AnsibleGuy WebUI

AnsibleGuy

May 20, 2024

USAGE

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

ONE

1 - INTRO

1.1 Comparison

There are multiple Ansible WebUI products - how do they compare to this product?

- Ansible AWX / Ansible Automation Platform

  If you want an enterprise-grade solution - you might want to use these official products.

  They have many neat features and are designed to run in containerized & scalable environments.

  The actual enterprise solution named 'Ansible Automation Platform' can be pretty expensive.

- Ansible Semaphore

  Semaphore is a pretty lightweight WebUI for Ansible.

  It is a single binary and built from Golang (backend) and Node.js/Vue.js (frontend).

  Ansible job execution is done using custom implementation.

  The project is managed by a single maintainer and has some issues. It seems to develop in the direction of large-scale containerized deployments.

  The 'Ansible-WebUI' project was inspired by Semaphore.

- **This project**

  It is built to be lightweight.

  As Ansible already requires Python3 - I chose it as primary language.

  The backend stack is built of gunicorn and the frontend consists of Django templates and vanilla JS/jQuery.

  Ansible job execution is done using the official ansible-runner library!

  Target users are small to medium businesses and Ansible users which just want a UI to run their playbooks.

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

TWO

2 - INSTALLATION

2.1 Ansible

See the documentation on how to install Ansible.

Make sure to read the Ansible best-practices on how to use Ansible!

2.2 Demo

Check out the demo at: demo.webui.ansibleguy.net

Login: User `demo`, Password `Ansible1337`

2.3 Install

Requires Python >=3.10

```
python3 -m pip install ansibleguy-webui
```

Using docker:

```
docker image pull ansible0guy/webui:latest
```

2.4 Start

TLDR:

```
cd $PLAYBOOK_DIR
python3 -m ansibleguy-webui
```

Using docker:

```
docker run -d --name ansible-webui --publish 127.0.0.1:8000:8000 ansible0guy/webui:latest
```

Details:

See: *Usage - Run*

Now you can open the Ansible-WebUI in your browser: http://localhost:8000

2.5 Proxy

You can find a nginx config example here: Nginx config example

---

2.6 Ansible Role

You can find an Ansible Role to install the app on Debian here: ansibleguy.sw_ansible_webui

---

2.7 Service

You might want to create a service-user:

```
sudo useradd ansible-webui --shell /usr/sbin/nologin --create-home --home-dir /home/
↪ansible-webui
```

You can find a service config example here: Systemd config example

Enabling & starting the service:

```
systemctl enable ansible-webui.service
systemctl start ansible-webui.service
```

For production usage you should use a proxy like nginx in from of the Ansible-WebUI webservice!

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

THREE

3 - RUN

3.1 Getting Started

You may want to:

- Set the `AW_SECRET` environmental variable with a length of at least 30 characters!
- Provide a Playbook base-directory - either:
  - Change into the target directory before executing `python3 -m ansibleguy-webui`
  - Create *a Repository*
  - Set the `AW_PATH_PLAY` to your Playbook base-directory (env-var or via WebUI)

See: *Usage - Config* for more details

3.2 Run Locally (PIP)

```
# foreground
python3 -m ansibleguy-webui

# or background
python3 -m ansibleguy-webui > /tmp/aw.log 2> /tmp/aw.err.log &

# at the first startup you will see the auto-generated credentials:

AnsibleGuy-WebUI Version 0.0.12
[2024-02-25 20:59:47 +0100] [5302] [INFO] Using DB: <PATH-TO-DB>
[2024-02-25 20:59:47 +0100] [5302] [WARN] Initializing database <PATH-TO-DB>..
[2024-02-25 20:59:50 +0100] [5302] [WARN] No admin was found in the database!
[2024-02-25 20:59:50 +0100] [5302] [WARN] Generated user: 'ansible'
[2024-02-25 20:59:50 +0100] [5302] [WARN] Generated pwd: '<PASSWORD>'
[2024-02-25 20:59:50 +0100] [5302] [WARN] Make sure to change the password!
[2024-02-25 20:59:50 +0100] [5302] [INFO] Listening on http://127.0.0.1:8000
[2024-02-25 20:59:50 +0100] [5302] [WARN] Starting..
[2024-02-25 20:59:50 +0100] [5302] [INFO] Starting job-threads
```

3.3 Run Dockerized

```
docker run -d --name ansible-webui --publish 127.0.0.1:8000:8000 ansible0guy/webui:latest

# or with persistent data (volumes: /data = storage for logs & DB, /play = ansible␣
↪playbook base-directory)
docker run -d --name ansible-webui --publish 127.0.0.1:8000:8000 --volume $(pwd)/ansible/
↪data:/data --volume $(pwd)/ansible/play:/play ansible0guy/webui:latest

# find initial password
docker logs ansible-webui
```

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

FOUR

4 - CONFIG

4.1 WebUI

Runtime system configuration can be managed using the WebUI `System - Config` page.

4.1.1 Jobs

Jobs can be managed at the `Jobs - Manage` page.

Tip: The file-browsing also allows you to use your keyboard. Select using Up/Down/Enter keys and auto-complete input using the Tab key!

4.1.2 Permissions, Users, Groups

See: *Usage - Privileges*

4.1.3 Repositories

See: *Usage - Repositories*

4.2 Environmental variables

You can find the currently set environmental variables at the `System - Config` page.

4.2.1 Config File

You are able to provide the following settings by using a YAML config-file.

> **WARNING**: If possible - set your secrets as environmental variables. Users that are allowed to execute/modify jobs *are be able to read the content of the config-file*

- Provide it by flag: `python3 -m ansibleguy-webui -c /etc/ansible-webui/config.yml`
- Provide it by env-var: `AW_CONFIG=/etc/ansible-webui/config.yml`

Example config:

```
# env-vars without the 'AW_' prefix
DB: '/etc/ansible-webui/aw.db'
PORT: 8000
SECRET: 'sflsjklfdsjlfsDlNDIDEÜNfsnfa-ehöajklsfnn,sf,sdfs,i3uo'
HOSTNAMES: ['webui.ansibleguy.net', 'cname.webui.ansibleguy.net']
RUN_TIMEOUT: 600
SSH_KNOWN_HOSTS: '/etc/ansible-webui/known_hosts'

AUTH: 'saml'
SAML:
    METADATA_AUTO_CONF_URL: 'https://<YOUR-IDP>/metadata'
    ...
```

Docker usage:

```
# safe config to /etc/ansible-webui/config.yml on your host system
sudo docker run -d ... --env AW_CONFIG=/etc/aw/config.yml --volume /etc/ansible-webui/:/
↪etc/aw/ ansible0guy/webui:latest
```

4.2.2 Settings

Only Env

Some settings are only available as environmental variables.

- **AW_SECRET**

  Define a secret key to use for cookie and password encryption. By default it will be re-generated at service restart. It **has to be set** for job-secrets like passwords to be loadable after restart. It has to be **at least 30 characters** long!

  WARNING: If possible - set this secret as environmental variable. Users that are allowed to execute/modify jobs *are be able to read the content of the config-file*

- **AW_ADMIN**

  Define the user-name for the initial admin user.

- **AW_ADMIN_PWD**

  Define the password for the initial admin user.

- **AW_PROXY**

  Set if Ansible-WebUI is operated behind a proxy-server.

- **AW_HOSTNAMES**

  Set a comma-separated list of hostnames that are in use and should be trusted. If not set you might encounter 'CSRF' errors.

- **AW_DB**

  Define the path where the SQLite3 database is placed. Default: `${HOME}/.config/ansible-webui/aw.db`

- **AW_PORT**

  Port to listen on. Default: `8000`

- **AW_LISTEN**

  IP Address to listen on. Default: `127.0.0.1`

- **AW_SSL_CERT**

  Optionally provide the path to a ssl certificate to use. Use a (full-)chain if not self-signed.

  WARNING: You should use a proxy in front of this application in production setups.

- **AW_SSL_KEY**

  Optionally provide the path to an unencrypted ssl key to use.

  WARNING: You should use a proxy in front of this application in production setups.

- **AW_AUTH**

  Choose the authentication mode you want to use.

  One of `saml`, `ldap` or `local`. Default: `local`

  If the mode is set to `saml` or `ldap` - you need to define its config inside the config file.

General System Settings

These settings are also configurable using the WebUI.

- **AW_PATH_LOG**

  Define the path where full job-logs are saved. Default: `${HOME}/.local/share/ansible-webui/`

- **AW_PATH_RUN**

  Base directory for Ansible-Runner runtime files. Default: `/tmp/ansible-webui/`

- **AW_PATH_PLAY**

  Path to the Ansible base/playbook directory. Default: current working directory (*when executing ansible-webui*)

- **AW_RUN_TIMEOUT**

  Timeout for the execution of a playbook in seconds. Default: 3.600 (1h) You might want to lower this value to a sane value for your use-cases.

- **AW_SESSION_TIMEOUT**

  Timeout for WebUI sessions in seconds. Default: 43.200 (12h)

- **AW_SSH_KNOWN_HOSTS**

Define the path to the known-hosts file that should be used. You can use `${AW_PATH_PLAY}` to reference paths relative to your playbook base-directory!

Default: None - fallback to user defaults

Default in docker: `${AW_PATH_PLAY}/known_hosts`

- **AW_TIMEZONE**

  Override the timezone used. Default is the system timezone. Fallback value is `UTC` if all others are invalid.

Advanced Settings

Normal users will not have to use these.

- **AW_SERVE_STATIC**

  If defined - the built-in static-file serving is disabled. Use this if in production and a proxy like nginx is in front of the Ansible-WebUI webservice.

  Path to serve: `/static/ => ${PATH_VENV}/lib/python${PY_VERSION}/site-packages/ansible-webui/aw/static/`

- **AW_DB_MIGRATE**

  Define to disable automatic database schema-upgrades. After upgrading the module you might have to run the upgrade manually:

```
# if running non-release version
python3 -m ansibleguy-webui.manage makemigrations
python3 -m ansibleguy-webui.manage makemigrations aw

# all
python3 -m ansibleguy-webui.manage migrate
```

- **AW_ENV**

  Used in development. If unset or value is neither 'dev' nor 'staging' the webservice will be in production mode. 'staging' mode is close to production behavior.

- **AW_DEBUG**

  Enable debug output.

  This debug mode **SHOULD ONLY BE ENABLED TEMPORARILY**! It could possibly open attack vectors.

- **AW_DOCKER**

  Used to notify the software that it is running inside a docker container. Needed for listen port.

4.2.3 Usage

Environmental variables can be set before/when starting Ansible-WebUI.

With basic setup:

```
export AW_SECRET=aaaaaaaaaaaaaaaaaaaaabaaaaaaaaaaaaa
export AW_PROXY=1
python3 -m ansibleguy-webui

# OR

AW_SECRET=aaaaaaaaaaaaaaaaaaaaabaaaaaaaaaaaaa python3 -m ansibleguy-webui
```

When using Docker:

```
docker run -d --name ansible-webui --env AW_SECRET=aaaaaaaaaaaaaaaaaaaaabaaaaaaaaaaaaa --
↪env AW_PROXY=1 ...
```

When running as Systemd service:

```
# add inside the '[Service]' area of the service-config-file
EnvironmentFile=/etc/ansible-webui/env.txt

# add variables to the file
echo 'AW_SECRET=aaaaaaaaaaaaaaaaaaaaabaaaaaaaaaaaaa' >> /etc/ansible-webui/env.txt
echo 'AW_PROXY=1' >> /etc/ansible-webui/env.txt

# make sure the access is limited so your secret(s) are safe
chown root /etc/ansible-webui/env.txt
chmod 600 /etc/ansible-webui/env.txt
```

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

FIVE

ALERTS

You can use the UI at `Settings - Alerts` to create alerting rules for your jobs.

Options are:

- **User** specific rule - only you are notified
- **Group** rules - all members of a specific group are notified (*if they have the privilege to view the job*)
- **Global** rules - all users are notified (*if they have the privilege to view the job*)

There are currently two types of alerts: E-Mail and plugins.

5.1 E-Mail

You need to configure your mailserver at the `System - Config` page.

After that you can receive e-mails on job finish/failure.

Example Mail:

You can modify the email templates by setting the `Template Directory` in your system config. If you want to do so - copy the existing templates and modify them as needed. Note: the Django template syntax is required. No external css is supported.

5.2 Plugins

There is a generic alert-plugin interface for custom solutions.

Usage:

- Create a script that can be called by AW

  It will receive a file-path as system-argument 1 that points to a JSON file containing data you might want to use.

  Example JSON:

```
{
  "alert": {
    "type": "user",
    "condition": "always"
  },
  "user": {
    "name": "ansible",
    "first_name": "",
    "last_name": "",
    "email": "ansible@localhost",
```

(continues on next page)

(continued from previous page)


```
    "phone": null,
    "description": "test",
    "is_active": true,
    "last_login": 1715487270,
    "groups": []
  },
  "execution": {
    "failed": true,
    "status": "Failed",
    "job_name": "test2",
    "job_id": 1,
    "execution_id": 85,
    "user_name": "ansible",
    "time_start": 1715502006,
    "time_start_pretty": "2024-05-12 08:20:06 CEST",
    "time_fin": 1715502007,
    "time_fin_pretty": "2024-05-12 08:20:07 CEST",
    "time_duration": 0.845428,
    "time_duration_pretty": "1s",
    "error_short": null,
    "error_med": null,
    "log_url": "http://localhost:8000/ui/jobs/log?job=1",
    "log_stdout": "/home/guy/.local/share/ansible-webui/test2_2024-
↪05-12_08-20-06_ansible_stdout.log",
    "log_stdout_url": "http://localhost:8000/api/job/1/85/log?
↪type=stdout",
    "log_stderr": null,
    "log_stderr_url": null,
    "log_stdout_repo": null,
    "log_stdout_repo_url": null,
    "log_stderr_repo": null,
    "log_stderr_repo_url": null
  },
  "errors": {
    "html": [
      "<span class=\"aw-log-err\">fatal: [localhost]: FAILED! => {\
↪"changed\": false, \"msg\": \"Just failing\"}</span>\n"
    ],
    "text": [
      "fatal: [localhost]: FAILED! = {changed: false, msg: Just␣
↪failing}"
    ]
  },
  "stats": {
    "localhost": {
      "unreachable": false,
      "tasks_skipped": 3,
      "tasks_ok": 0,
      "tasks_failed": 1,
      "tasks_rescued": 0,
      "tasks_ignored": 0,
      "tasks_changed": 0
```


(continues on next page)

(continued from previous page)

```
    }
  }
}
```

- Create a plugin at `Settings - Alerts` that points to your executable
- Link the plugin in alerts
- You can use the user attributes `phone` and `description` to add user-specific information your script might need.
- Test it

5.2.1 Example Plugin

```python
#!/usr/bin/env python3

from sys import argv
from sys import exit as sys_exit
from json import loads as json_loads

with open(argv[1], 'r', encoding='utf-8') as _f:
    data = json_loads(_f.read())

# implement alerting

if data['execution']['failed']:
    # failure action

    for host, stats in data['stats'].items():
        if stats['unreachable'] or stats['tasks_failed'] > 0:
            # hosts that failed
            pass

sys_exit(0)
```

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

SIX

API

This project has a API first development approach!

To use the API you have to create an API key. You can use the UI at `Settings - API Keys` to do so.

You can also create API keys using the CLI: `python3 -m ansibleguy-webui.cli -a api-key.create -p <USER>`

6.1 Examples

Requests must have the API key set in the `X-Api-Key` header.

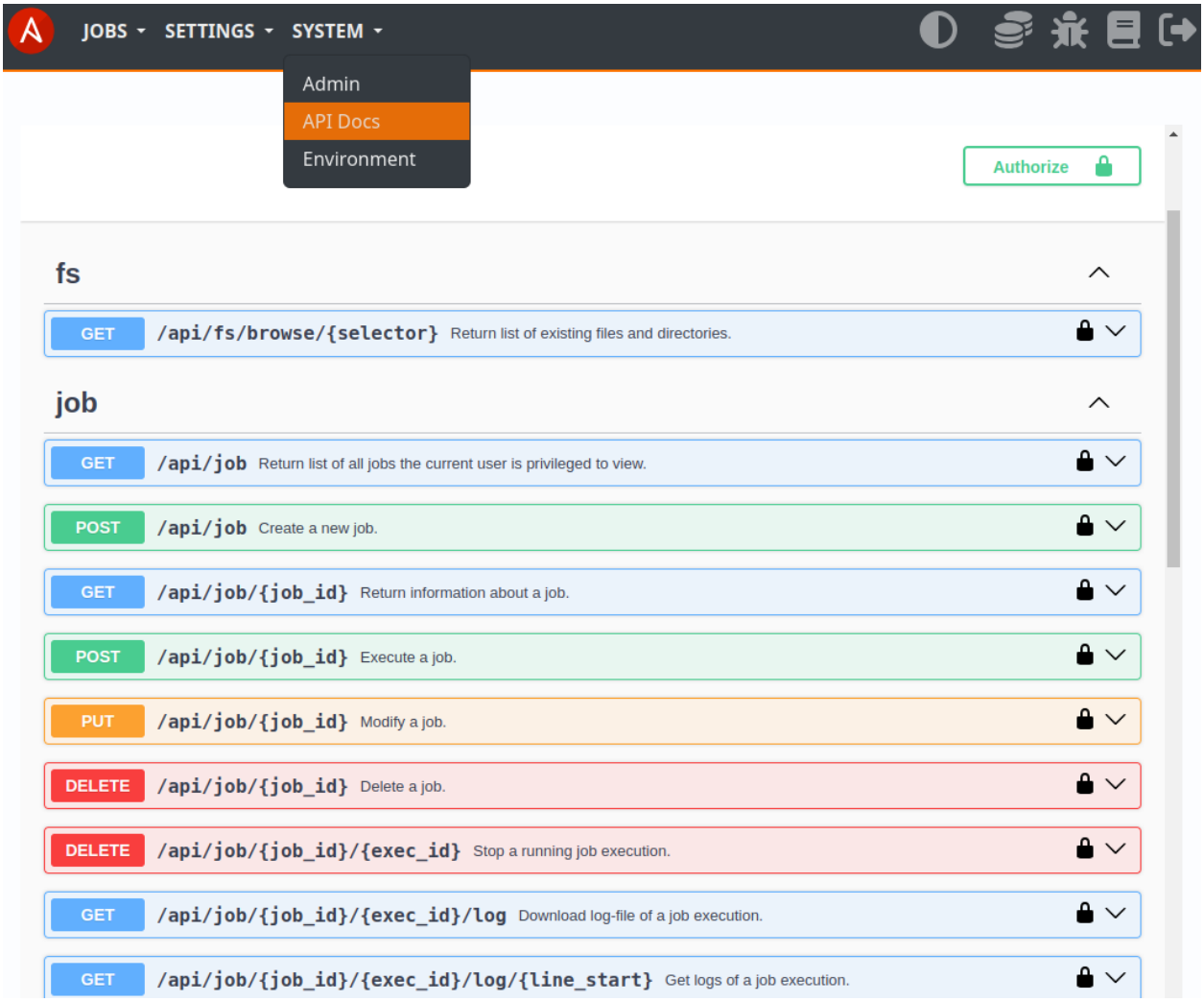
```
# list own api keys
curl -X 'GET' 'http://localhost:8000/api/key' -H 'accept: application/json' -H "X-Api-
↪Key: <KEY>"
> {"tokens":["ansible-2024-01-20-16-50-51","ansible-2024-01-20-16-10-42"]}

# list jobs
curl -X 'GET' 'http://localhost:8000/api/job' -H 'accept: application/json' -H "X-Api-
↪Key: <KEY>"
> [{"id":34,"name":"Deploy App","inventory":"inventories/dev/hosts.yml","playbook":"app.
↪yml","schedule":"22 14 * * 4,5","limit":"dev1,dev3","verbosity":0,"comment":"Deploy my␣
↪app to the first two development servers","environment_vars":"MY_APP_ENV=DEV,TZ=UTC"}]

# execute job
curl -X 'POST' 'http://localhost:8000/api/job/34' -H 'accept: application/json' -H "X-
↪Api-Key: <KEY>"
> {"msg":"Job 'Deploy App' execution queued"}
```

6.2 API Docs

You can see the available API-endpoints in the built-in API-docs at `System - API Docs` (*swagger*)



Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

SEVEN

AUTHENTICATION

In case your primary authentication method is not working for some reason - you can enter the application with a local user at: `/a/login/fallback`

7.1 SAML SSO

Tested config examples: Google Workspace

This app is integrating the grafana/django-saml2-auth module (indirect pysaml2).

If you have troubles with getting SAML to work - check out *Usage - Troubleshooting - SAML*

7.1.1 Setup

1. Add the `SAML` config-block to your config-file. See: *Usage - Config - File*

   For options see: Module settings

   Example:

```yaml
HOSTNAMES: '<YOUR-DOMAIN>'
AUTH: 'saml'
SAML:
    METADATA_AUTO_CONF_URL: 'https://<YOUR-IDP>/metadata'
    # METADATA_LOCAL_FILE_PATH: '/etc/ansible-webui/saml-metadata.txt'

    # replace with your scheme, domain and port!
    ASSERTION_URL: 'http://localhost:8000'
    ENTITY_ID: 'http://localhost:8000/a/saml/acs/'
    DEFAULT_NEXT_URL: 'http://localhost:8000/'

    CREATE_USER: true
    NEW_USER_PROFILE:
        USER_GROUPS: []  # The default group name when a new user logs in
        ACTIVE_STATUS: true
        STAFF_STATUS: true  # allow user to view 'System - Admin' page
        SUPERUSER_STATUS: false  # full system admin privileges

    ATTRIBUTES_MAP:  # email or username and token are required!
```

(continues on next page)

(continued from previous page)

```
        # mapping: django => IDP
        email: 'email'
        username: 'email'
        token: 'id'
        # optional:
        first_name: 'firstName'
        last_name: 'lastName'
        groups: 'Groups'  # Optional

    DEBUG: false  # DO NOT PERMANENTLY ENABLE!

    GROUPS_MAP:  # map IDP groups to django groups
        'IDP GROUP': 'AW Job Managers'

    # NAME_ID_FORMAT: 'user.email'
    # KEY_FILE: '/etc/ansible-webui/saml.key'
    # CERT_FILE: '/etc/ansible-webui/saml.crt'
```

2. SSO identity provider settings:

   ACS URL: `http://localhost:8000/a/saml/acs/`

   Entity ID/Audience URL: `http://localhost:8000/a/saml/acs/`

   Note: Replace *http://localhost:8000* with your scheme, domain and port

3. For non-Docker setups: Install the `xmlsec` package that is used internally (see: details)

You should now be able to see `[INFO] [main] Using Auth-Mode:  saml` logged on startup.

7.1.2 Docker

Example:

```
# save all needed SAML files to /etc/ansible-webui/ on your host system
sudo docker run -d --name ansible-webui --publish 127.0.0.1:8000:8000 --env AW_CONFIG=/
↪etc/aw/config.yml --volume /etc/ansible-webui/:/etc/aw/ ansible0guy/webui:latest
```

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

EIGHT

BACKUP

The only data to back-up is:

- Your encryption key
- The database - placed at `${HOME}/.config/ansible-webui/aw.db` or as configured
- The logs - placed at `${HOME}/.local/share/ansible-webui/` or as configured

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

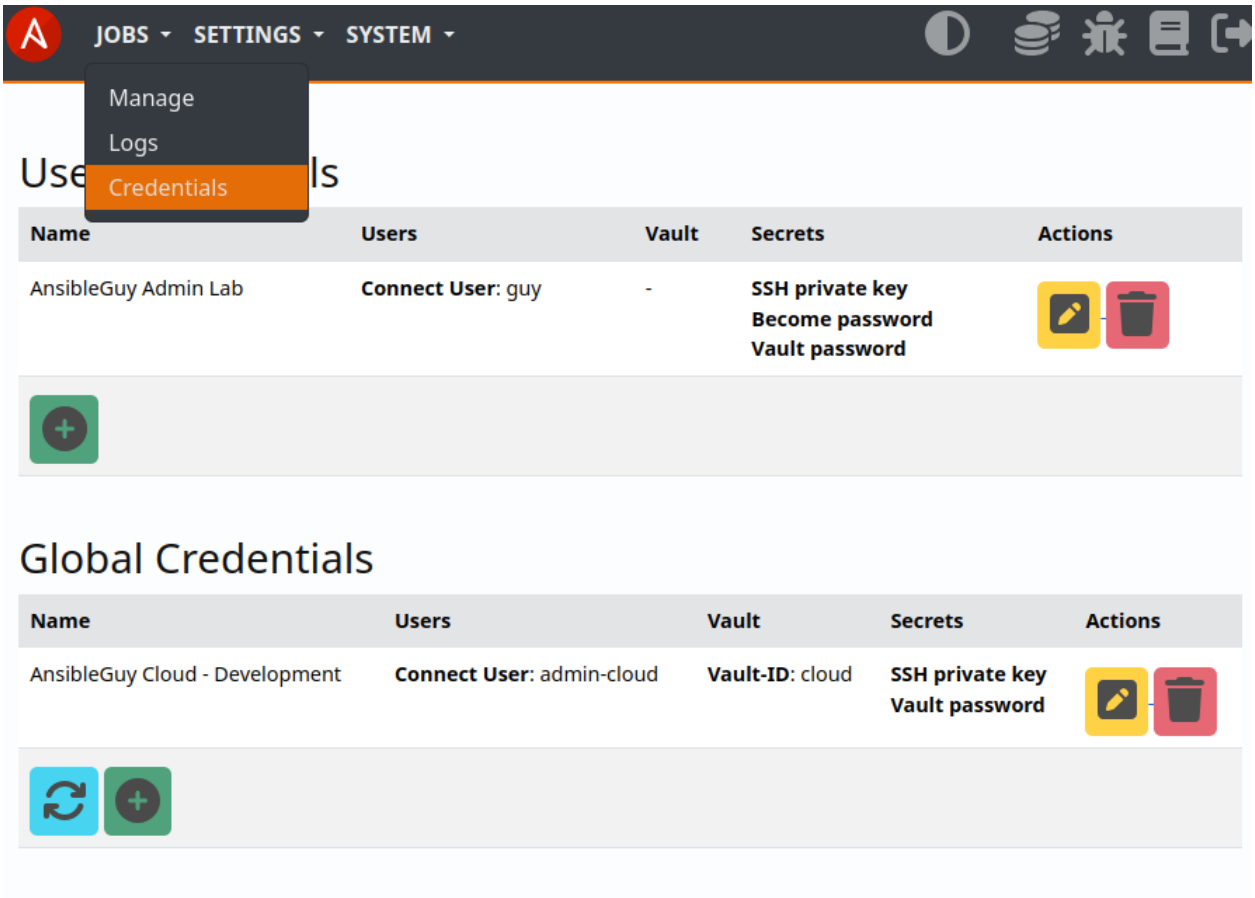
NINE

CREDENTIALS

You can define `global` and `user` credentials.

The saved credential secrets are never returned to the user/Web-UI! They are saved encrypted to the database!

The UI at `Jobs - Credentials` allows you to manage them.

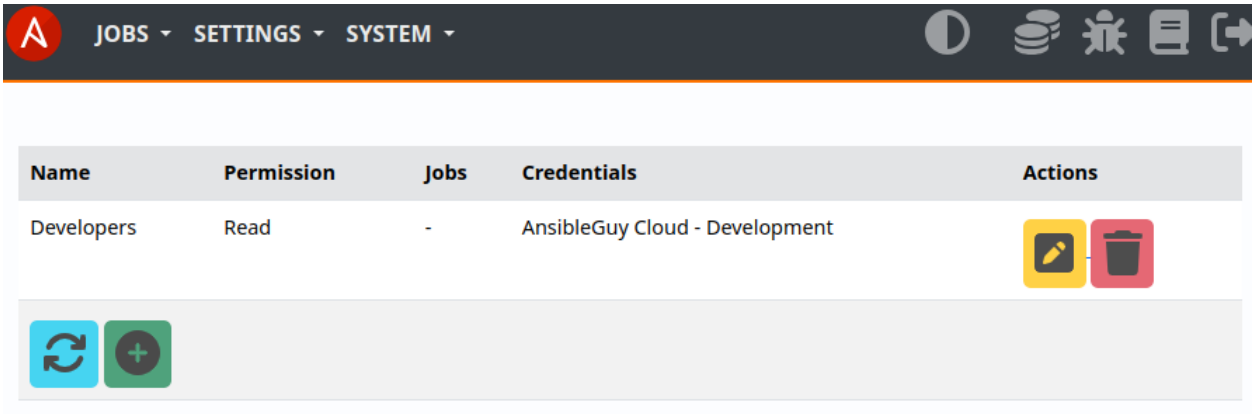


9.1 Global Credentials

Global credentials can be used for scheduled job executions.

Users that are members of the `AW Credentials Managers` group are able to create and manage global credentials.

Access to global credentials can be controlled using *permissions*.



9.2 User Credentials

User credential can only be used and accessed by the user that created them.

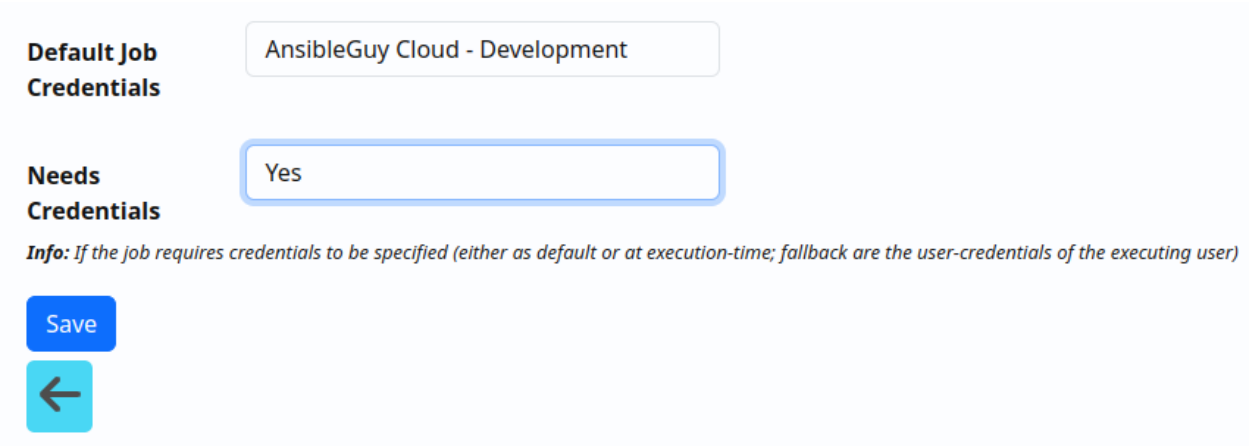
Jobs that are executed by an user will use: (*if the job is set to need credentials*)

- the user-credentials matching the jobs `credential category`
- or the first user-credentials found as a fallback in case no other credentials were provided/configured

---

9.3 Jobs

You can define if a job needs credentials to run in its settings:



Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

TEN

DEVELOPMENT

Feel free to contribute to this project using pull-requests, issues and discussions!

Testers are also very welcome! Please give feedback

For further details - see: Contribute

Read into the *Troubleshooting Guide* to get some insight on how the stack works.

10.1 Install Unstable Version

WARNING: If you run non-release versions you will have to save your `src/ansibleguy-webui/aw/migrations/*` else your database upgrades might fail. Can be ignored if you do not care about losing the Ansible-WebUI config.

```
# download
git clone https://github.com/ansibleguy/webui

# install dependencies (venv recommended)
cd webui
python3 -m pip install --upgrade requirements.txt
bash scripts/update_version.sh

# run
python3 src/ansibleguy-webui/
```

Using docker:

```
docker image pull ansible0guy/webui:unstable
docker run -it --name ansible-webui-dev --publish 127.0.0.1:8000:8000 --volume /tmp/
↪awdata:/data ansible0guy/webui:unstable
# to safe db-migrations use:
# --volume /var/local/ansible-webui/migrations/:/usr/local/lib/python3.10/site-packages/
↪ansible-webui/aw/migrations
```

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

ELEVEN

DOCKER

You can find the dockerfiles and scripts used to build the images in the Repository

11.1 Ansible Requirements

Our docker image ansible0guy/webui enables you to install Ansible dependencies on container startup.

Files inside the container:

- Python3 Modules: `/play/requirements.txt`
- Ansible Roles & Collections: `/play/requirements.yml`
  - Only Ansible Roles: `/play/requirements_roles.yml` or `/play/roles/requirements.yml`
  - Only Ansible Collections: `/play/requirements_collections.yml` or `/play/collections/requirements.yml`

11.2 Unprivileged

There are images for running Ansible-WebUI as unprivileged user `aw` with UID/GID `8785` inside the container:

- Latest: `ansible0guy/webui-unprivileged:latest`
- Unstable: `ansible0guy/webui-unprivileged:unstable`

11.3 Persistent Data

It might make sense for you to mount these paths in the container:

- `/data` (`AW_DB` & `AW_PATH_LOG` env-vars) - for database & execution-logs
- `/play` (`AW_PATH_PLAY` env-var) - for static Ansible playbook base-directory

If you are running an `unprivileged` image - you will have to allow the service-user to write to the directories. The UID needs to match!

Basic example:

```
# add matching service-user on the host system
sudo useradd ansible-webui --shell /usr/sbin/nologin --uid 8785 --user-group
chown ansible-webui:ansible-webui ${YOUR_DATA_DIR}
```

11.4 AWS CLI Support

There is also an image that has AWS-CLI support pre-enabled: `ansible0guy/webui-aws:latest` (needed for `community.aws.*` modules)

Its base-image is `ansible0guy/webui-unprivileged:latest`

11.5 Custom build

If you want to build a custom docker image - make sure to set those environmental variables:

`AW_VERSION=X.X.X AW_DOCKER=1 PYTHONUNBUFFERED=1`

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

TWELVE

INTEGRATIONS

12.1 ARA Records Ansible

ARA can be used to gather detailed statistics of Ansible executions.

To enable AW to send data to an ARA server - you need to:

- Install the `ara` Python3 module on your controller system
- Configure the server at `System - Config`

Quote: ara provides Ansible reporting by recording ansible and ansible-playbook commands regardless of how and where they run.

Documentation | Repository

12.2 Identity Providers using SAML SSO

Easily integrate with SAML2 SSO identity providers like Okta, Azure AD and others.

For configuration - see: *Usage - Authentication*

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

THIRTEEN

JOBS

You can use the UI at `Jobs – Manage` to create and execute jobs.

13.1 Create

To get an overview - Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

The job creation form will help you by browsing for playbooks and inventories. For this to work correctly - you should first select the repository to use (*if any is in use*).

You can optionally define a `schedule` in Cron-format to automatically execute the job. Schedule jobs depend on *Global Credentials* (*if any are needed*).

`Credential categories` can be defined if you want to use user-specific credentials to manage your systems. The credentials of the executing user will be dynamically matched if the job is set to `Needs credentials`.

For transparency - the full command that is executed is added on the logs-view.

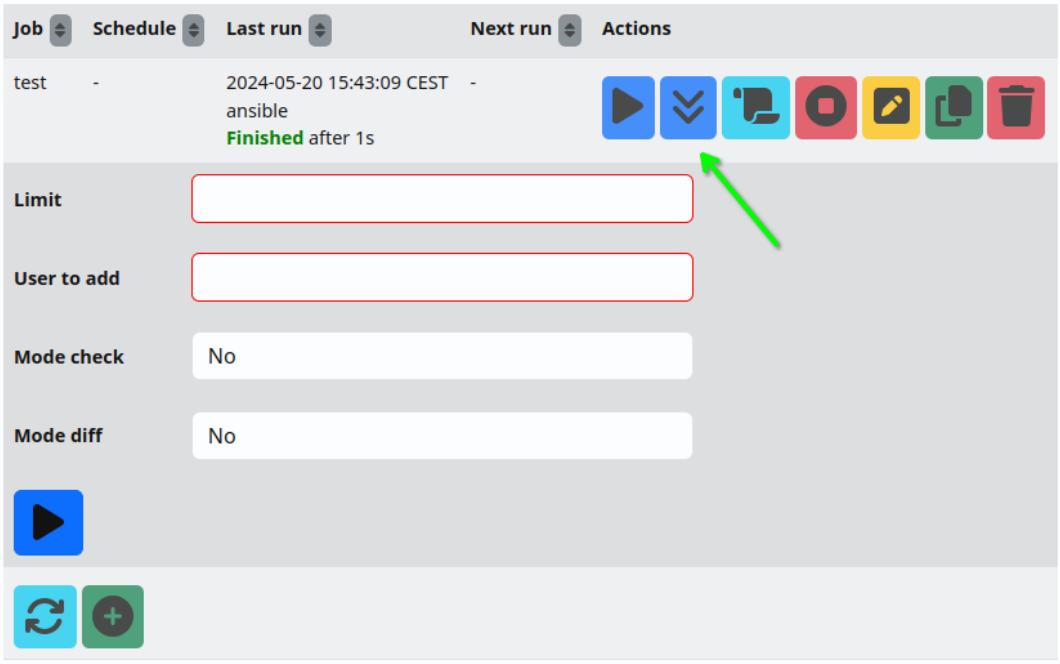
13.2 Execute

You have two options to execute a job:

- **Quick execution** - run job as configured without overrides
- **Custom execution** - run job with execution-specific overrides

  The fields available as overrides can be configured in the job settings!

  You can define required and optional overrides.



Extra-vars can also be prompted. These need to be supplied in the following format: `var={VAR-NAME}#{DISPLAY-NAME}` per example: `var=add_user#User to add`

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

FOURTEEN

PRIVILEGES

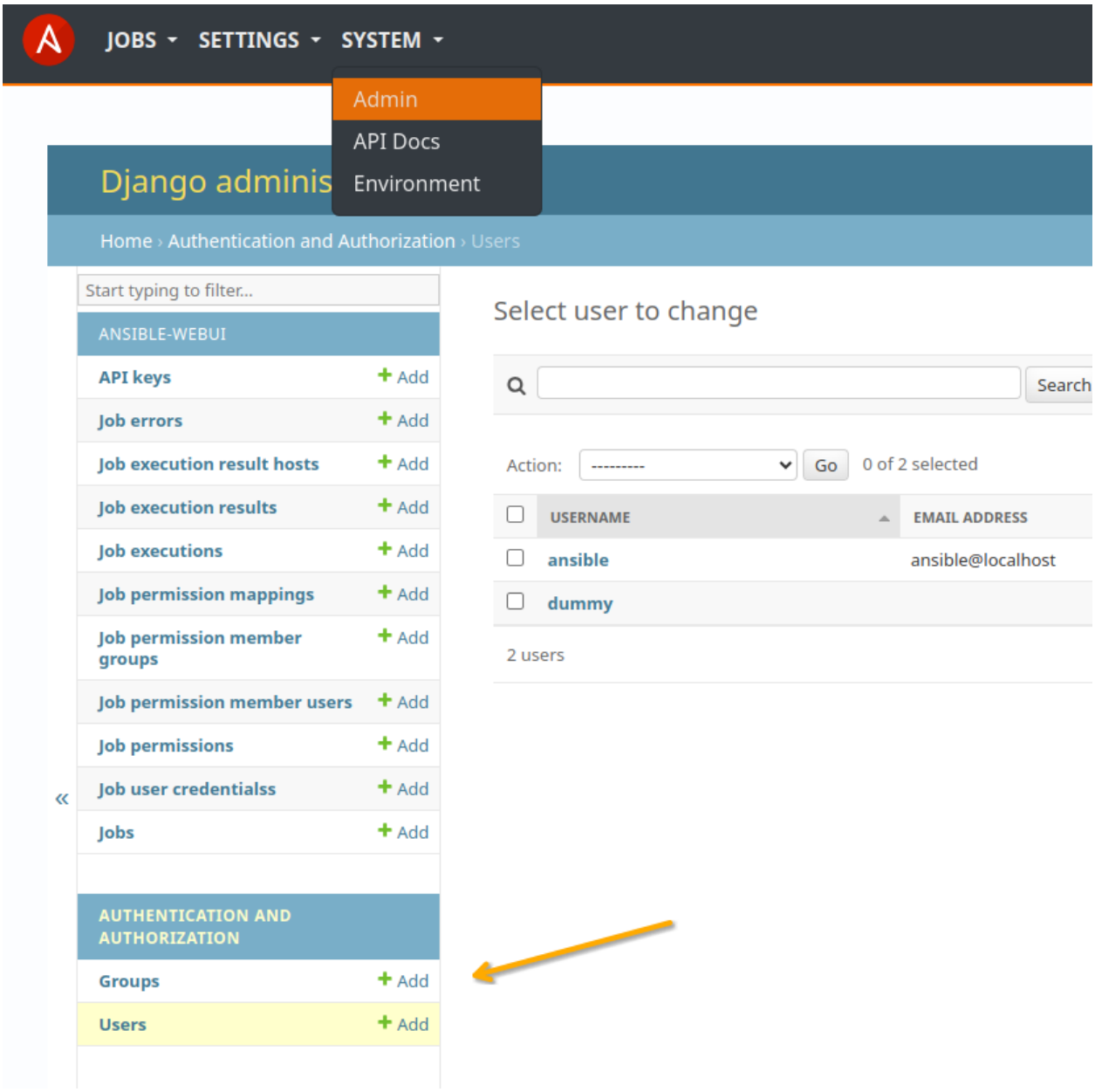
You can set permissions to limit user actions.

14.1 Users & Groups

The `System - Admin - Users/Groups` admin-page allows you to create new users and manage group memberships.

Users can change their own password at `System - Password`

The `Superuser` flag can be used to grant all privileges to a user.



14.2 Managers

To allow a users to perform management actions - add them to the corresponding system-group.

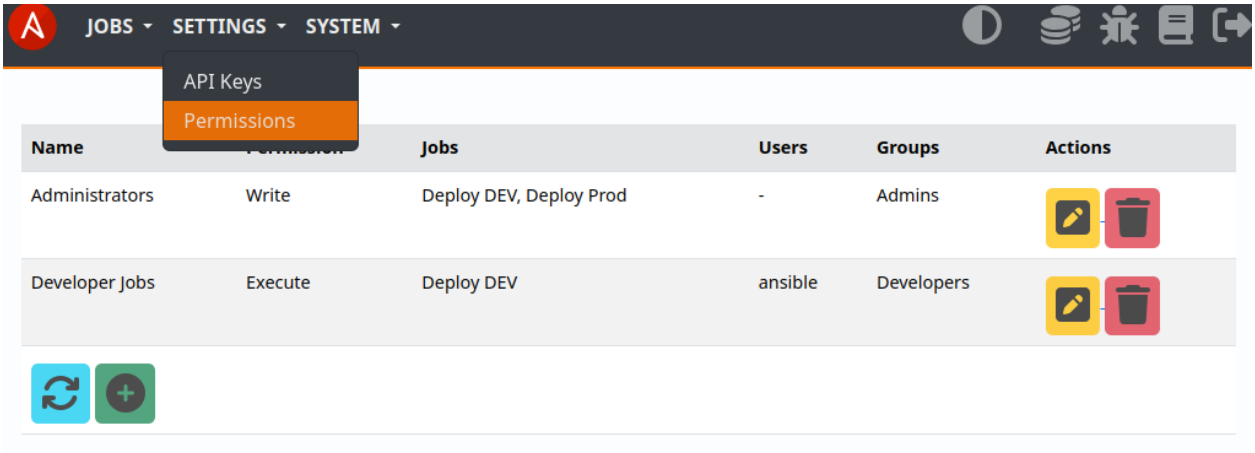
Available ones are:

- `AW Job Managers` - create new jobs, view and update all existing ones
- `AW Permission Managers` - create, update and delete permissions
- `AW Repository Managers` - create new repositories, view and update all existing ones
- `AW Credentials Managers` - create new global credentials, view and update all existing ones

- `AW System Managers` - configure system settings

14.3 Permissions

The UI at `Settings - Permissions` allows you to create job, credential & repository permissions and link them to users and groups.



Each job, credential & repository can have multiple permissions linked to it.

Permission types:

- **Read** - only allow user to read job and job-logs
- **Execute** - allow user to start & stop the job + 'Read'
- **Write** - allow user to modify the job + 'Execute'
- **Full** - allow user to delete the job + 'Write'

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

FIFTEEN

REPOSITORIES

By default the static Repository set by `AW_PATH_PLAY` is used.

You are able to create multiple Repositories that act as Ansible-Playbook base-directories.

15.1 Static

Absolute path to an existing local static directory that contains your playbook directory structure.

15.2 Git

Git repositories are also supported.

They can either be updated at execution or completely re-created (*isolated*).

The timeout for any single git-command is 5min.

15.2.1 Override commands

If you have some special environment or want to tweak the way your repository is cloned - you can override the default git-commands!

Default commands:

Create

```
git clone --branch ${BRANCH} (--depth ${DEPTH}) ${ORIGIN}
# if LFS is enabled
git lfs fetch
git lfs checkout
```

Update

```
git reset --hard
git pull (--depth ${DEPTH})
# if LFS is enabled
git lfs fetch
git lfs checkout
```

15.2.2 Hook commands

You are able to run some hook-commands before and after updating the repository.

If you want to run multiple ones - they need to be comma-separated.

These hooks will not be processed if you override the actual create/update command.

15.2.3 Clone via SSH

You can specify which `known_hosts` file AW should use using the *System config*!

You are able to append the port to the origin string like so: `git@git.intern -p1337`

The SSH-key configured in the linked credentials will be used.

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

> **Warning:** **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

> **Warning:** This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

SIXTEEN

SECURITY

Ansible needs to handle sensible secrets like administrative passwords to function.

That's why it is very important to keep security in our mind.

You are very welcome to search for security vulnerabilities and report them!

16.1 Known Issues

- **Remote-Code-Execution on Controller**

  As mentioned in this issue the Ansible-Execution is done in the same context as the Web-Service.

  So you should be aware that every user that can supply the Web-Service with playbooks, or execute ad-hoc commands, is able to execute code in the context of the service-user.

  This **includes reading the config-file**!

  So if possible - you should set your **AW_SECRET** (*and other secrets*) as environmental variable!

  Possible future fixes:

  - Run Ansible-Runner with process-isolation (execution in container) enabled (*not yet implemented in AW*)
  - Run Ansible-Runner as dedicated user (*not yet implemented in Ansible-Runner and AW*)

16.2 Features

Security considerations this project does take into account:

- The encryption key is randomized at startup by default - if none was provided by the user.
- The encryption key has to be at least 30 characters long
- Job secrets like passwords are stored encrypted (*AES256-CBC*)
- Job secrets like passwords are never returned to the user/Web-UI
- Job secrets are not passed as commandline-arguments but written to files:

  Example:

```
[INFO] [play] Running job 'test': 'ansible-playbook --become-password-file /
↪tmp/ansible-webui/2024-01-26_21-14-0066101/.aw_become_pass --vault-
↪password-file /tmp/ansible-webui/2024-01-26_21-14-0066101/.aw_vault_pass -
↪i inventory/hosts.yml --limit myHost playbook1.yml'
```

These files are:

- created with mode 0600
- overwritten and deleted at execution-cleanup

- Usage of GitHub's dependabot and CodeQL

16.3 Setup

- You should use a proxy like nginx in front of AW

  Recommended Config: (Example)

  - use HTTPS with a valid certificate
  - restrict the HTTP security headers (X-Frame-Options, X-Content-Type, Content-Security-Policy and Referrer-Policy, HSTS)
  - limit the networks able to access the Web-application using your firewall(s)
  - limit the request rate on the login form `/a/*` and API `/api/*`
  - serve static files using the proxy

    ```
    /static/ => ${PATH_VENV}/lib/python${PY_VERSION}/site-packages/
    ansible-webui/aw/static/
    ```

- Make sure the Account passwords and API keys are kept/used safe

---

Tip: Check out the repository on GitHub

Check out the demo at: demo.webui.ansibleguy.net | Login: User `demo`, Password `Ansible1337`

Warning: **DISCLAIMER**: This is an **unofficial community project**! Do not confuse it with the vanilla Ansible product!

Warning: This project still in early development! **DO NOT USE IN PRODUCTION!**

CHAPTER

SEVENTEEN

TROUBLESHOOTING

17.1 Topology

AnsibleGuy WebUI is made of a few main components.

It will be beneficial for the troubleshooting process if we find out in which the error occurs.

17.2 Debugging

You can enable the debug mode at the `System - Config` page.

If that is not possible you can alternatively set the `AW_DEBUG` environmental variable.

This debug mode **SHOULD ONLY BE ENABLED TEMPORARILY**! It could possibly open attack vectors.

You might need to restart the application to apply this setting.

17.3 Versions

You can find the versions of software packages in use at the `System - Environment` page.

Alternatively you can check it from the cli: `python3 -m ansibleguy-webui.cli --version`

17.4 Job Execution

If you want to troubleshoot a job execution, you will have to find out if it is an issue with Ansible or the WebUI system.

The Ansible execution itself can fail because of some common issues:

- Unable to connect
  - Network issue
  - Wrong credentials supplied

  - Target system is mis-configured
- Controller dependencies
  - Ansible needs Python Modules and in some cases Ansible Collections and Ansible Roles to function correctly

    These need to be installed and should be up-to-date.

    You can find the current versions used by your Controller system at the `System - Environment` page
  - If you are using Docker - you can install those dependencies using requirements-files. See *Usage - Docker*
- to be continued..

17.5 Common Issues

17.5.1 SSH Hostkey Verification

Error: While executing Ansible you see: `Host key verification failed`

Problem:

- SSH has a security feature that should keep you safe from man-in-the-middle attacks which could allow the attacker to take over your SSH account/credentials.

  See also: Ansible Docs - Hostkey Verification
- As this security feature is important you **SHOULD NOT DISABLE IT IN PRODUCTION** by adding the environmental variable *ANSIBLE_HOST_KEY_CHECKING=False* to your jobs!
- In production you might want to either:
  - Maintain a list of known-good hostkeys

    You can specify which `known_hosts` file AW should use, using the config setting `AW_SSH_KNOWN_HOSTS`
  - Implement CA signed-hostkeys

17.5.2 Python Module not installed

Error: While executing Ansible you see: `No module named '<MODULE>'`

Problem:

- Your Ansible controller system is missing a required Python3 module!
- If you are NOT using Docker, you can install it manually using PIP: `python3 -m pip install <MODULE>`

  You could also find and install the module using your systems package manager: `sudo apt install python3-<MODULE>` (NOTE: these packages are older versions)
- If you are using Docker, you can create and mount a `requirements.txt` and restart your container. See also: *Usage - Docker*

17.5.3 CSRF Failed

Error: After submitting a form you see: `Forbidden (403) CSRF verification failed. Request aborted.`

Problem:

- The hostname you are using to access AW is probably not configured as/listed in `AW_HOSTNAMES`

17.5.4 SSH Shared connection

Error: While executing Ansible you see: `Shared connection to <IP> closed`

Problem:

- This seems to be an issue of how Ansible calls SSH. Have seen it happen on a few systems - even with using vanilla Ansible via CLI.

  The issue is that a `mux` process has not terminated gracefully.

  Search for the process: `ps -aux | grep mux` and kill it *kill -9 <PID>* (*the PID is the number in the second column*)

17.5.5 SAML Issues

To get more information - you can enable its logging by adding this block to the config file:

```
...

SAML:
    LOGGING:
        version: 1
        formatters:
            simple:
              format: '[%(asctime)s] [%(levelname)s] [%(name)s.%(funcName)s] %(message)s'
        handlers:
            stdout:
                class: 'logging.StreamHandler'
                stream: 'ext://sys.stdout'
                level: 'DEBUG'
                formatter: 'simple'
        loggers:
            saml2:
                level: 'DEBUG'
        root:
            level: 'DEBUG'
            handlers: ['stdout']
```

Note: The SAML config-file is only reloaded on restart.

Common errors you might encounter:

- `CSRF validation failed` - the ACS url may not be configured correctly

- If you see a page with an error-code - you can look-up their references here

  Per example:

  - `1107` means you supplied an invalid SAML configuration or the `xmlsec` package is not installed
  - `1110` means you might need to check your IDPs metadata and modify the `NAME_ID_FORMAT` setting
  - `1113` and `1114` mean you have not or mis-configured your attribute mappings

Note: SAML testing has been done using the mocksaml.com service

17.6 Edge-Case Issues

17.6.1 Connection in use

Error: While starting AW you see: `Connection in use: ('127.0.0.1', 8000)`

Problem

- Make sure no other process is binding to port 8000: `netstat -tulpn | grep 8000`

  If that is the case - you can set the `AW_PORT` env-var to change the port to be used.

- The app failed last time. There is still an old process running. If this happens repeatedly - open an issue!

  You can find and kill it:

```
# find it
pgrep -f ansibleguy-webui
netstat -tulpn | grep 8000
ps -aux | grep ansibleguy-webui | grep -v grep

# kill it
pkill -f ansibleguy-webui
kill -9 <PID>
```

17.6.2 Database is locked

Error: The Web interface shows a plain `Error 500` and the console shows `django.db.utils.OperationalError: database is locked`

Problem:

- I've encountered this issue a few times. It occurs because the SQLite database is locked by a write-operation.

  Restarting the application is the easiest way of working around it.

  If it occurs more often - please open an issue!

- If you are running many jobs - you could try to keep a minute between their scheduled executions.

17.6.3 Too Many Log Files exist

Error: Job logs are currently not cleaned automatically. You may want to clean them manually periodically.

Resolution:

- You can easily remove all log-files older than N days with this command:

```
MAX_LOG_AGE=7  # days
cd ~/.local/share/ansible-webui/
find -type f -mtime +${MAX_LOG_AGE} -name "*.log" -delete
```

17.6.4 Database Migration Issues

Note: This is a general guide on how to handle Django migration issues. It could also be helpful if you are running another Django app.

Error: After a version upgrade you see `django.db.utils.OperationalError: no such column` or even `django.db.utils.OperationalError: no such table`

Problem:

- It seems the database schema was not upgraded. This is normally done automatically at application startup.
- You can try to execute the migrations manually:
  - Stop the application
  - Enter the application context & try to upgrade

```
# when running as local service-user
su <SERVICE-USER> --login --shell /bin/bash

# when running in docker
docker exec -it ansible-webui /bin/sh

# set the path to your database
export AW_DB=<PATH-TO-YOUR-DB>

# upgrade DB schema
python3 -m ansibleguy-webui.manage migrate
```

Error: While running the database schema upgrade you see `django.db.utils.OperationalError: duplicate column name` or `django.db.utils.OperationalError: duplicate table name`

Problem:

- This should never happen if you are running a release version (*AW_ENV=prod*) and did not already run migrations manually.
- Make sure you set the `AW_DB` env-var correctly before running the migrations.
- You will have to find out which migrations were already applied:

  `python3 -m ansibleguy-webui.manage showmigrations`

  - Or check your database manually:

```
sqlite3 <PATH-TO-YOUR-DB>
SELECT name,applied FROM django_migrations WHERE app = "aw";
```

  - You can also check the current schema of the table you see mentioned in the error message

```
sqlite3 <PATH-TO-YOUR-DB>
.table
.schema <TABLE>
```

- Check which migrations are available: `python3 -m ansibleguy-webui.cli -a migrations.list`
- With that information you should be able to determine which migrations you can `fake` and which ones to apply.

```
# migrations that are available and already are applied to the database -␣
↪can be faked (only last one)
python3 -m ansibleguy-webui.manage migrate --fake aw 0001_v0_0_12

# you should then be able to apply the un-applied migrations
python3 -m ansibleguy-webui.manage migrate aw 0002_v0_0_13
```

17.6.5 Database Startup Issue

Error: While starting AW - you see the error `sqlite3.DatabaseError: database disk image is malformed`

Problem:

- The service may have been force-terminated without being able to close the database connection gracefully.

  You can try to re-/move the `aw.db-shm` and `aw.db-wal` files that can be found in the same directory as your database-file.