_secret"
    vault_name      = "cariad-kms"
    key_name        = "vault3"
  }
```

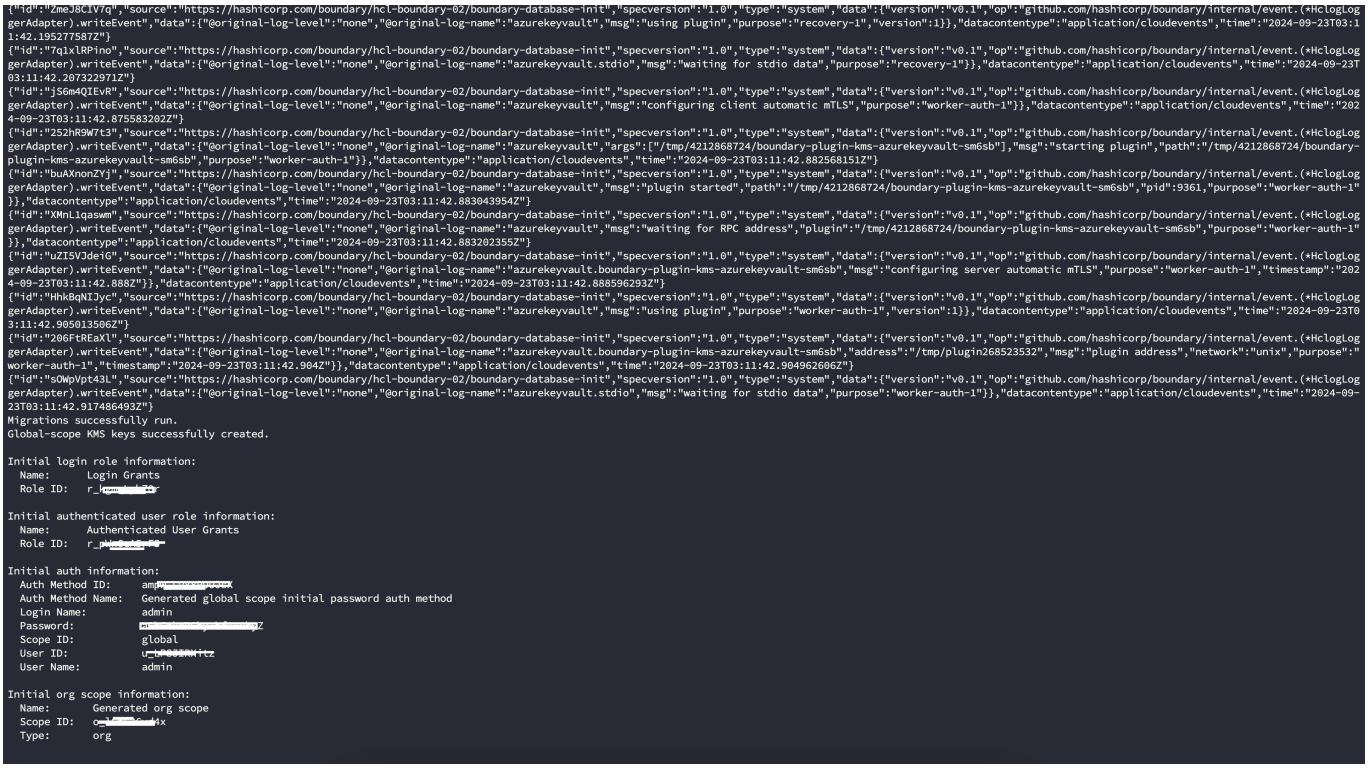
Please remember to change value of `tenant_id`, `client_id`, `client_secret`, `vault_name`, `key_name`.

5.3 Run the Boundary

Initialize the Database

```
boundary database init -config /etc/boundary.d/boundary.hcl
```

The output look like. You will get the initial credential from output console. Please save them to use later



Start Boundary Service

First all first, We need to update File Permissions for `boundary`

```
sudo chown -R boundary:boundary /etc/boundary.d
sudo chown -R boundary:boundary /var/log/boundary
sudo chown -R boundary:boundary /etc/ssl/boundary
```

Then, enable boot-start and start the service

```
sudo systemctl enable boundary
sudo systemctl start boundary
```

Verify the status:

```
sudo systemctl status boundary
```

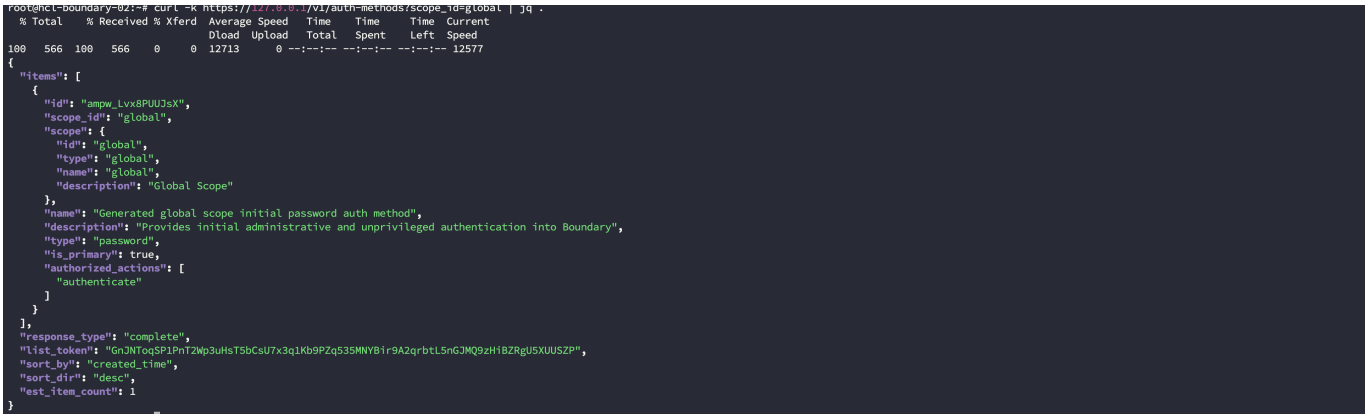
Output:

```
root@hcl-boundary-02:/etc/apache2/sites-available# systemctl status
boundary
● boundary.service - "HashiCorp Boundary - Identity-based access
management for dynamic infrastructure"
     Loaded: loaded (/lib/systemd/system/boundary.service; enabled; vendor
preset: enabled)
     Active: active (running) since Mon 2024-09-23 03:29:52 UTC; 33min ago
       Docs: https://www.boundaryproject.io/docs
   Main PID: 10134 (boundary)
      Tasks: 47 (limit: 9519)
     Memory: 292.6M
        CPU: 38.152s
     CGroup: /system.slice/boundary.service
             ├─10134 /usr/bin/boundary server -
config=/etc/boundary.d/boundary.hcl
             ├─10141 /tmp/3372176301/boundary-plugin-kms-azurekeyvault-
gmMTT
             ├─10149 /tmp/3012504364/boundary-plugin-kms-azurekeyvault-
wSSDg
             ├─10158 /tmp/3225556827/boundary-plugin-kms-azurekeyvault-
roA0R
             ├─10168 /tmp/2276464319/boundary-plugin-aws-bMDcj
             └─10175 /tmp/3612182203/boundary-plugin-azure-aXead

Sep 23 03:29:56 hcl-boundary-02 boundary[10134]:
{"id":"1kWltL3NTO","source":"https://hashicorp.com/boundary/hcl-boundary-
02/controller","specversion":"1.0","type":"system","data":
{"version":"v0.1","op":"github.com/hashic>
Sep 23 03:29:56 hcl-boundary-02 boundary[10134]:
{"id":"W3acCR1pHE","source":"https://hashicorp.com/boundary/hcl-boundary-
02/controller","specversion":"1.0","type":"system","data":
{"version":"v0.1","op":"github.com/hashic>
Sep 23 03:29:56 hcl-boundary-02 boundary[10134]:
{"id":"mYAlwSax9z","source":"https://hashicorp.com/boundary/hcl-boundary-
02/controller","specversion":"1.0","type":"system","data":
{"version":"v0.1","op":"github.com/hashic>
Sep 23 03:29:56 hcl-boundary-02 boundary[10134]:
{"id":"nAordNphSw","source":"https://hashicorp.com/boundary/hcl-boundary-
02/controller","specversion":"1.0","type":"system","data":
```

```
{"version":"v0.1","op":"github.com/hashic>
Sep 23 03:29:56 hcl-boundary-02 boundary[10134]:
{"id":"v8bjYAXfKL","source":"https://hashicorp.com/boundary/hcl-boundary-
02/controller","specversion":"1.0","type":"system","data":
{"version":"v0.1","op":"github.com/hashic>
Sep 23 03:29:56 hcl-boundary-02 boundary[10134]:
{"id":"K3tih9EXZR","source":"https://hashicorp.com/boundary/hcl-boundary-
02/controller","specversion":"1.0","type":"system","data":
{"version":"v0.1","op":"github.com/hashic>
Sep 23 03:29:56 hcl-boundary-02 boundary[10134]:
{"id":"4bWoHYLDLY","source":"https://hashicorp.com/boundary/hcl-boundary-
02/controller","specversion":"1.0","type":"system","data":
{"version":"v0.1","op":"server.RotateRoot>
Sep 23 03:56:48 hcl-boundary-02 boundary[10134]:
{"id":"Fn2WdPWJO1","source":"https://hashicorp.com/boundary/hcl-boundary-
02/controller","specversion":"1.0","type":"audit","data":
{"id":"e_b5AE2jVvVG","version":"v0.1","typ>
Sep 23 03:57:30 hcl-boundary-02 boundary[10134]:
{"id":"rmRZFgKYFQ","source":"https://hashicorp.com/boundary/hcl-boundary-
02/controller","specversion":"1.0","type":"audit","data":
{"id":"e_h821ZA6Mw3","version":"v0.1","typ>
Sep 23 03:59:07 hcl-boundary-02 boundary[10134]:
{"id":"nGsu5tFWSO","source":"https://hashicorp.com/boundary/hcl-boundary-
02/controller","specversion":"1.0","type":"audit","data":
{"id":"e_Hh506rUocN","version":"v0.1","typ>
lines 1-26/26 (END)
```

Verify with `Boundary API`

```
curl -k https://127.0.0.1/v1/auth-methods?scope_id=global
```



5.4 Config reverse proxy

Create config file `/etc/apache2/site-avaiable/boundary.conf` with content below to public Boundary Management Web UI

```
<IfModule mod_ssl.c>
    <VirtualHost *:443>
        ServerName boundary.example.com
        ServerAdmin admin@example.com

        ErrorLog ${APACHE_LOG_DIR}/boundary.example.com_error.log
        CustomLog ${APACHE_LOG_DIR}/boundary.example.com_access.log
combined

        ProxyPass / https://127.0.0.1:9200/

        ProxyRequests Off

        SSLEngine on
        SSLProxyEngine on
        SSLProxyVerify none
        SSLProxyCheckPeerCN off
        SSLProxyCheckPeerName off

        SSLCertificateFile /etc/ssl/boundary/boundary.crt
        SSLCertificateKeyFile /etc/ssl/boundary/private.key

        #<LocationMatch "^/">
        #    Require all denied
        #</LocationMatch>

        #<LocationMatch "^/v1/auth-methods|^/v1/targets">
        #    Require all granted
        #</LocationMatch>
    </VirtualHost>
</IfModule>
```

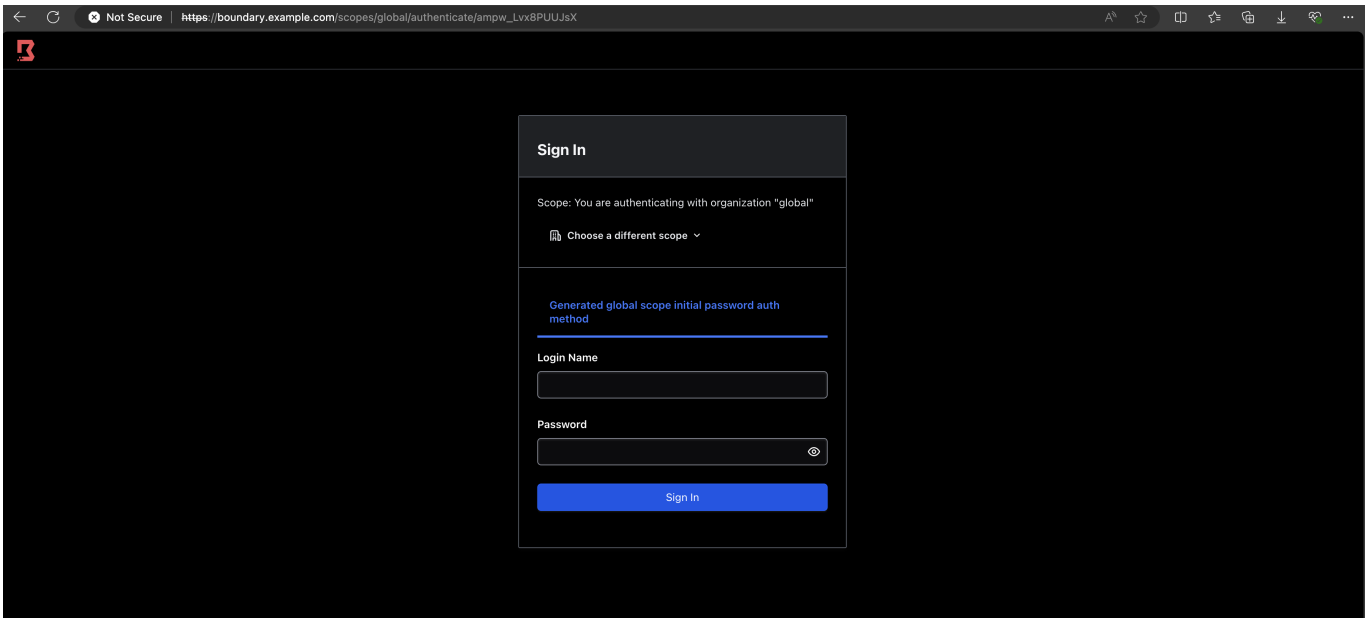
Enable boundary site

```
a2ensite boundary
```

Then reload Apache service to apply the change

```
systemctl reload apache2
```

Check access on browser:



Login with `Initial auth information` that we got from step **5.3**

5.5 Create Boundary Worker

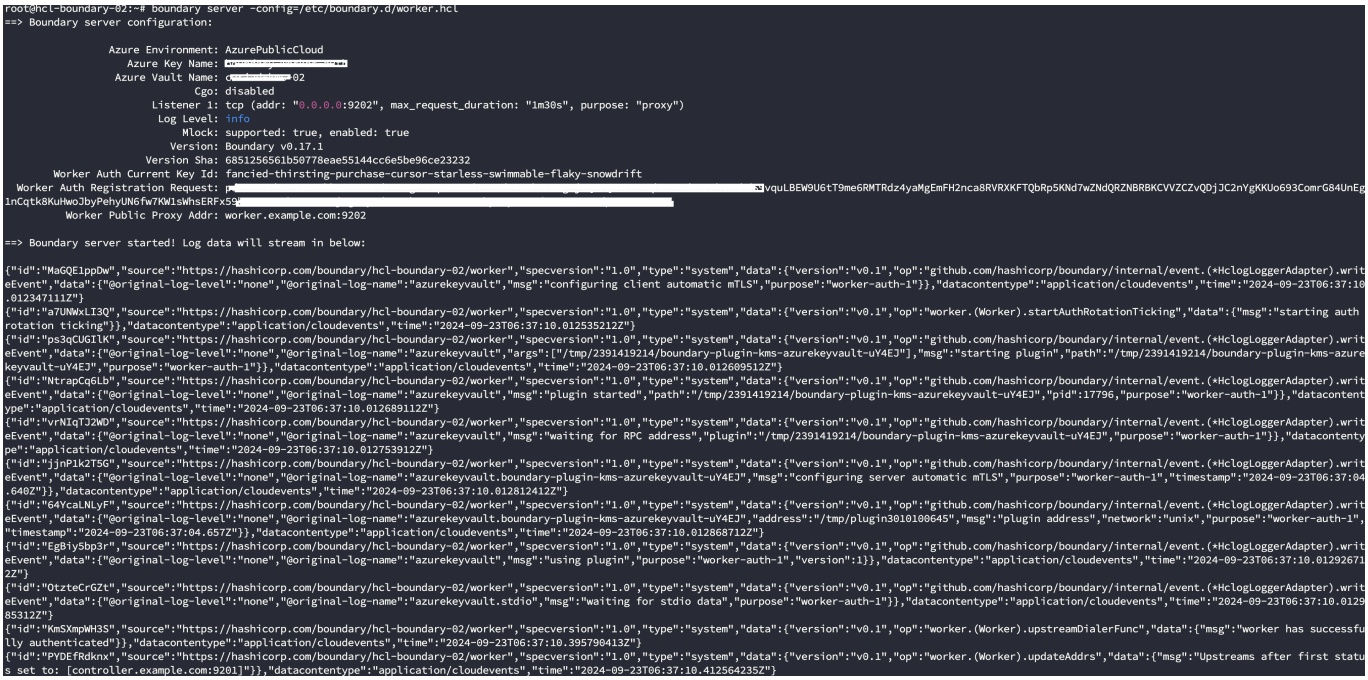
Create worker configuration file as bellow:

```
listener "tcp" {
  address = "0.0.0.0:9202"
  purpose = "proxy"
}

# worker block for configuring the specifics of the
# worker service
worker {
  name = "worker1"
  public_addr = "worker.example.com"
  initial_upstreams = ["controller.example.com:9201"]
  tags {
    type = ["worker", "egress"]
  }
}
kms "azurekeyvault" {
  purpose        = "worker-auth"
  tenant_id      = "tenant_id"
  client_id      = "client_id"
  client_secret  = "client_secret"
  vault_name     = "cariad-kms"
  key_name       = "boundary-worker-auth"
}
```

Please remember to update `kms` information before deploying

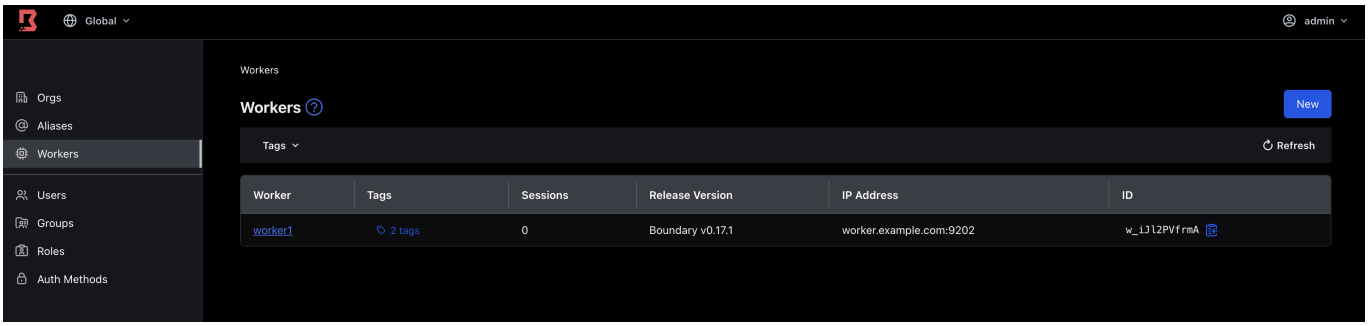
Deploy the worker in same VM

```
boundary server -config=/etc/boundary.d/worker.hcl &
```



The Worker will be connected to `Boundary Controller` aotumatically

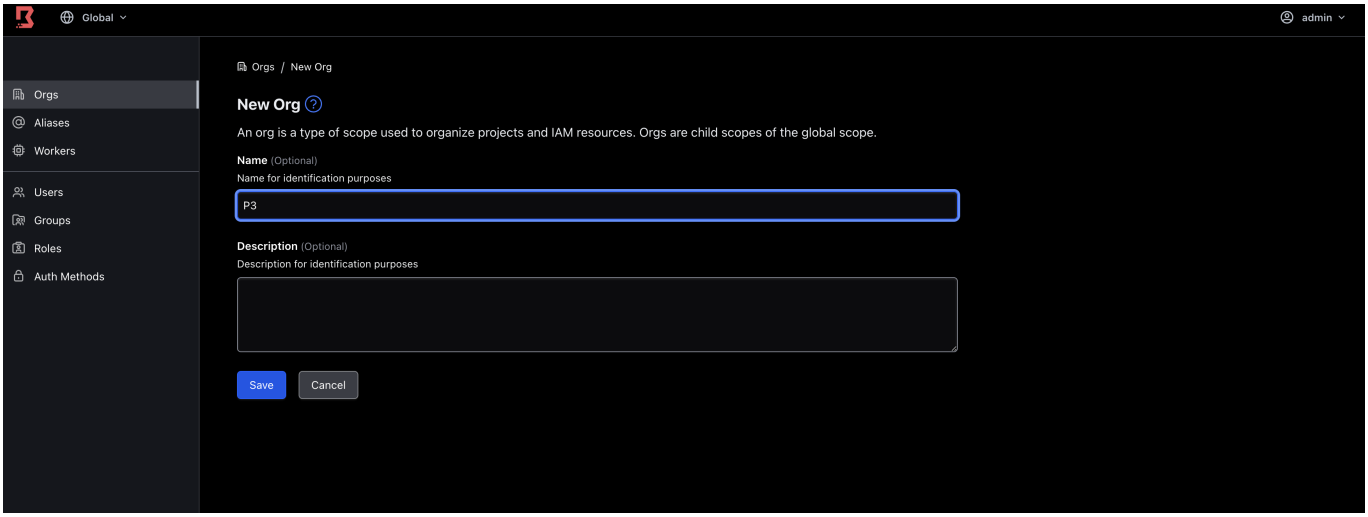
From Web Admin UI, select Worker, you will see the worker is connected



6. Create user, target

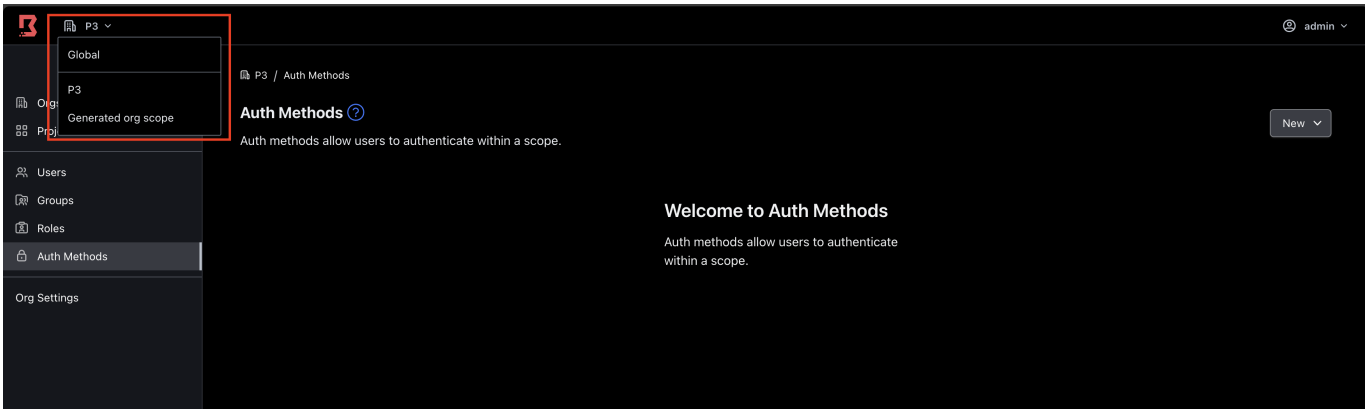
6.1 Create new Org

We will create a new `org` with name `P3`

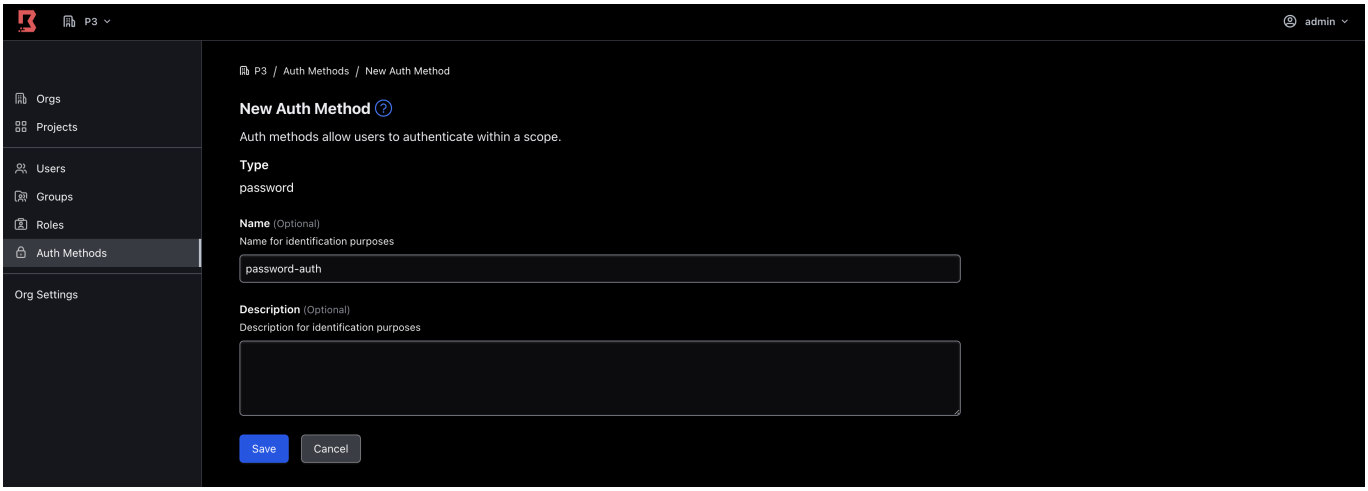


6.2. Create Authentication method

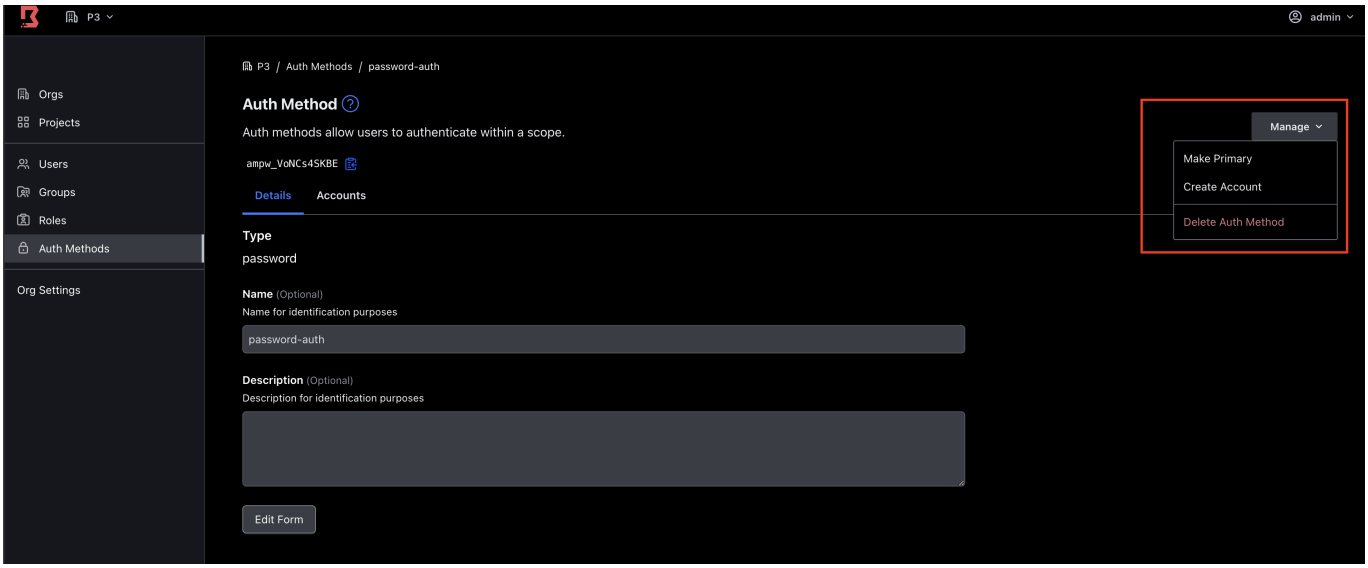
Go to `P3` org



Select `Auth Methods` and create a new **Password Auth Methods**

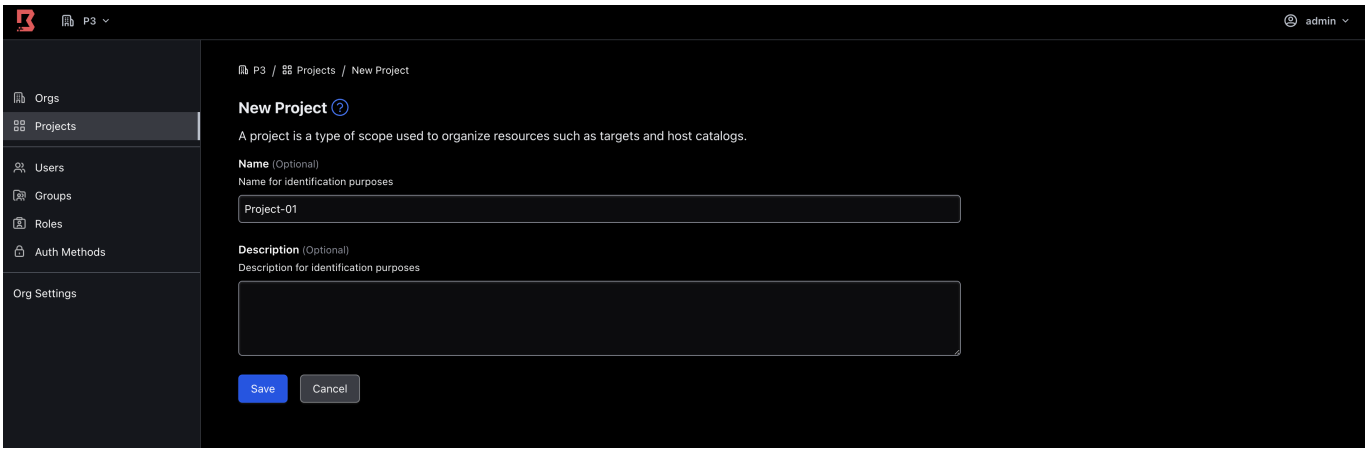


Then make `password-auth` as primary Auth Methods



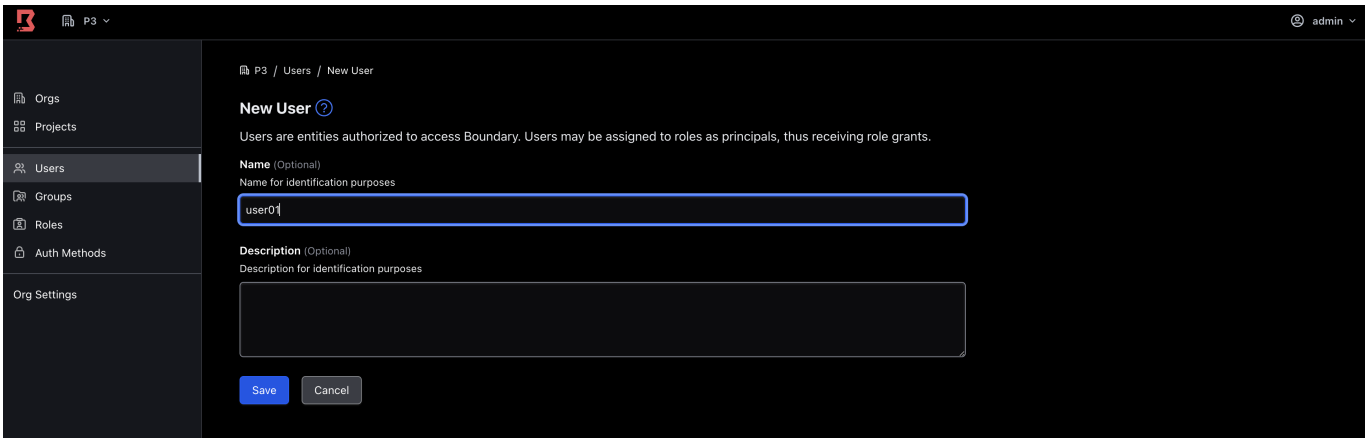
6.3 Create new Project

We will create new project named `Project-01`



6.4. Create new User

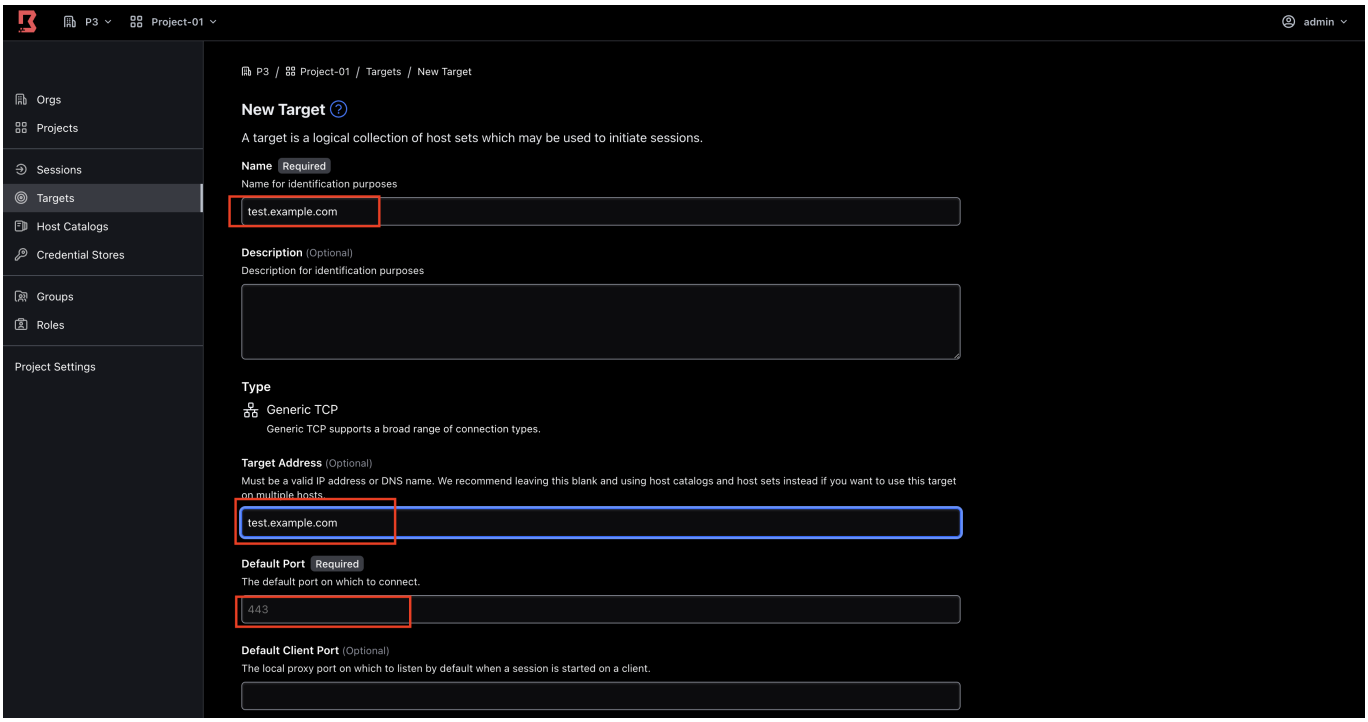
Go to **Users** and create new one



6.5. Add new target

Next, we will create new target, that will be pointed to dummy website with domain `test.example.com`

Go to `Project-01`, then create new **Targets**

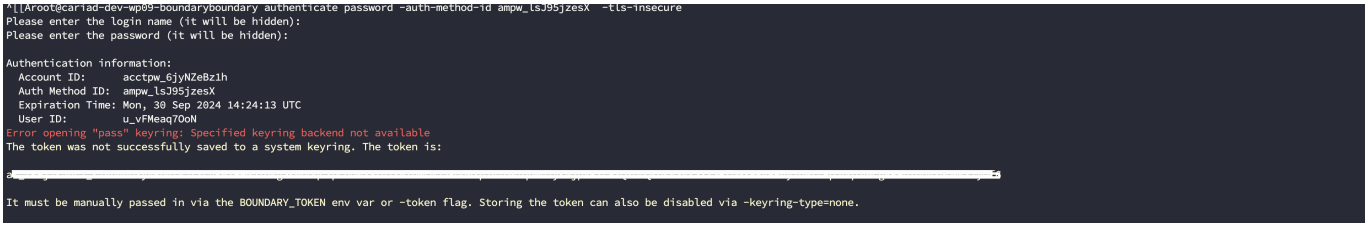


6.6. Connect the target

Login in CLI

```
export BOUNDARY_ADDR=https://boundary.example.com
boundary authenticate password -auth-method-id <method-id> --tls-insecure
```

Please get auth method id that we got from step **6.2**



Save token to env

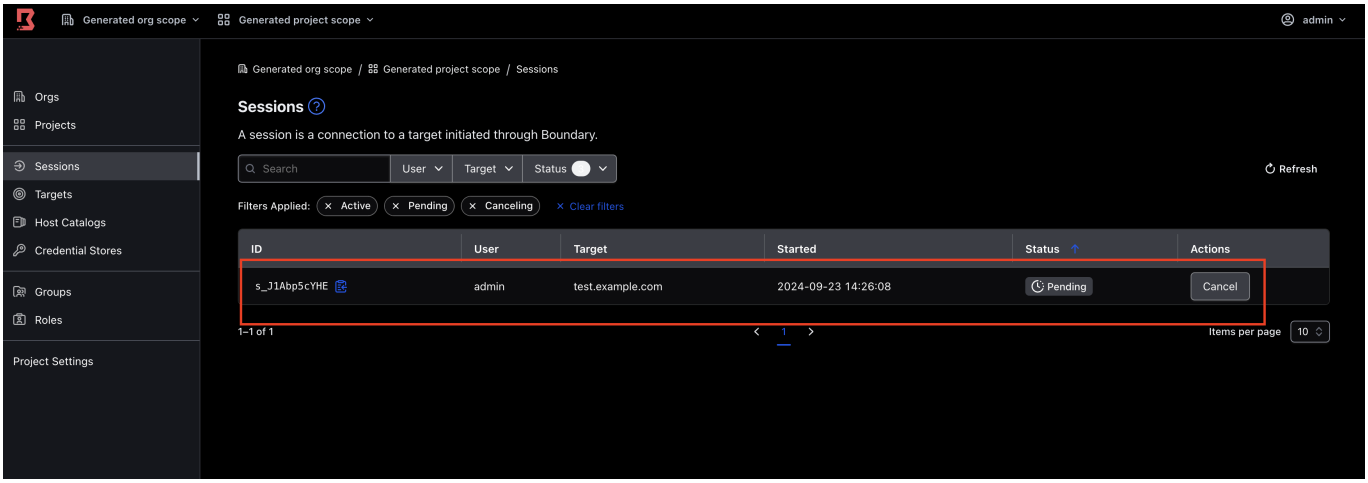
```
export BOUNDARY_TOKEN=<token>
```

Connect to target

```
boundary connect -target-id <target-id> -token env://BOUNDARY_TOKEN --tls-
insecure
```

```
root@cariad-dev-wp09-boundary:~# boundary connect -target-id ttcp_Lcp210ISs1 -token env://BOUNDARY_TOKEN -tls-insecure

Proxy listening information:
  Address:             127.0.0.1
  Connection Limit:    0
  Expiration:          Mon, 23 Sep 2024 22:26:08 UTC
  Port:                41205
  Protocol:            tcp
  Session ID:          s_J1Abp5cYHE
```

Check session on Web UI



7. Custom Boundary Client

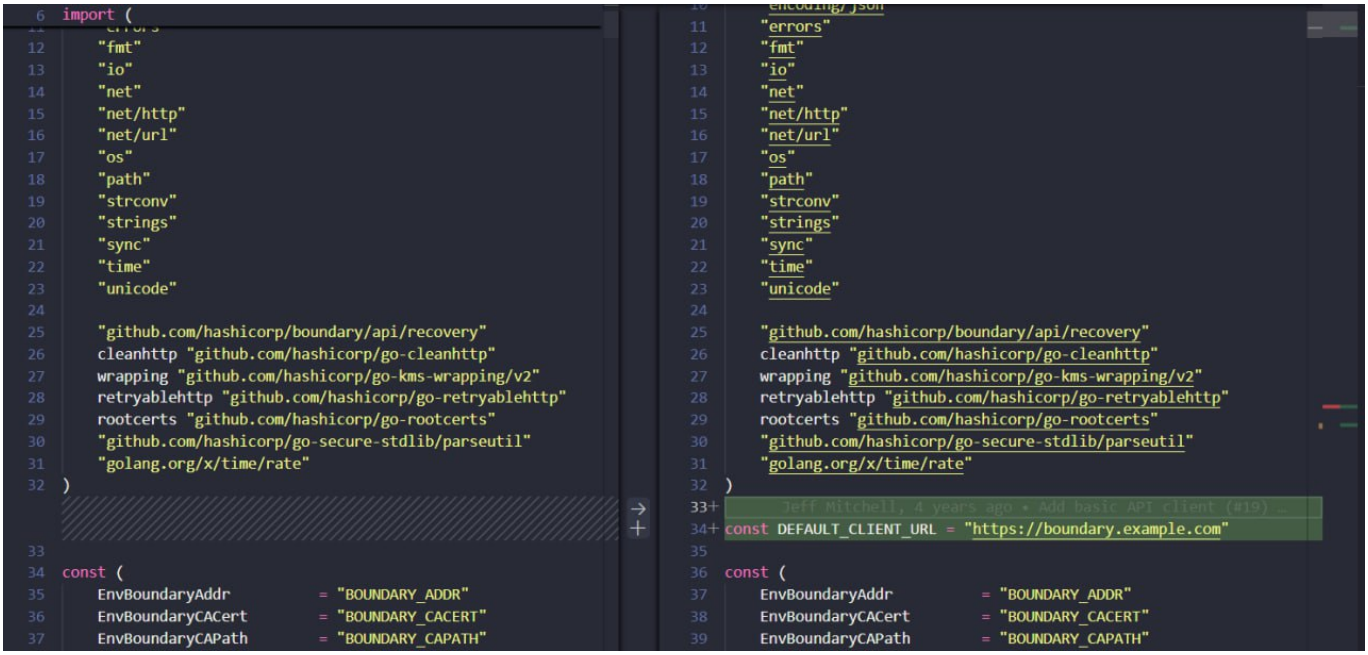
Download Boundary source code from github Boundary Source

```
git clone https://github.com/hashicorp/boundary.git
```

7.1. Hardcode Boundary URL

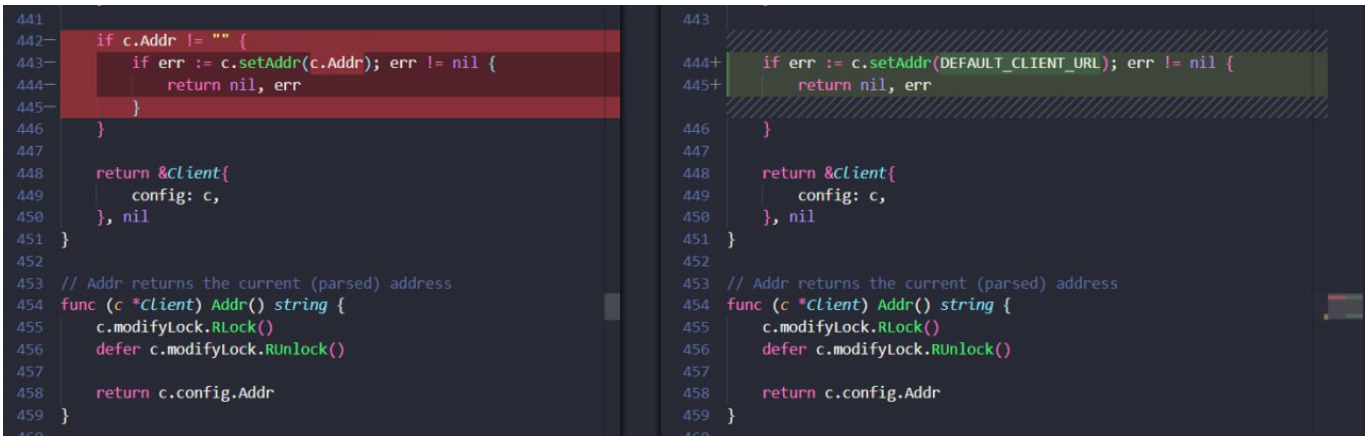
Open file `/api/client.go`, add content to `line 33-34`

```
const DEFAULT_CLIENT_URL="https://boundary.example.com
```



Change the code from `line 442-445` to

```
  if err:=c.setAddr(DEFAULT_CLIENT_URL); err != nil {
    return nil, err
  }
```



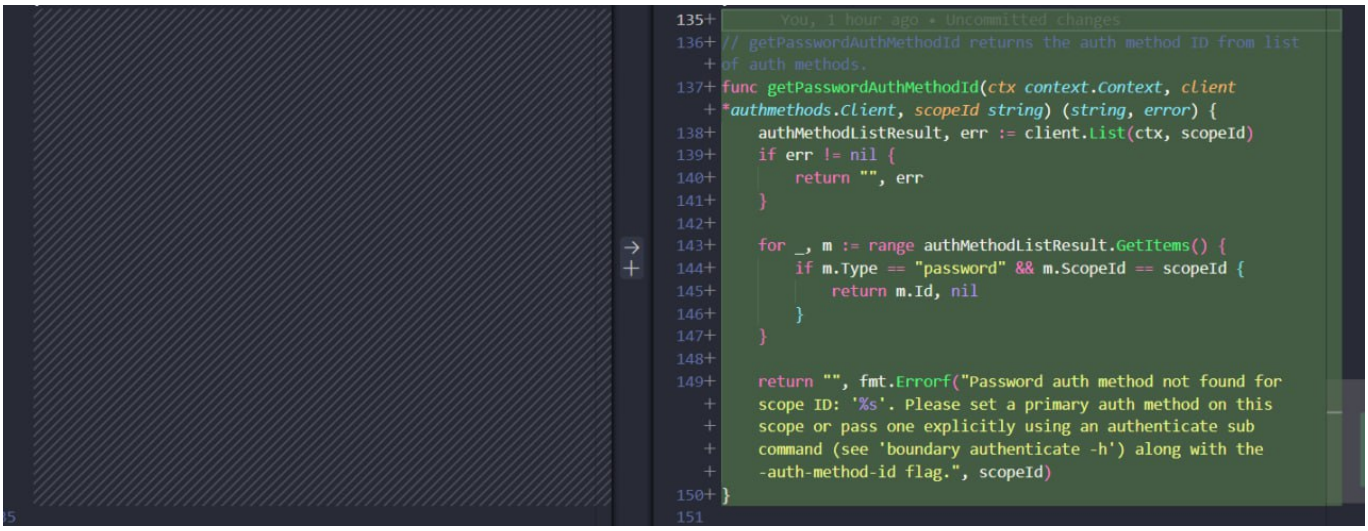
```
441
442-    if c.Addr != "" {
443-        if err := c.setAddr(c.Addr); err != nil {
444-            return nil, err
445-        }
446     }
447
448     return &Client{
449         config: c,
450     }, nil
451 }
452
453 // Addr returns the current (parsed) address
454 func (c *Client) Addr() string {
455     c.modifyLock.RLock()
456     defer c.modifyLock.RUnlock()
457
458     return c.config.Addr
459 }
```

```
443
444+     if err := c.setAddr(DEFAULT_CLIENT_URL); err != nil {
445+         return nil, err
446     }
447
448     return &Client{
449         config: c,
450     }, nil
451 }
452
453 // Addr returns the current (parsed) address
454 func (c *Client) Addr() string {
455     c.modifyLock.RLock()
456     defer c.modifyLock.RUnlock()
457
458     return c.config.Addr
459 }
```

7.2. Modify the code to fetch Auth Method ID aotumatically

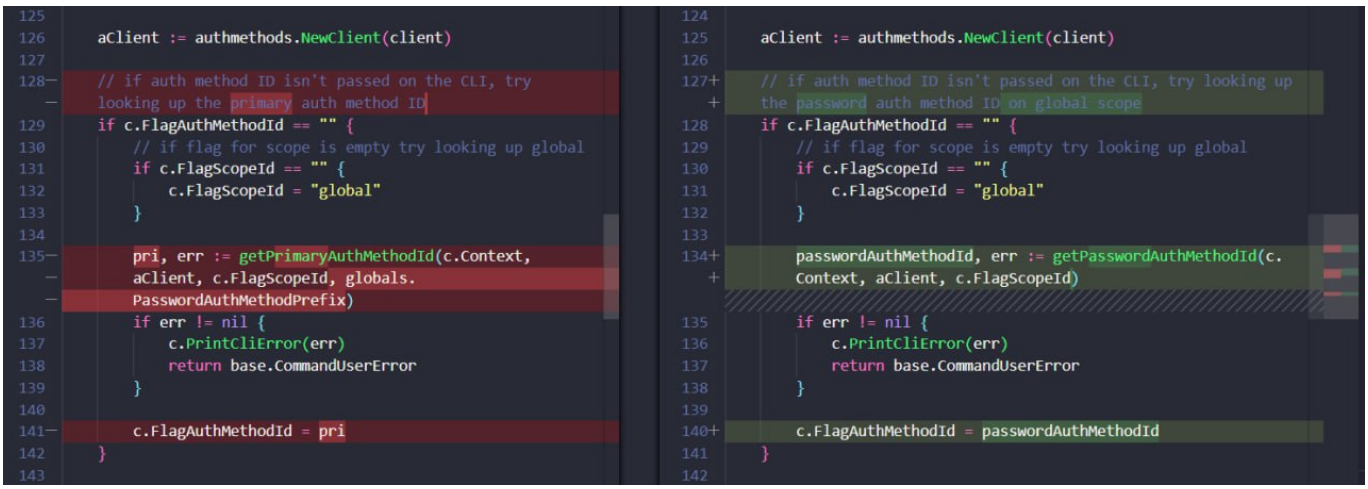
Open the file `internal/cmd/commands/authenticate/funcs.go` and paste the code in from `line 135`

```
/ getPasswordAuthMethodId returns the auth method ID from list of auth
methods.
func getPasswordAuthMethodId(ctx context.Context, client
*authmethods.Client, scopeId string) (string, error) {
  authMethodListResult, err := client.List(ctx, scopeId)
  if err != nil {
    return "", err
  }

  for _, m := range authMethodListResult.GetItems() {
    if m.Type == "password" && m.ScopeId == scopeId {
      return m.Id, nil
    }
  }

  return "", fmt.Errorf("Password auth method not found for scope ID:
'%s'. Please set a primary auth method on this scope or pass one
explicitly using an authenticate sub command (see 'boundary authenticate -
h') along with the -auth-method-id flag.", scopeId)
}
```



```
135+        You, 1 hour ago • Uncommitted changes
136+ // getPasswordAuthMethodId returns the auth method ID from list
   + of auth methods.
137+ func getPasswordAuthMethodId(ctx context.Context, client
   + *authmethods.Client, scopeId string) (string, error) {
138+     authMethodListResult, err := client.List(ctx, scopeId)
139+     if err != nil {
140+         return "", err
141+     }
142+
143+     for _, m := range authMethodListResult.GetItems() {
144+         if m.Type == "password" && m.ScopeId == scopeId {
145+             return m.Id, nil
146+         }
147+     }
148+
149+     return "", fmt.Errorf("Password auth method not found for
   +     scope ID: '%s'. Please set a primary auth method on this
   +     scope or pass one explicitly using an authenticate sub
   +     command (see 'boundary authenticate -h') along with the
   +     -auth-method-id flag.", scopeId)
150+ }
151
```

Open the file `internal/cmd/commands/authenticate/password.go` and change the code as following



```
125
126     aClient := authmethods.NewClient(client)
127
128-    // if auth method ID isn't passed on the CLI, try
   -    looking up the primary auth method ID
129     if c.FlagAuthMethodId == "" {
130         // if flag for scope is empty try looking up global
131         if c.FlagScopeId == "" {
132             c.FlagScopeId = "global"
133         }
134
135-        pri, err := getPrimaryAuthMethodId(c.Context,
   -        aClient, c.FlagScopeId, globals.
   -        PasswordAuthMethodPrefix)
136         if err != nil {
137             c.PrintCliError(err)
138             return base.CommandUserError
139         }
140
141-        c.FlagAuthMethodId = pri
142     }
143
```

```
124
125     aClient := authmethods.NewClient(client)
126
127+    // if auth method ID isn't passed on the CLI, try looking up
   +    the password auth method ID on global scope
128     if c.FlagAuthMethodId == "" {
129         // if flag for scope is empty try looking up global
130         if c.FlagScopeId == "" {
131             c.FlagScopeId = "global"
132         }
133
134+        passwordAuthMethodId, err := getPasswordAuthMethodId(c.
   +        Context, aClient, c.FlagScopeId)
135         if err != nil {
136             c.PrintCliError(err)
137             return base.CommandUserError
138         }
139
140+        c.FlagAuthMethodId = passwordAuthMethodId
141     }
142
```

7.3 Re-build the Boundary Client

Re-build boundary client with command

```
CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -o boundary
cmd/boundary/main.go
```

Then you will get new `boundary` binary

7.4 Test new Boundary Client

Copy new binary to `/tmp`

Grant execution permission for binary

```
chmod +x /tmp/boundary
```

Run test command

```
./boundary authenticate password -tls-insecure
```

As you can see, we can connect to the Boundary without entering the authen method id