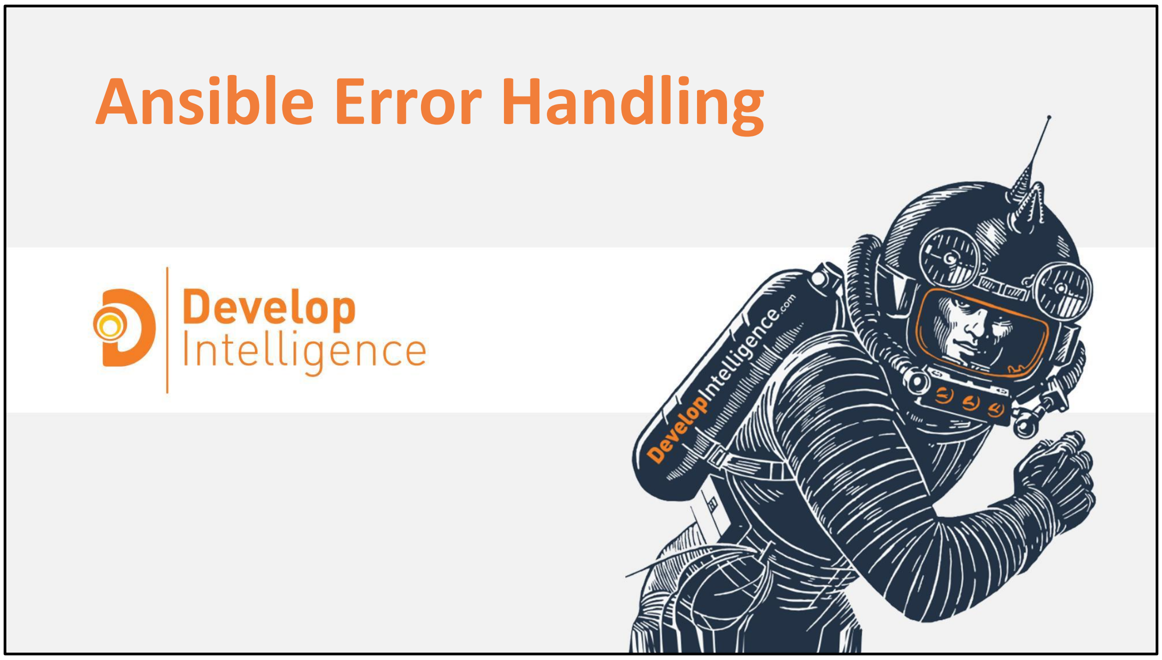
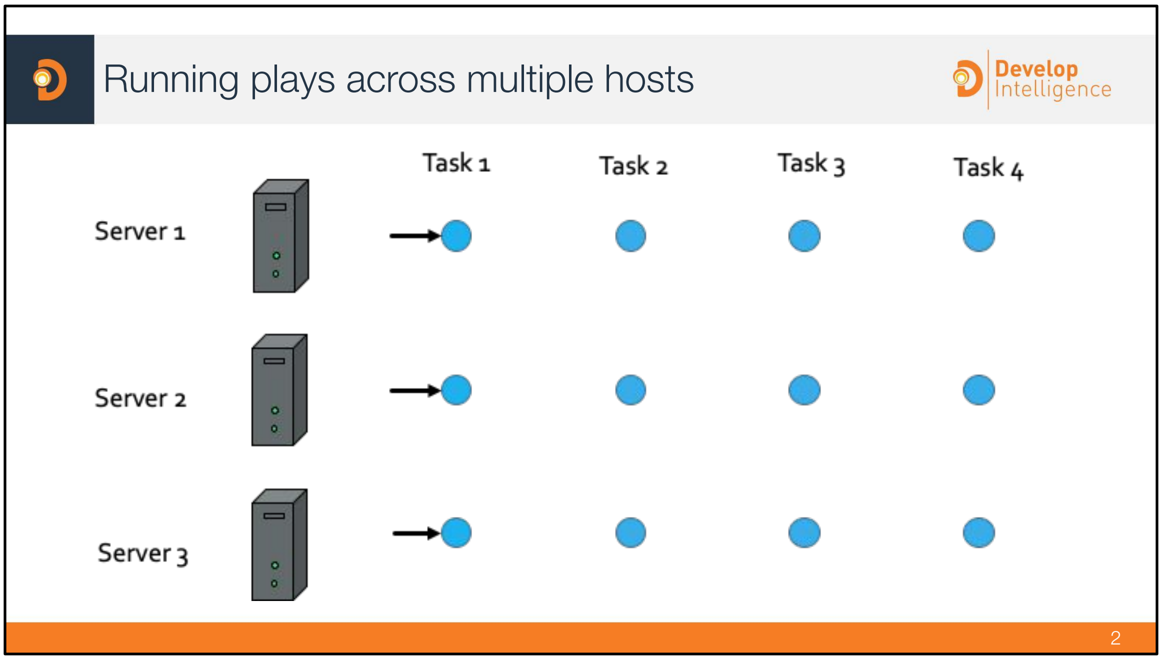


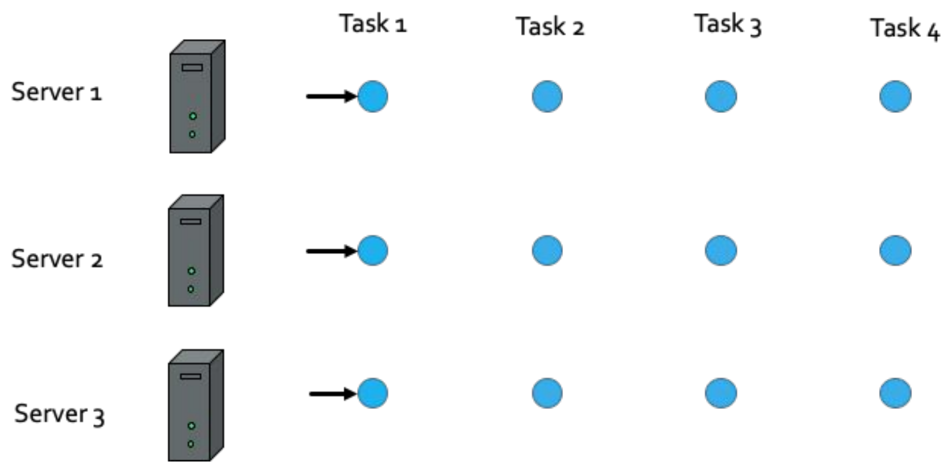
Ansible Error Handling

Develop Intelligence



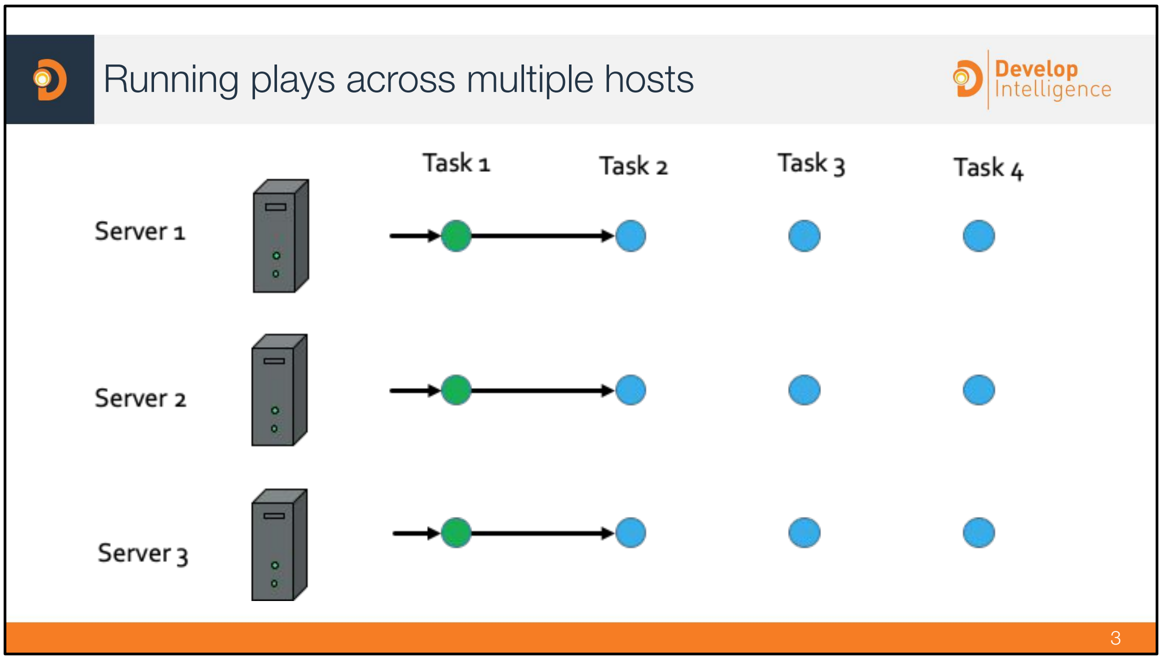
Running plays across multiple hosts

Develop Intelligence



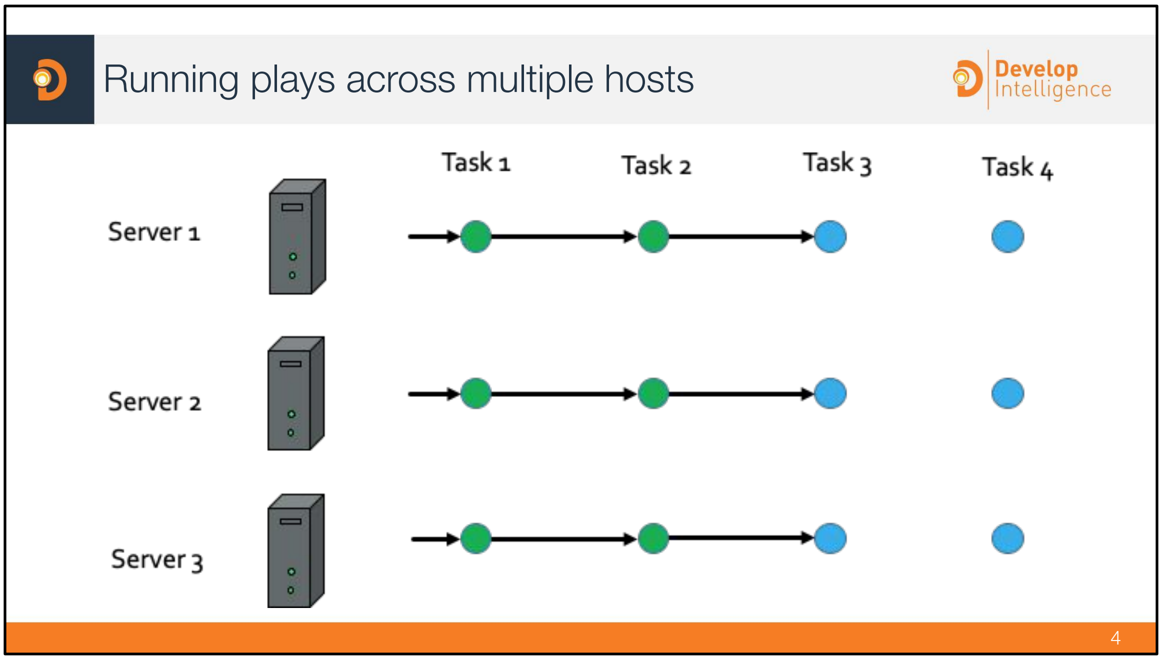
2

A play running tasks across multiple servers will perform one task at a time across all servers. When that task completes on all server, the next task will begin.



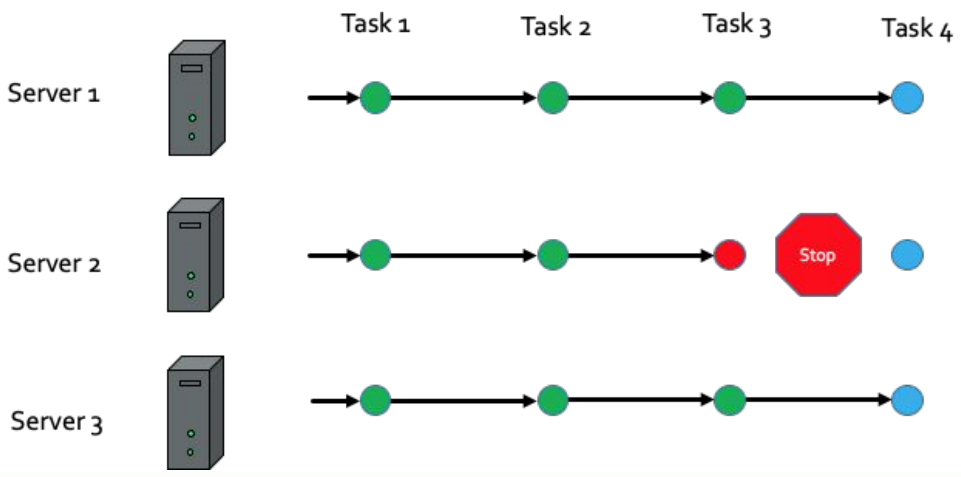
Running plays across multiple hosts

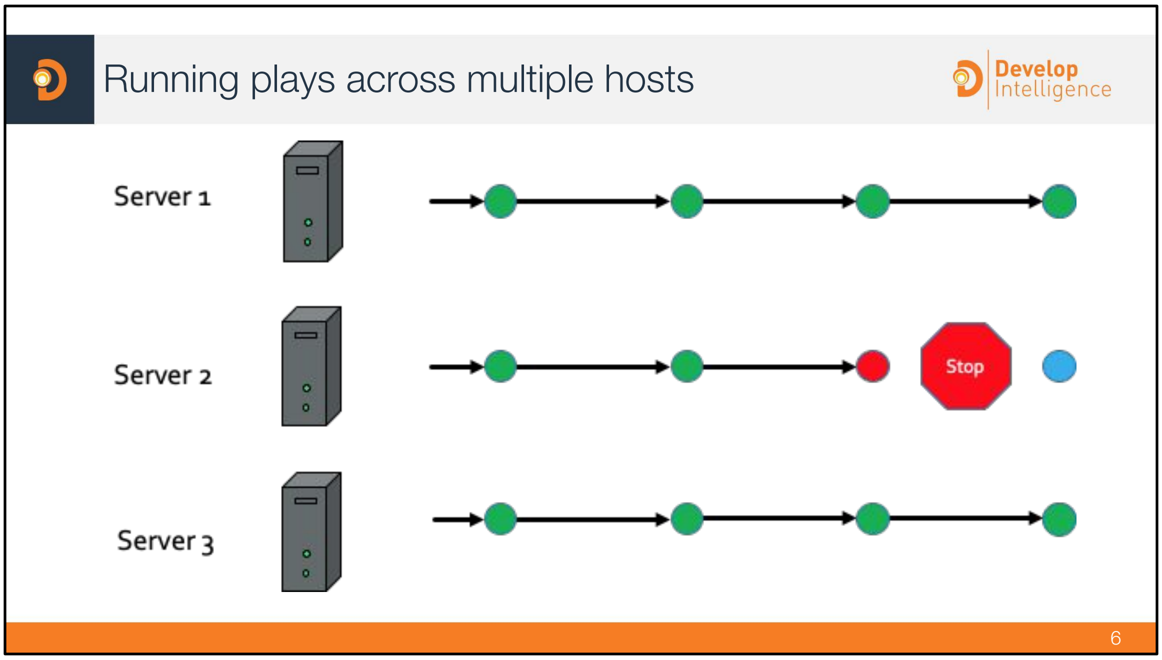
3



Running plays across multiple hosts

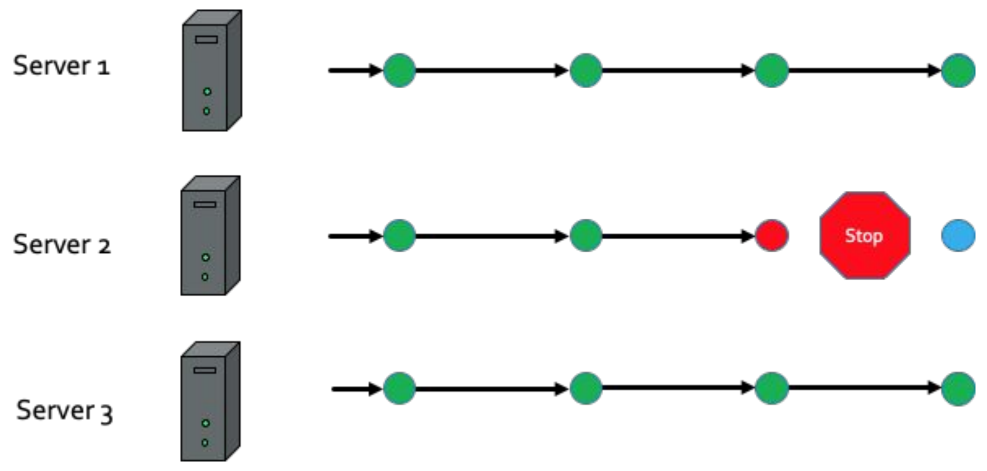
Running plays across multiple hosts





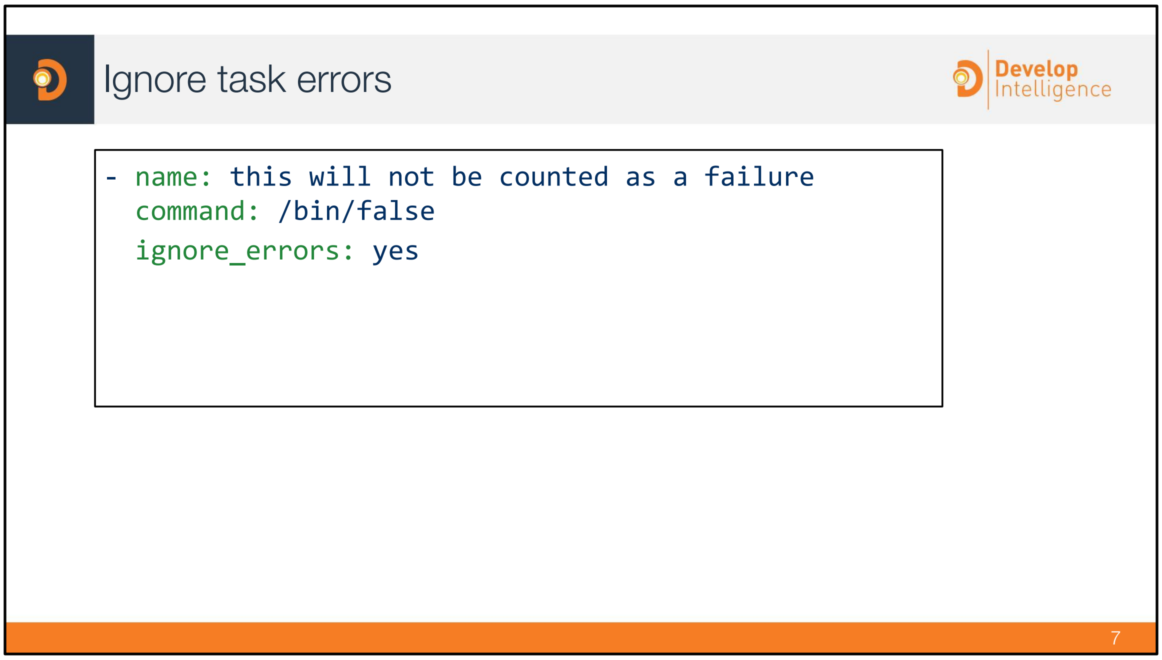
Running plays across multiple hosts

Develop Intelligence



6

If a task fails, that host fails (by default) and tasks stop execution. However, ansible will attempt to complete as many hosts as it can



Ignore task errors

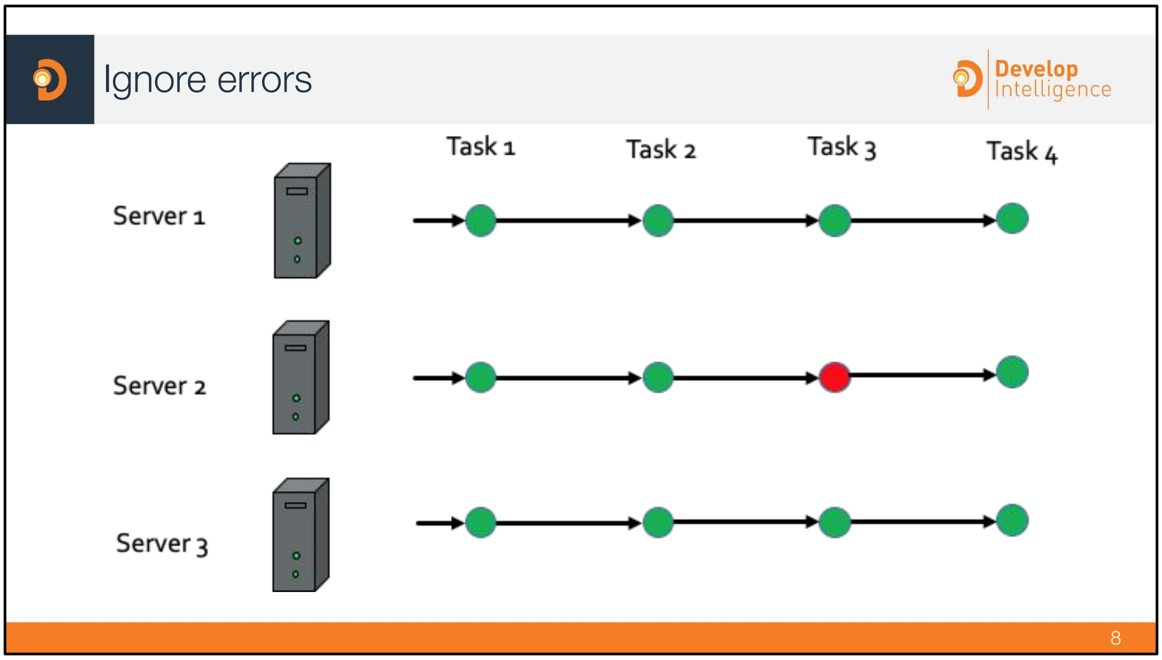
Develop Intelligence

```
- name: this will not be counted as a failure
  command: /bin/false
  ignore_errors: yes
```

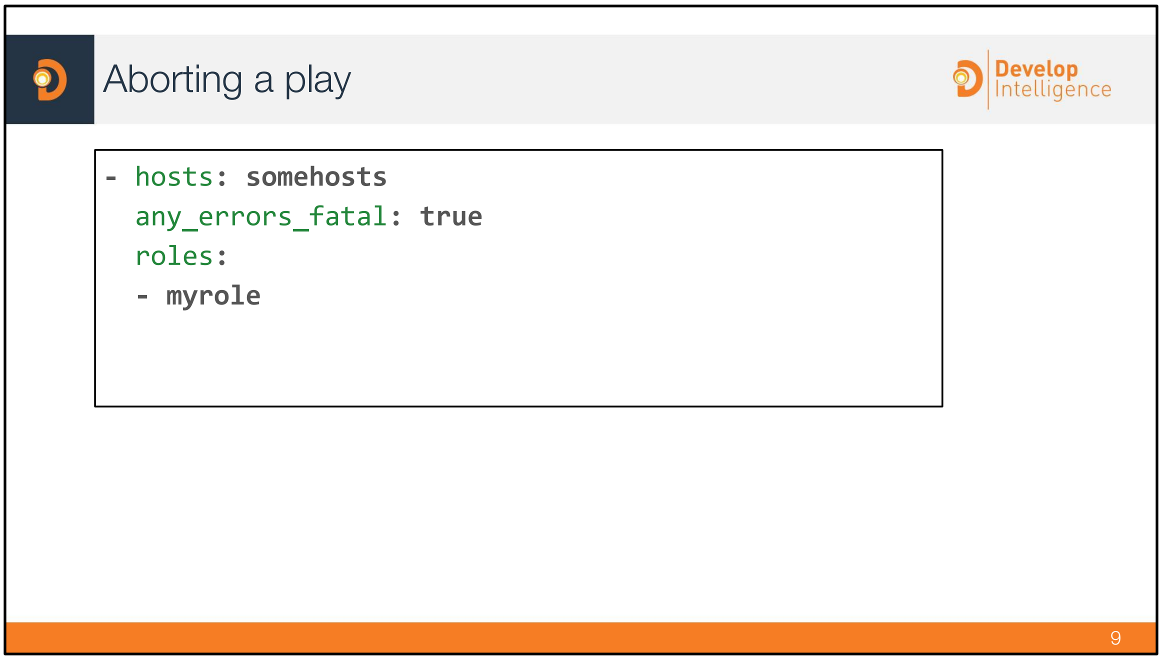
7

If a task fails, a playbook will stop executing immediately. Using the ignore_errors parameter will allow the play to continue even if the task fails.

Examples of use:
* When you want to perform a more complex error check in a later task (https://unix.stackexchange.com/questions/355573/how-to-ignore-a-particular-output-error-string-in-ansible-and-consider-successfu/355584)
* Useful while developing playbooks



Ignore errors



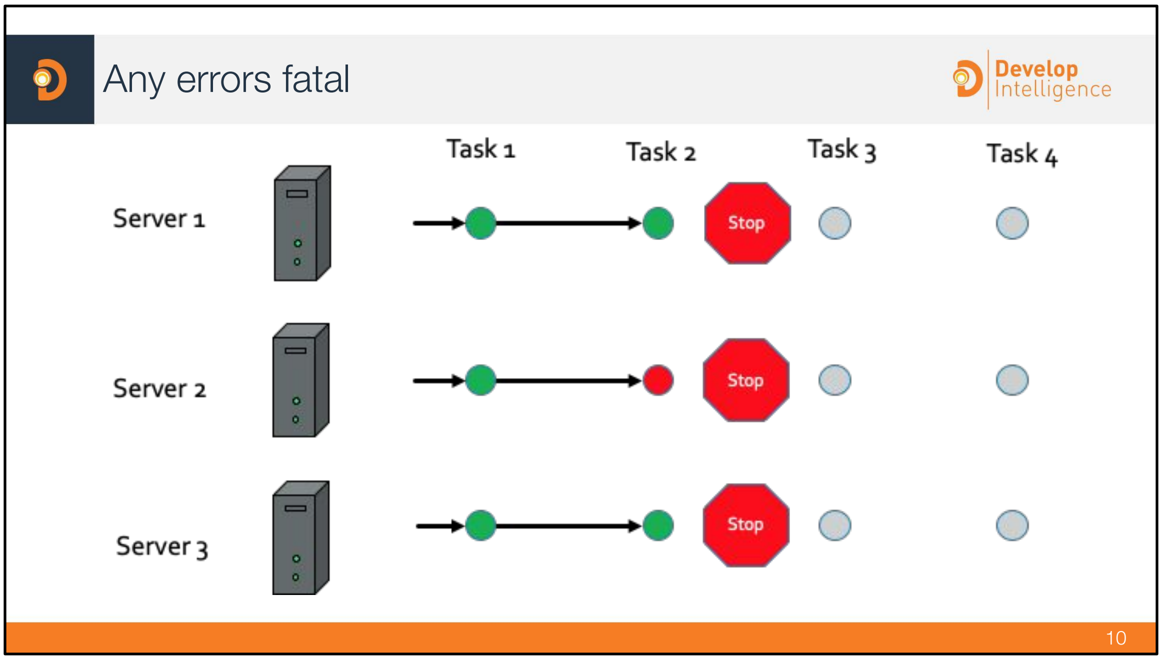
Aborting a play

```
- hosts: somehosts
  any_errors_fatal: true
  roles:
  - myrole
```

9

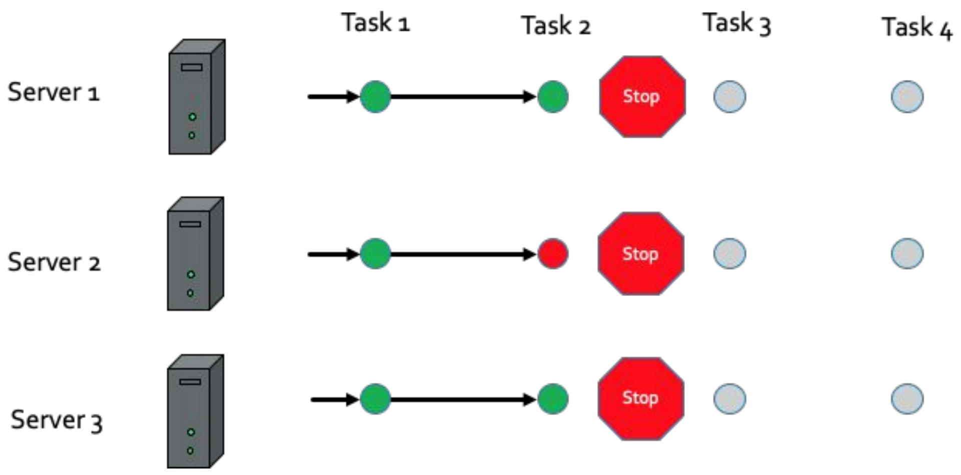
If you are executing a play across multiple hosts, any_errors_fatal will mark all hosts as failed if any host fails and immediately abort. max_fail_percentage can also be used if you wish to abort all hosts after a certain number of hosts.

Examples of use:
The play result is not useful if all hosts do not finish. This will save time by not waiting for other hosts to finish

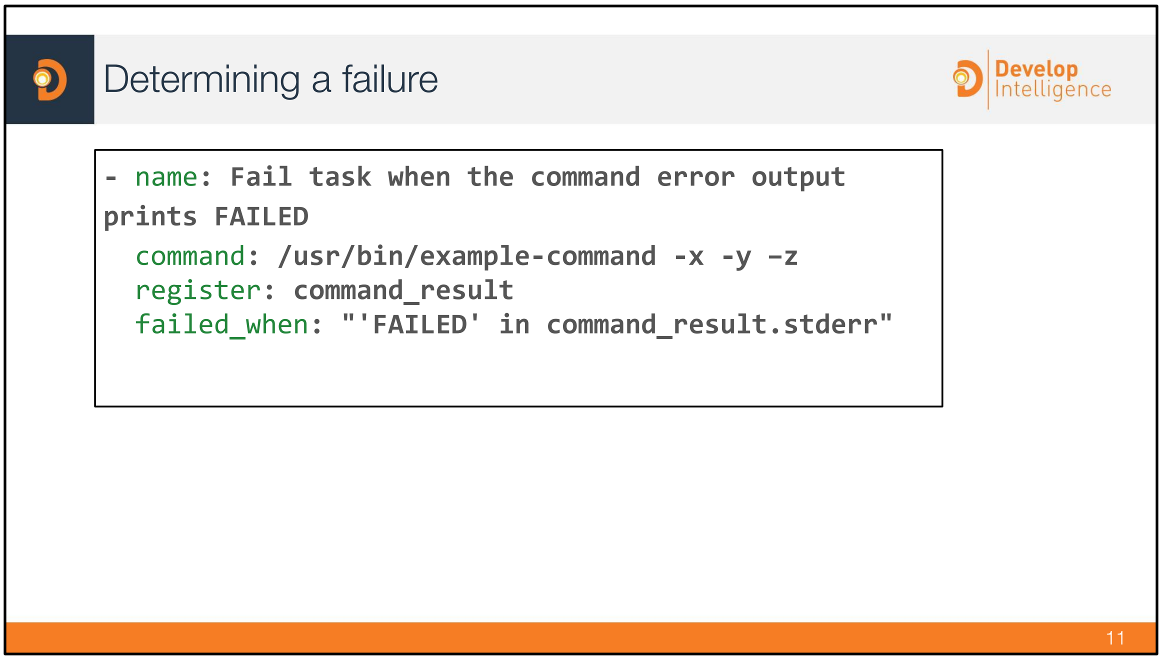


Any errors fatal

Develop Intelligence



10



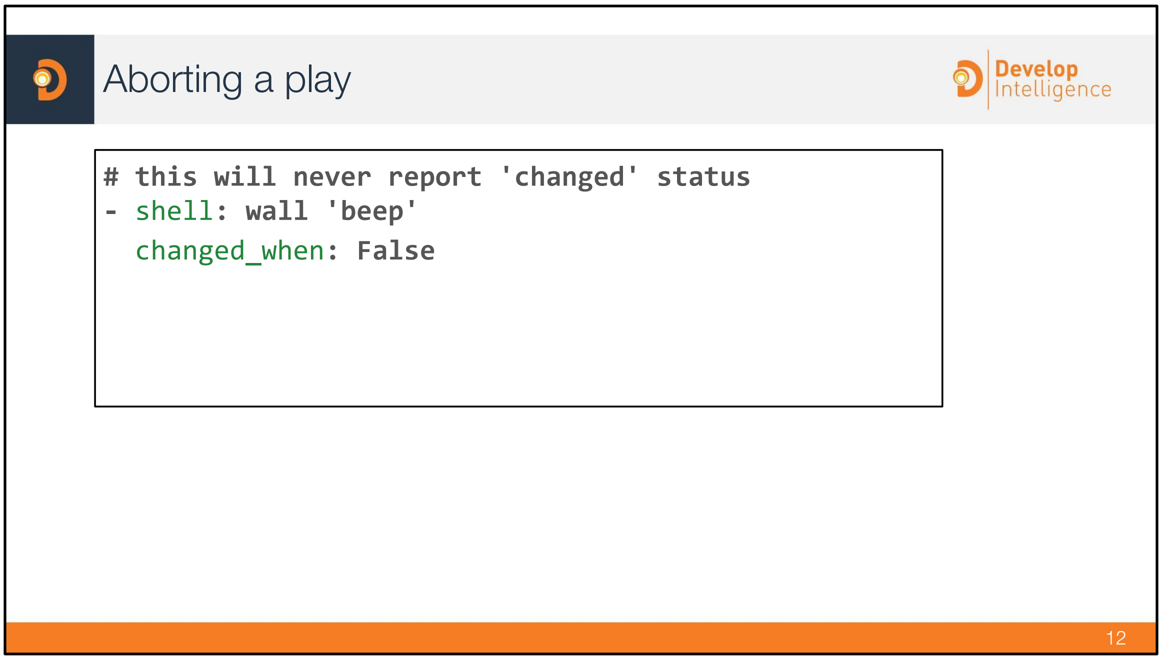
Determining a failure

```
- name: Fail task when the command error output
prints FAILED
  command: /usr/bin/example-command -x -y -z
  register: command_result
  failed_when: "'FAILED' in command_result.stderr"
```

11

Sometimes there may be special cases when a task should fail.

Examples of use:
* a task that performs an HTTP request and wants to ensure the response code is ‘404’, failing when it is anything else.



Aborting a play

```
# this will never report 'changed' status
- shell: wall 'beep'
  changed_when: False
```

12

Sometimes you will know, based on the return code or output that it did not make any changes
* Makes the report output cleaner
* Will not trigger handlers

Examples of uses:
A task that sends a metric (obvs doesn’t change anything but should occur every time)
A notification like the example above