Ansible

Preview

Red Hat | 131,213 installs | ★★★★☆ (19) | Free

Ansible language support

Install Trouble Installing?

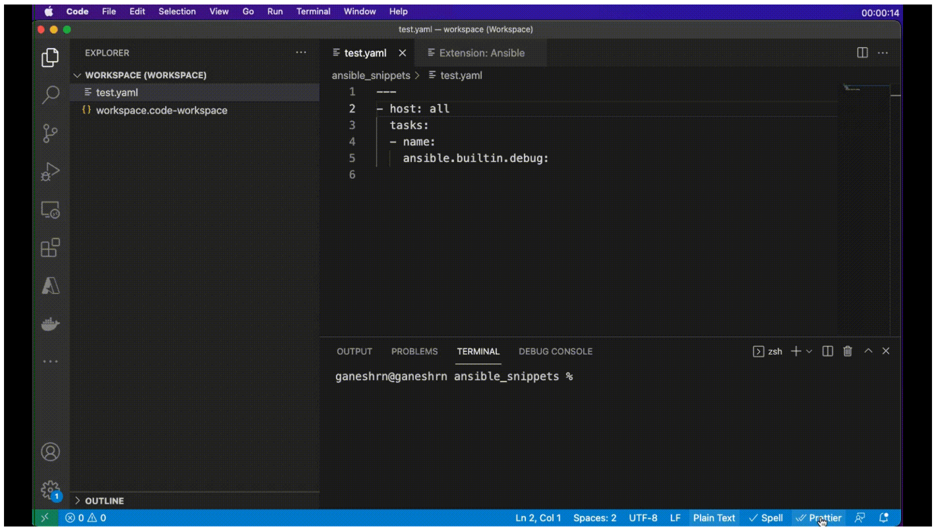
Overview Version History Rating & Review

Ansible VS Code Extension by Red Hat

This extension adds language support for Ansible to Visual Studio Code and OpenVSX compatible editors by leveraging ansible-language-server.

Activating Red Hat Ansible extension

It is recommended to open a folder containing Ansible files with a VS Code workspace.


Code File Edit Selection View Go Run Terminal Window Help
00:00:14
test.yaml — workspace (Workspace)
EXPLORER
WORKSPACE (WORKSPACE)
test.yaml
workspace.code-workspace
test.yaml
Extension: Ansible
ansible_snippets > test.yaml
OUTPUT PROBLEMS TERMINAL DEBUG CONSOLE
zsh
OUTLINE
Ln 2, Col 1 Spaces: 2 UTF-8 LF Plain Text Spell Prettier

```
---
- host: all
  tasks:
  - name:
    ansible.builtin.debug:
```

```
ganeshrn@ganeshrn ansible_snippets %
```

Note:

- For Ansible files open in an editor window ensure the language mode is set to `Ansible` (bottom right of VS Code window).
- The runtime status of extension should be in activate state. It can be verified in the `Extension` window `Runtime Status` tab for `Ansible` extension.

Features

Syntax highlighting

Categories

Programming Languages Linters

Tags

ansible ansible-jinja autocompletion json validation yaml

Works with

Universal

Resources

Issues
Repository
Homepage
License
Changelog
Download Extension

Project Details

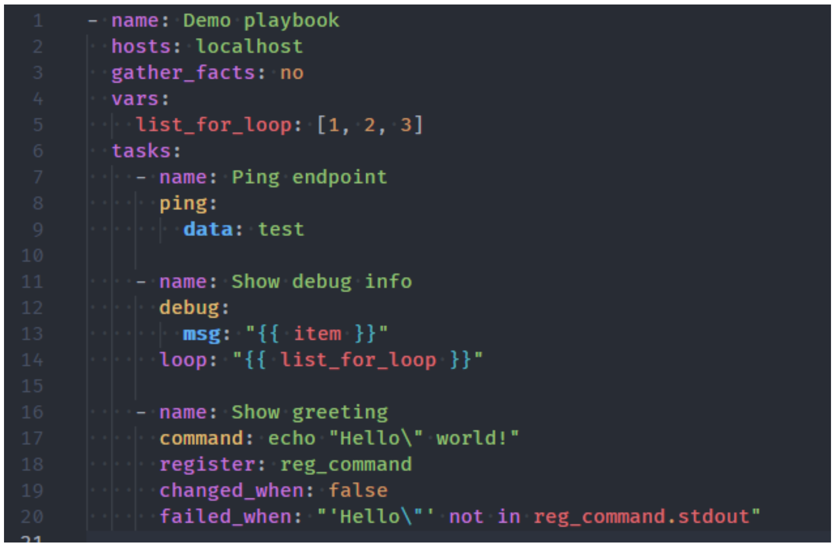
ansible/vscode-ansible
Last Commit: a week ago
4 Pull Requests
40 Open Issues

Visual Studio Marketplace v0.10.0

build failing

More Info

Version	0.10.0
Released on	8/24/2021, 4:01:54 AM
Last updated	5/24/2022, 12:23:00 PM
Publisher	Red Hat
Unique Identifier	redhat.ansible
Report	Report Abuse

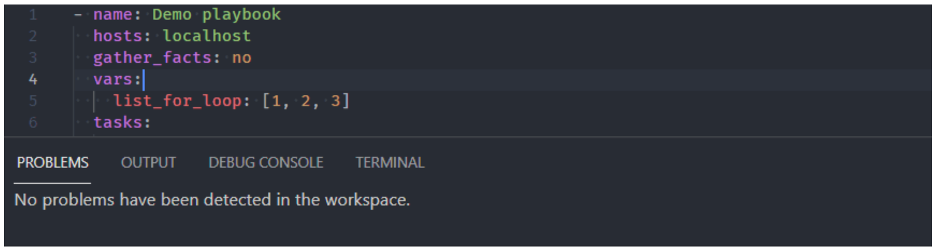


```yaml
- name: Demo playbook
  hosts: localhost
  gather_facts: no
  vars:
    list_for_loop: [1, 2, 3]
  tasks:
    - name: Ping endpoint
      ping:
        data: test

    - name: Show debug info
      debug:
        msg: "{{ item }}"
      loop: "{{ list_for_loop }}"

    - name: Show greeting
      command: echo "Hello\" world!"
      register: reg_command
      changed_when: false
      failed_when: "'Hello\"' not in reg_command.stdout"
```

Ansible keywords, **module names** and **module options**, as well as standard YAML elements are recognized and highlighted distinctly. Jinja expressions are supported too, also those in Ansible conditionals (`when`, `failed_when`, `changed_when`, `check_mode`), which are not placed in double curly braces.

> The screenshots and animations presented in this README have been taken using the One Dark Pro theme. The default VS Code theme will not show the syntax elements as distinctly, unless customized. Virtually any theme other than default will do better.

Validation

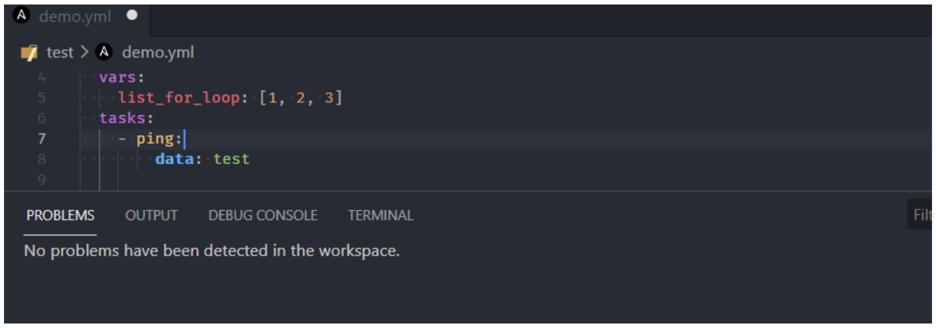


```yaml
- name: Demo playbook
  hosts: localhost
  gather_facts: no
  vars:
    list_for_loop: [1, 2, 3]
  tasks:
```

No problems have been detected in the workspace.

While you type, the syntax of your Ansible scripts is verified and any feedback is provided instantaneously.

Integration with ansible-lint



```yaml
  vars:
    list_for_loop: [1, 2, 3]
  tasks:
    - ping:
        data: test
```

No problems have been detected in the workspace.

On opening and saving a document, `ansible-lint` is executed in the background and any findings are presented as errors. You might find it useful that rules/tags added to `warn_list` (see Ansible Lint Documentation) are shown as warnings instead.

Smart autocompletion

```
  -
```

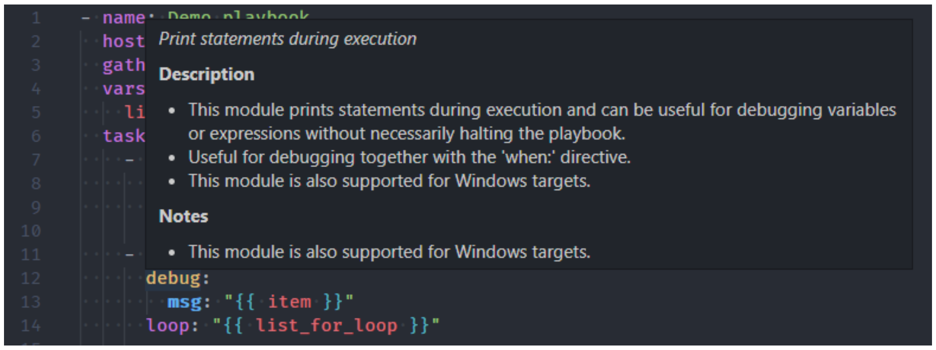
The extension tries to detect whether the cursor is on a play, block or task etc. and provides suggestions accordingly. There are also a few other rules that improve user experience:

- the `name` property is always suggested first
- on module options, the required properties are shown first, and aliases are shown last, otherwise ordering from the documentation is preserved
- FQCNs (fully qualified collection names) are inserted only when necessary; collections configured with the `collections` keyword are honored. This behavior can be disabled in extension settings.

Auto-closing Jinja expressions

When writing a Jinja expression, you only need to type `"{{`, and it will be mirrored behind the cursor (including the space). You can also select the whole expression and press `space` to put spaces on both sides of the expression.

Documentation reference



```
- name: Demo playbook
  host
  gath
  vars
    li
  task
    -

    -
      debug:
        msg: "{{ item }}"
      loop: "{{ list_for_loop }}"
```

Print statements during execution

Description

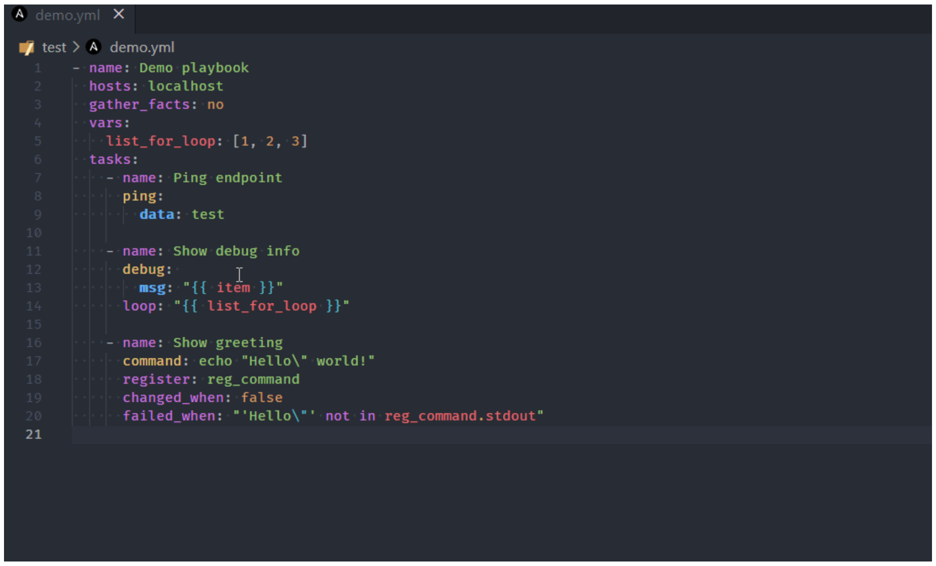
- This module prints statements during execution and can be useful for debugging variables or expressions without necessarily halting the playbook.
- Useful for debugging together with the 'when:' directive.
- This module is also supported for Windows targets.

Notes

- This module is also supported for Windows targets.

Documentation is available on hover for Ansible keywords, modules and module options. The extension works on the same principle as `ansible-doc`, providing the documentation straight from the Python implementation of the modules.

Jump to module code



```
- name: Demo playbook
  hosts: localhost
  gather_facts: no
  vars:
    list_for_loop: [1, 2, 3]
  tasks:
    - name: Ping endpoint
      ping:
        data: test

    - name: Show debug info
      debug:
        msg: "{{ item }}"
      loop: "{{ list_for_loop }}"

    - name: Show greeting
      command: echo "Hello\" world!"
      register: reg_command
      changed_when: false
      failed_when: "'Hello\"' not in reg_command.stdout"
```

You may also open the implementation of any module using the standard *Go to Definition* operation, for instance, by clicking on the module name while holding `ctrl`/`cmd`.

Requirements

- Ansible 2.9+
- Ansible Lint (required, unless you disable linter support; install without `yamllint`)

For Windows users, this extension works perfectly well with extensions such as `Remote - WSL` and `Remote - Containers`.

> If you have any other extension providing language support for Ansible, you might need to uninstall it first.

Configuration

This extension supports multi-root workspaces, and as such, can be configured on any level (User, Remote, Workspace and/or Folder).

- `ansible.ansible.path`: Path to the `ansible` executable.
- `ansible.ansible.useFullyQualifiedCollectionNames`: Toggles use of fully qualified collection names (FQCN) when inserting a module name. Disabling it will only use FQCNs when necessary, that is when the collection isn't configured for the task.
- `ansible.ansibleLint.arguments`: Optional command line arguments to be appended to `ansible-lint` invocation. See `ansible-lint` documentation.
- `ansible.ansibleLint.enabled`: Enables/disables use of `ansible-lint`.
- `ansible.ansibleLint.path`: Path to the `ansible-lint` executable.
- `ansible.ansibleNavigator.path`: Path to the `ansible-navigator` executable.
- `ansible.executionEnvironment.containerEngine`: The container engine to be used while running with execution environment. Valid values are `auto`, `podman` and `docker`. For `auto` it will look for `podman` then `docker`.
- `ansible.executionEnvironment.containerOptions`: Extra parameters passed to the container engine command example: `--net=host`
- `ansible.executionEnvironment.enabled`: Enable or disable the use of an execution environment.
- `ansible.executionEnvironment.image`: Specify the name of the execution environment image.
- `ansible.executionEnvironment.pull.arguments`: Specify any additional parameters that should be added to the pull command when pulling an execution environment from a container registry. e.g. `--tls-verify=false`
- `ansible.executionEnvironment.pull.policy`: Specify the image pull policy. Valid values are `always`, `missing`, `never` and `tag`. Setting `always` will always pull the image when extension is activated or reloaded. Setting `missing` will pull if not locally available. Setting `never` will never pull the image and setting tag will always pull if the image tag is 'latest', otherwise pull if not locally available.
- `ansible.executionEnvironment.volumeMounts`: The setting contains volume mount information for each entry in the list. Individual entry consist of a
  - `src`: The name of the local volume or path to be mounted within execution environment.
  - `dest`: The path where the file or directory are mounted in the container.
  - `options`: The field is optional, and is a comma-separated list of options, such as ro,Z
- `ansible.python.interpreterPath`: Path to the `python/python3` executable. This setting may be used to make the extension work with `ansible` and `ansible-lint` installations in a Python virtual environment.
- `ansible.python.activationScript`: Path to a custom `activate` script, which will be used instead of the setting above to run in a Python virtual environment.
- `ansibleServer.trace.server`: Traces the communication between VSCode and the ansible language server.

Known limitations

- The shorthand syntax for module options (key=value pairs) is not supported.
- Nested module options are not supported yet.
- Only Jinja *expressions* inside Ansible YAML files are supported. In order to have syntax highlighting of Jinja template files, you'll need to install other extension.
- Jinja *blocks* (inside Ansible YAML files) are not supported yet.

Credit

Based on the good work done by Tomasz Maciążek