**CyberArk CCP ↔ AWX/Ansible — Step‑by‑Step Guide:**

This is a **single-method** guide using **CyberArk CCP (Central Credential Provider****).**

**What is CCP?**

* **CCP is a web API** from CyberArk that **gives passwords/keys on demand**.
* It checks **who you are** (via **AppID** + runner identity) and **what you can read** (via **Safe** and **Object**).
* If allowed, CCP **returns the secret**. Your job uses it **in memory** and moves on.

**We do NOT store secrets** in Git or AWX DB. We **fetch** them just-in-time.

**Why use CCP?**

* **Security:** No hard-coded passwords in playbooks or repos.
* **Rotation friendly:** CyberArk rotates passwords. Our jobs still work (we always fetch latest).
* **Audit:** Every retrieval is logged in CyberArk.
* **Simple:** Works with Ansible’s built-in modules (no extra plugin required).

**What to check with CyberArk team**

1. **Endpoints & trust**

* **CCP URL** (example: <https://ccp>.company.com/AIMWebService/api/Accounts)
* **CA certificate chain** (so our runner trusts the HTTPS cert)

**B. Identity**

* **AppID** string we must use (example: awx-dev)
* Confirm our **AWX runner identity** is allowed for that AppID (IP/hostname/cert)

**C. Secrets by environment**

* For **Server** login: **Safe** + **Object** name
* For **DB** login: **Safe** + **Object** name
* For **App/API token**: **Safe** + **Object** name
* Share the **naming convention** (so we can template more later)

**D. Policies & operations**

* **Rotation schedule** for each secret
* Any **Dual Control/approval** needed?
* **HA/DR** details for CCP (SLA, maintenance windows)
* How to access **audit logs** for retrievals

We will **paste** these values into AWX.

**Below values we expect to receive from CyberARc team**

* CCP\_URL = <https://ccp>.company.com/AIMWebService/api/Accounts
* APPID = awx-dev
* **Server (Linux)**: SAFE = APP-POC-EC2-DEV, OBJECT = ec2-dev-svc\_ansible
* **DB (Postgres)**: SAFE = APP-POC-DB-DEV, OBJECT = postgres-app-user
* **App/API**: SAFE = APP-POC-API-DEV, OBJECT = deploy-api-key
* **CA**: company CA bundle file

**High-level workflow:**

[AWX Job Template]  
 |  
 | (inject CCP URL, AppID, Safe, Object as ENV VARS)  
 v  
[Playbook] --builds URL--> https://ccp/... ?AppID=...&Safe=...&Object=...  
 |  
 | (HTTPS request with company CA)  
 v  
[CyberArk CCP] -> checks AppID + runner identity + Safe/Object perms  
 |  
 | (if allowed)  
 v  
[Secret returned] -> used in memory (no printing) -> task runs (SSH/DB/API)  
 |  
 v  
[CyberArk Audit] (retrieval logged) [AWX Job Log] (no secret value)

**One‑time AWX setup**

**Create a Credential Type**

AWX → **Administration → Credential Types → Add**

* **Name:** CyberArk CCP Secret
* **Input Configuration**

fields:  
 - id: ccp\_url  
 type: string  
 label: CCP URL  
 - id: appid  
 type: string  
 label: AppID  
 - id: safe  
 type: string  
 label: Safe  
 - id: object  
 type: string  
 label: Object  
 - id: verify\_ssl  
 type: boolean  
 label: Verify SSL  
 default: true  
required:  
 - ccp\_url  
 - appid  
 - safe  
 - object

* **Injector Configuration**

env:  
 CYBR\_CCP\_URL: '{{ ccp\_url }}'  
 CYBR\_APPID: '{{ appid }}'  
 CYBR\_SAFE: '{{ safe }}'  
 CYBR\_OBJECT: '{{ object }}'  
 CYBR\_VERIFY\_SSL: '{{ verify\_ssl | default(true) }}'

* **Save**

Create a **Credential** (per use case)

AWX → **Resources → Credentials → Add**

* **Credential Type:** CyberArk CCP Secret
* Fill fields using values from CyberArk (URL, AppID, Safe, Object)
* Keep **Verify SSL = true** (after CA is installed in your EE)
* **Save**

Create one credential for **Server**, one for **DB**, one for **API** (reuse the same type).

Project & Inventory

* AWX **Projects**: point to your Git repo with playbooks
* AWX **Inventory**: add hosts (for server/db jobs). For localhost-only tasks, inventory can be a dummy.

Job Template

AWX → **Templates → Add → Job Template**

* Select **Inventory**, **Project**, **Playbook**
* **Execution Environment**: must reach CCP URL over 443 and trust company CA
* **Credentials**: add the CyberArk credential (Server/DB/API depending on the job)
* **Save**

**Smoke test**

Create ccp\_smoke.yml in your repo:

- name: CCP fetch smoke test  
 hosts: localhost  
 gather\_facts: false  
 tasks:  
 - name: Build CCP query URL from injected env vars  
 set\_fact:  
 ccp\_query: >-  
 {{ lookup('env','CYBR\_CCP\_URL') }}?AppID={{ lookup('env','CYBR\_APPID') }}  
 &Safe={{ lookup('env','CYBR\_SAFE') }}&Object={{ lookup('env','CYBR\_OBJECT') }}  
  
 - name: Fetch secret from CCP  
 uri:  
 url: "{{ ccp\_query | regex\_replace('\\s','') }}"  
 method: GET  
 return\_content: true  
 validate\_certs: "{{ (lookup('env','CYBR\_VERIFY\_SSL') | default('true')) | bool }}"  
 register: cybr  
 no\_log: true  
  
 - name: Prove success without printing value  
 debug:  
 msg: "Fetched secret OK (length={{ (cybr.json.Content | default(cybr.json.Password)) | length }})"

Run it in a Job Template with your CyberArk credential attached. You should see: Fetched secret OK (length=...).

**If it fails:**

* 401/403 → ask CyberArk to fix AppID/Safe permissions
* 404 → wrong Object name
* TLS error → install CA into your Execution Environment

Server (Linux SSH)

* Goal: use CCP password to SSH to a server and run a command.

- name: Server check via CCP  
 hosts: web # your host/group  
 gather\_facts: false  
 vars:  
 ccp\_query: >-  
 {{ lookup('env','CYBR\_CCP\_URL') }}?AppID={{ lookup('env','CYBR\_APPID') }}  
 &Safe={{ lookup('env','CYBR\_SAFE') }}&Object={{ lookup('env','CYBR\_OBJECT') }}  
 tasks:  
 - name: Fetch server login password  
 uri:  
 url: "{{ ccp\_query | regex\_replace('\\s','') }}"  
 method: GET  
 return\_content: true  
 validate\_certs: true  
 register: cybr  
 no\_log: true  
  
 - set\_fact:  
 login\_pass: "{{ (cybr.json.Content | default(cybr.json.Password)) | default('') }}"  
 no\_log: true  
  
 - name: Run id as svc user  
 ansible.builtin.shell: id  
 vars:  
 ansible\_user: "svc\_ansible" # username matches the Object account  
 ansible\_password: "{{ login\_pass }}"  
 ansible\_become: true  
 ansible\_connection: ssh  
 ansible\_ssh\_common\_args: "-o StrictHostKeyChecking=no"  
 no\_log: true

Database (Postgres example)

* Goal: fetch DB user password and run a simple query.

- name: DB health via CCP  
 hosts: db  
 gather\_facts: false  
 tasks:  
 - name: Fetch DB password  
 uri:  
 url: "{{ lookup('env','CYBR\_CCP\_URL') }}?AppID={{ lookup('env','CYBR\_APPID') }}&Safe=APP-POC-DB-DEV&Object=postgres-app-user"  
 method: GET  
 return\_content: true  
 validate\_certs: true  
 register: dbp  
 no\_log: true  
  
 - set\_fact:  
 db\_pass: "{{ (dbp.json.Content | default(dbp.json.Password)) | default('') }}"  
 no\_log: true  
  
 - name: Select 1  
 community.postgresql.postgresql\_query:  
 db: appdb  
 login\_user: app\_user  
 login\_password: "{{ db\_pass }}"  
 query: "select 1;"  
 no\_log: true

*(MySQL variant: use* ***community.mysql.mysql\_query*** *with* ***login\_user****/****login\_password****.)*

Application/API token

* Goal: fetch an API token and call an internal API.

- name: Deploy via API token from CCP  
 hosts: localhost  
 gather\_facts: false  
 tasks:  
 - name: Fetch API token  
 uri:  
 url: "{{ lookup('env','CYBR\_CCP\_URL') }}?AppID={{ lookup('env','CYBR\_APPID') }}&Safe=APP-POC-API-DEV&Object=deploy-api-key"  
 method: GET  
 return\_content: true  
 validate\_certs: true  
 register: apir  
 no\_log: true  
  
 - set\_fact:  
 api\_token: "{{ (apir.json.Content | default(apir.json.Password)) | default('') }}"  
 no\_log: true  
  
 - name: Call release endpoint  
 uri:  
 url: "https://deploy.internal/api/v1/release"  
 method: POST  
 headers:  
 Authorization: "Bearer {{ api\_token }}"  
 body\_format: json  
 body:  
 version: "1.0.0"  
 no\_log: true

Rotation, audit, and guardrails

* **Rotation:** CyberArk rotates by policy. We always fetch latest → jobs keep working.
* **Audit:** CyberArk logs each retrieval; AWX logs the job (never the secret value).
* **Masking:** Use no\_log: true on any task that touches secrets.
* **RBAC:** Separate Safes per env; separate AWX credentials per env.

Test plan

1. **Smoke test** (localhost) — proves fetch works.
2. **Server test** (SSH) — run id on a dev host using CCP password.
3. **DB test** — run select 1; using CCP password.
4. **API test** — call an internal endpoint with CCP token.
5. **Rotation test** — CyberArk rotates one secret; re-run all three jobs → should pass with no AWX change.
6. **Negative tests** — wrong Object (404), no permission (401/403), TLS fail → errors are clear; no secret printed.