Interactive input: prompts

- Hashing values supplied by vars_prompt
- Allowing special characters in vars_prompt values

If you want your playbook to prompt the user for certain input, add a 'vars_prompt' section. Prompting the user for variables lets you avoid recording sensitive data like passwords. In addition to security, prompts support flexibility. For example, if you use one playbook across multiple software releases, you could prompt for the particular release version.

Here is a most basic example:

```
---
- hosts: all
  vars_prompt:

    - name: username
      prompt: What is your username?
      private: false

    - name: password
      prompt: What is your password?

  tasks:

    - name: Print a message
      ansible.builtin.debug:
        msg: 'Logging in as {{ username }}'
```

The user input is hidden by default but it can be made visible by setting `private: false`. Starting with `ansible-core` 2.22, a value entered for a private prompt is also registered as a secret and masked in Ansible output, see `playbooks_secret_masking`.

> Prompts for individual `vars_prompt` variables will be skipped for any variable that is already defined through the command line `--extra-vars` option, or when running from a non-

interactive session (such as cron or Ansible AWX). See `passing_variables_on_the_command_line`.

If you have a variable that changes infrequently, you can provide a default value that can be overridden.

```
vars_prompt:

  - name: release_version
    prompt: Product release version
    default: "1.0"
```

Hashing values supplied by `vars_prompt`

You can hash the entered value so you can use it, for example, with the user module to define a password:

```
vars_prompt:

  - name: my_password2
    prompt: Enter password2
    private: true
    encrypt: sha512_crypt
    confirm: true
    salt_size: 7
```

If you have Passlib installed, you can use any crypt scheme the library supports:

- *des_crypt* - DES Crypt
- *bsdi_crypt* - BSDi Crypt
- *bigcrypt* - BigCrypt
- *crypt16* - Crypt16
- *md5_crypt* - MD5 Crypt
- *bcrypt* - BCrypt
- *sha1_crypt* - SHA-1 Crypt
- *sun_md5_crypt* - Sun MD5 Crypt
- *sha256_crypt* - SHA-256 Crypt
- *sha512_crypt* - SHA-512 Crypt

- *apr_md5_crypt* - Apache's MD5-Crypt variant
- *phpass* - PHPass' Portable Hash
- *pbkdf2_digest* - Generic PBKDF2 Hashes
- *cta_pbkdf2_sha1* - Cryptacular's PBKDF2 hash
- *dlitz_pbkdf2_sha1* - Dwayne Litzenberger's PBKDF2 hash
- *scram* - SCRAM Hash
- *bsd_nthash* - FreeBSD's MCF-compatible nthash encoding

The only parameters accepted are 'salt' or 'salt_size'. You can use your own salt by defining 'salt', or have one generated automatically using 'salt_size'. By default, Ansible generates a salt of size 8.

2.7

If you do not have Passlib installed, Ansible uses the crypt library as a fallback. Ansible supports at most four crypt schemes, depending on your platform at most the following crypt schemes are supported:

- *bcrypt* - BCrypt
- *md5_crypt* - MD5 Crypt
- *sha256_crypt* - SHA-256 Crypt
- *sha512_crypt* - SHA-512 Crypt

2.8

Allowing special characters in `vars_prompt` values

Some special characters, such as `{` and `%` can create templating errors. If you need to accept special characters, use the `unsafe` option:

```
vars_prompt:
  - name: my_password_with_weird_chars
    prompt: Enter password
    unsafe: true
    private: true
```

`playbooks_intro`
An introduction to playbooks

`playbooks_conditionals`
Conditional statements in playbooks

`playbooks_variables`
All about variables

`Communication<communication>`
Got questions? Need help? Want to share your ideas? Visit the Ansible communication guide

---

□□□ #2
□ □□□ □□□ 2026-10-31 14:40:46 UTC
□ □□□ □□□ 2026-10-31 14:40:47 UTC