The Pitfalls of Ansible's Variables and Template Expressions

Ruben Opdebeeck *ropdebee@vub.be*

Coen De Roover *cderoove@vub.be*

Ansible: An Introduction

Configuration management

Multi-machine deployments

Cloud provisioning

Tool	#	Usage %
Docker	26	59.0%
Ansible	23	52.2%
Vagrant	19	43.1%
Kubernetes	18	40.9%
Chef	16	36.3%
Terraform	15	34.1%

Guerriero et al., 2019

Unconventional Semantics

```
---
- hosts: localhost
  tasks:
    - include_vars: my_vars.yml
    - debug:
        msg: '{{ a + 1 }}'
      vars:
        a: 1
```

```
---
# my_vars.yml
a: 2
```

Unconventional Semantics

```yaml
---
- hosts: localhost
  tasks:
    - include_vars: my_vars.yml
    - debug:
        msg: '{{ a + 1 }}'
      vars:
        a: 1
```

Playbook

```yaml
---
# my_vars.yml
a: 2
```

Tasks

Unconventional Semantics

```
---
- hosts: localhost
  tasks:
    - include_vars: my_vars.yml
    - debug:
        msg: '{{ a + 1 }}'
      vars:
        a: 1
```

```
---
# my_vars.yml
a: 2
```

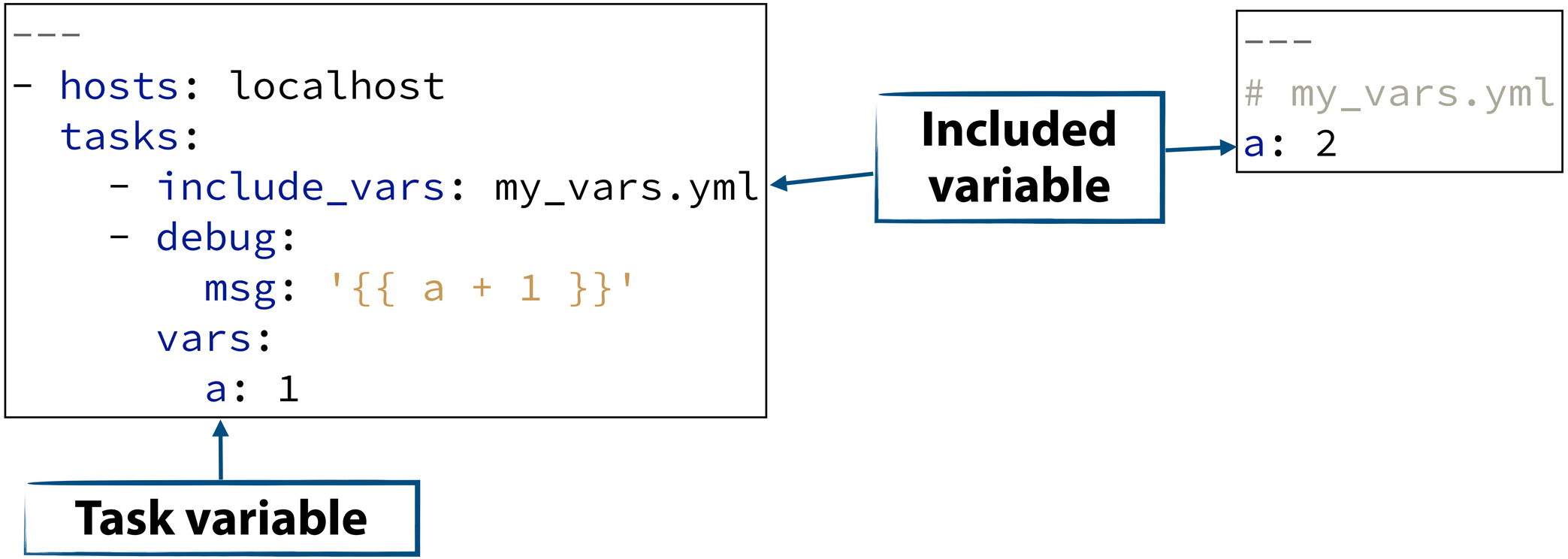
Unconventional Semantics

```yaml
---
- hosts: localhost
  tasks:
    - include_vars: my_vars.yml
    - debug:
        msg: '{{ a + 1 }}'
      vars:
        a: 1
```

Task variable

```yaml
---
# my_vars.yml
a: 2
```

Unconventional Semantics



Unconventional Semantics

```yaml
---
- hosts: localhost
  tasks:
    - include_vars: my_vars.yml
    - debug:
        msg: '{{ a + 1 }}'
      vars:
        a: 1
```

Task variable

Included variable

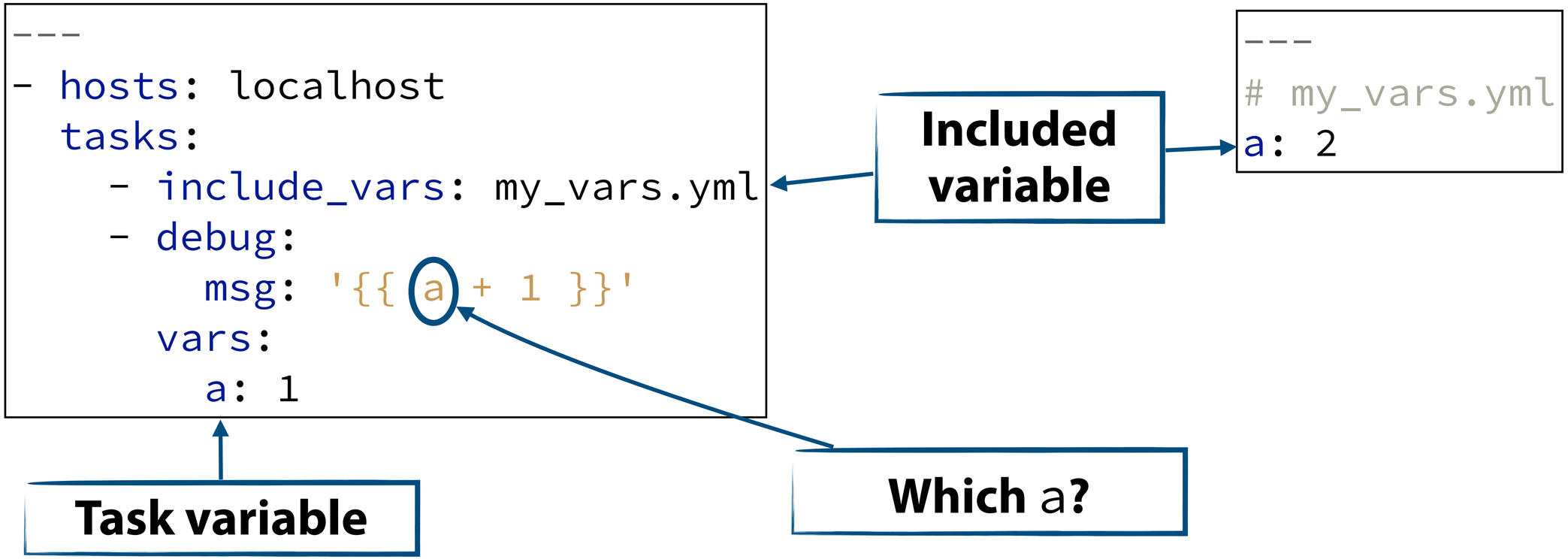
```yaml
---
# my_vars.yml
a: 2
```

Q: What'll be printed?

A 2

B 3

Unconventional Semantics



Q: What'll be printed?

A 2

B 3

Variable Precedence

1. command line values (for example, `-u my_user`, these are not variables)
2. role defaults (defined in role/defaults/main.yml) [1]
3. inventory file or script group vars [2]
4. inventory group_vars/all [3]
5. playbook group_vars/all [3]
6. inventory group_vars/* [3]
7. playbook group_vars/* [3]
8. inventory file or script host vars [2]
9. inventory host_vars/* [3]
10. playbook host_vars/* [3]
11. host facts / cached set_facts [4]
12. play vars
13. play vars_prompt
14. play vars_files
15. role vars (defined in role/vars/main.yml)
16. block vars (only for tasks in block)
17. task vars (only for the task)
18. include_vars
19. set_facts / registered vars
20. role (and include_role) params
21. include params
22. extra vars (for example, `-e "user=my_user"`)(always win precedence)

Lower

Higher

17. Task vars

18. `include_vars`

22 (!) precedence rules

Unconventional Semantics

```
---
- hosts: localhost
  tasks:
    - include_vars: my_vars.yml
    - debug:
        msg: '{{ a + 1 }}'
      vars:
        a: 1
```

```
---
# my_vars.yml
a: 2
```

Q: What'll be printed?

A 2

B 3

Unconventional Semantics

```yaml
---
- hosts: localhost
  tasks:
    - include_vars: my_vars.yml
    - debug:
        msg: '{{ a + 1 }}'
      vars:
        a: 1
```

```yaml
---
# my_vars.yml
a: 2
```

Q: What'll be printed?

A 2

B 3

Unconventional Semantics

```yaml
---
- hosts: localhost
  tasks:
    - include_vars: my_vars.yml
    - debug:
        msg: '{{ a + 1 }}'
      vars:
        a: 1
```

```yaml
---
# my_vars.yml
a: 2
```

Q: What'll be printed?

A 2

B 3 ✓

In Summary

1. *Complicated variable precedence*
2. *Expressions lazily evaluated*
3. *Expressions don't close over scopes*

1. *Easy to introduce bugs*
2. *Difficult to debug bugs*

Not convinced? Check out the full presentation!

In Summary

1. *Complicated variable precedence*
2. *Expressions lazily evaluated*
3. *Expressions don't close over scopes*

⇒

1. *Easy to introduce bugs*
2. *Difficult to debug bugs*

Not convinced? Check out the full presentation!

Call to action:

1. Spreading awareness
2. Tool support: Static analyses, bug detectors, debuggers, …
3. Study impact on safety and reliability of infrastructure