openQA Project - action #66781

coordination # 15132 (Blocked): [saga][epic] Better structure of test plans in main.pm

coordination # 44360 (Blocked): [epic] Parameterize test suites within job groups

coordination # 66727 (Resolved): [epic] Define structure to define test suites not in openQA database

hidden keys in yaml job templates

2020-05-13 07:58 - okurz

Status:	Resolved	**Start date:**	2020-05-13
Priority:	Normal	**Due date:**	
Assignee:	cdywan	**% Done:**	0%
Category:	Feature requests	**Estimated time:**	0.00 hour
Target version:	Current Sprint		
Difficulty:	easy		

Description

Motivation

See #66727 . To reuse same settings among multiple scenarios we can use YAML anchors on one scenario to reuse. However this can make it non-obvious where common settings are defined if the list of scenarios grows as well as needs the defining points to be complete scenarios. With hidden keys like in gitlab CI as defined in https://docs.gitlab.com/ee/ci/yaml/#hide-jobs we can have explicit places where we can find templates to be reusable in multiple scenario definitions

Acceptance criteria

- **AC1:** scenario names with leading dot are not scheduled as tests
- **AC2:** definitions from scenarios with leading dot can be reused in other scenarios

Suggestions

Probably we do not need to change anything in the YAML schema but just ignore the hidden scenarios from evaluating the complete schedule

History

#1 - 2020-05-13 11:41 - tinita

Just a note: YAML Merge Keys << are not performing a "deep merge". It only merges at the top level.
So in the following example the settings would only contain the ANOTHER_SETTING.

```
- .testsuite: &crypt_no_lvm
    description: |
      bla bla bla
    settings:
      YAML_SCHEDULE: schedule/yast/encryption/crypt_no_lvm.yaml
      YAML_TEST_DATA: test_data/yast/encryption/encrypt_no_lvm.yaml
- crypt_no_lvm:
    settings:
      ANOTHER_SETTING: foo
    <<: *crypt_no_lvm
```

Resulting data with resolved merge keys:

```
- .testsuite:
    description: |
      bla bla bla
    settings:
      YAML_SCHEDULE: schedule/yast/encryption/crypt_no_lvm.yaml
      YAML_TEST_DATA: test_data/yast/encryption/encrypt_no_lvm.yaml
- crypt_no_lvm:
    description: |
      bla bla bla
    settings:
      ANOTHER_SETTING: foo
```

#2 - 2020-05-14 10:40 - cdywan

- Status changed from Workable to In Progress

- Assignee set to cdywan

- Target version changed from Ready to Current Sprint

#3 - 2020-05-14 13:55 - cdywan

tinita wrote:

> Just a note: YAML Merge Keys << are not performing a "deep merge". It only merges at the top level.

I filed #66865 since I think that's an orthogonal problem.

Although a new key definitions might be a more versatile alternative to the . notation:

```
definitions:
  foo: &mytemplate:
    description: bla
    settings:
      FOO: bar
```

This would fulfill the AC and support more use cases beyond scenarios i.e. it could be used for *products*, too.

#4 - 2020-05-14 17:34 - okurz

I think we should start with hidden keys as you have it in the PR. Why not just allowing to define anything on the top level of the yaml document instead of a section definitions?

#5 - 2020-05-14 17:45 - cdywan

I proposed two PRs:

- Hidden keys prefixed with a . as suggested
- A new definitions key comparable to CircleCI aliases

okurz wrote:

> I think we should start with hidden keys as you have it in the PR. Why not just allowing to define anything on the top level of the yaml document instead of a section definitions?

Because that sort of thing bites us in the future. We just had that same point come up with the new openqa-cli and the commands.

Trying out both was easy enough so I prepared both.

#6 - 2020-05-15 08:44 - okurz

cdywan wrote:

> Because that sort of thing bites us in the future. We just had that same point come up with the new openqa-cli and the commands.

I don't understand what you mean here. I just like the flexibility of YAML in general and I like the way that gitlab CI handles it with hidden keys on the top level. With our schema we are restricting it which is certainly helpful but do you see problems with something like this?:

```
.base_prio: &base_prio
  priority: 40

defaults:
  i586:
    machine: 64bit
    <<: *base_prio

products:
  opensuse-13.1-DVD-i586:
    distri: opensuse
    flavor: DVD
    version: '13.1'

.my_custom_template: &my_custom_template
  testsuite: null
  priority: 70
```

```
  settings:
    ADVANCED: '1'
    DESKTOP: advanced_kde

scenarios:
  i586:
    opensuse-13.1-DVD-i586:
    - kde_usb:
        settings:
          USB: '1'
        <<: *my_custom_template
```

#7 - 2020-05-15 09:40 - tinita

There is one thing we should be aware of when allowing arbitrary data with "hidden keys" anywhere. This applies to a definitions section as well.

Without applying a schema on these keys users can put any data into it, which could make us vulnerable to the Billion Laughs attack
https://en.wikipedia.org/wiki/Billion_laughs_attack#Variations

Of course we only have authenticated users editing the YAML, and only loading the YAML is not directly dangerous regarding this attack, because in Perl we use references. (There is a minor attack possibility, though, by aliasing large strings, because we can't use references for them in Perl.)
But we have to be careful when dumping the data, encoding it to JSON, for debugging or whatever reason.

But we should think about this and see if we can use a solution where we still can validate all data.
That's one of the advantages of a Schema and I wouldn't want to lose it.

(I actually want to add a protection for this vulnerability in YAML::PP, but it needs more thinking.)

#8 - 2020-05-15 09:52 - tinita

cdywan maybe we can just validate such data by doing a oneOf and add the rules for testsuites, settings etc. in the list (by using aliases). This way we can still prevent any attack.

#9 - 2020-05-15 18:06 - okurz

https://github.com/os-autoinst/openQA/pull/3092 merged. Do you want to still add the oneOf validation?

#10 - 2020-05-15 18:28 - tinita

Maybe we don't need it. We're not directly vulnerable. I think I might try to implement the attack protection in YAML::PP soon.

#11 - 2020-05-15 22:26 - cdywan

- Status changed from In Progress to Feedback

For the record, I observed these 2 types of opaque errors, presumably coming from the validator, with no specifics:

1. [error] YAML::XS::Load Error: The problem:
2. [error] Invalid JSON specification HASH(0x559cab67af40):

okurz wrote:

> https://github.com/os-autoinst/openQA/pull/3092 merged. Do you want to still add the oneOf validation?

I spent a few minutes coming up with validation for the hidden keys. Like we discussed, if it wasn't too much work, I wanted to give it a try. But I'll leave it at almost working, in part because the error handling is very bad.

#12 - 2020-05-19 13:20 - tinita

cdywan wrote:

> For the record, I observed these 2 types of opaque errors, presumably coming from the validator, with no specifics:
>
> 1. [error] YAML::XS::Load Error: The problem:

I'm able to reproduce this error by using invalid YAML.
The problem is that we're stripping anything from the error message after the first line, and YAML::XS errors have the actual error message in multiple lines.

#13 - 2020-05-19 13:25 - tinita

One problem I see with the current regex for hidden keys is that it allows anything because the dot is not escaped.
See my comment here: https://github.com/os-autoinst/openQA/pull/3092/files#r427298894

Edit: preparing a fix.

PR: https://github.com/os-autoinst/openQA/pull/3104

#14 - 2020-05-19 16:32 - tinita

PR https://github.com/os-autoinst/openQA/pull/3104 was merged

#15 - 2020-05-20 08:58 - tinita

Created PR to get the full error message in case the schema contains invalid YAML: https://github.com/os-autoinst/openQA/pull/3108

#16 - 2020-06-12 11:10 - cdywan

- Status changed from Feedback to Resolved