feat!: Rewrite templating. #6442

Closed alexec wants to merge 6 commits into `argoproj:master` from `alexec:map-tmpl`

Conversation 21 Commits 6 Checks 15 Files changed 14

alexec commented on Jul 28, 2021 Collaborator

Signed-off-by: Alex Collins alex_collins@intuit.com

Our current templating system is a source of many edge-case bugs and is full of sticking-plasters to deal with these edge-cases. It also has a security issue that allows users to re-write a workflow using input parameters (see #6441) that cannot be fixed with the current solution.

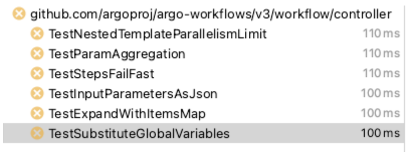
I expect some users are probably using the ability to re-write templates to insert or change workflows, so this is a breaking change.

This PoC changes the solution, so that rather than templating the whole object, we only template values that are strings.

`feat!: Rewrite templating. BREAKING CHANGE` ··· ✗ 78c7925

alexec commented on Jul 28, 2021 Collaborator Author

The following tests are failing:



github.com/argoproj/argo-workflows/v3/workflow/controller	
TestNestedTemplateParallelismLimit	110 ms
TestParamAggregation	110 ms
TestStepsFailFast	110 ms
TestInputParametersAsJson	100 ms
TestExpandWithItemsMap	100 ms
TestSubstituteGlobalVariables	100 ms

TestNestedTemplateParallelismLimit

```
withParam value could not be parsed as a JSON list: [1,2,3,4]\\n: invalid character '\\\\' after top-level value
```

TestParamAggregation

```
                                expected: "[\"odd\",\"even\"]"
                                actual  : "[\\\"odd\\\",\\\"even\\\"]"
```

TestStepsFailFast

```
withParam value could not be parsed as a JSON list: [\\\"a\\\", \\\"b\\\", \\\"c\\\"]\\n: invalid character '\\\\' looking for beginning of value
```

TestInputParametersAsJson

```
                                expected: "Workflow: [{\"name\":\"parameter1\",\"value\":\"value1\"}]. Template: [{\"name\":\"parameter1\",\"value\":\"value1\"},
{\"name\":\"parameter2\",\"value\":\"template2\"}]"
                                actual  : "Workflow: [{\\\"name\\\":\\\"parameter1\\\",\\\"value\\\":\\\"value1\\\"}]. Template:
[{\\\"name\\\":\\\"parameter1\\\",\\\"value\\\":\\\"value1\\\"},{\\\"name\\\":\\\"parameter2\\\",\\\"value\\\":\\\"template2\\\"}]"
```

TestExpandWithItemsMap

```
                                expected: "debian 9.1 JSON({\"os\":\"debian\",\"version\":9.1})"
                                actual  : "debian 9.1 JSON({\\\"os\\\":\\\"debian\\\",\\\"version\\\":9.1})"
```

TestSubstituteGlobalVariables

```
                Error:          "[{\"name\":\"whalesay1\",\"inputs\":{},\"outputs\":{},\"metadata\":{},\"container\":
{\"name\":\"\",\"image\":\"docker/whalesay:latest\",\"command\":[\"cowsay\"],\"args\":[\"mutex1\"],\"resources\":{}}}]" does not contain "{{workflow.parameters.message}}"
```

alexec commented on Jul 28, 2021 Collaborator Author

@mac9416 I'd love to get your thoughts on this.

crenshaw-dev commented on Jul 28, 2021 Contributor

Will take a look!

crenshaw-dev reviewed on Jul 28, 2021

View changes

crenshaw-dev left a comment — Contributor

Having only really looked at `replace.go`, I love the approach. Will read some more and try to see why the tests might be failing / how to fix that.

util/template/replace.go

```
+        if err != nil {
+                return err
+        }
+        data, err = json.Marshal(y)
```

crenshaw-dev on Jul 28, 2021 — Contributor

I *think* you could equivalently call `json.Valid` and return `y` directly. But I guess then you wouldn't be able to bubble up the marshalling error.

crenshaw-dev commented on Jul 28, 2021 — Contributor

I haven't validated this yet, but I wonder if the test failures are because `simpleReplace` still calls `strconv.Quote`.

I'm not sure what issues the `strconv.Quote` was trying to resolve. But if that call was just "don't let the result break the JSON", then I think only running the templating engine on `string` s resolves that issue, and the `strconv.Quote` call can be removed.

`remove strconv.Quote` … ✗ e43ae16

crenshaw-dev commented on Jul 28, 2021 — Contributor

@alexec is that a real test failure or a flake test failure? I get the same thing consistently locally.

alexec commented on Jul 28, 2021 — Collaborator Author

`TestGetPodByNode` did not fail locally for me. I'd like **@sarabala1979** to comment on `TestSubstituteGlobalVariables`, looks like correct behaviour to me.

👍 1

sarabala1979 commented on Aug 2, 2021 • edited — Member

@alexec Template level `Workflow.parameters` has to be substituted during the template execution otherwise template will have a stale value if that parameter is updated during another step/dag execution.
#6401

alexec commented on Aug 2, 2021 • edited — Collaborator Author

It seems to me that we have some edge case behavior going on that it might be hard to compensate for.

sarabala1979 commented on Aug 2, 2021 — Member

> It seems to me that we have some edge case behavior going on that it might be hard to compensate for.

yes this edge case scenario, but it is a real usecase.

you can add below code will work. Templates substitution will happen in `executeTemplate` function with updated value.

```
        execWfSpec := woc.execWf.Spec

        // To Avoid the stale Global parameter value substitution to templates.
        // Updated Global parameter values will be substituted in 'executetemplate' for templates.
        execWfSpec.Templates = nil
```

crenshaw-dev reviewed on Aug 2, 2021

View changes

workflow/controller/operator.go Outdated

```
-                return err
-        }
-        return nil
+        return template.Replace(&woc.execWf.Spec, woc.globalParams, true)
```

crenshaw-dev on Aug 2, 2021 • edited — Contributor

Suggested change

```
-        return template.Replace(&woc.execWf.Spec, woc.globalParams, true)
+        execWfSpec := woc.execWf.Spec
+
+        // To Avoid the stale Global parameter value substitution to templates.
+        // Updated Global parameter values will be substituted in 'executetemplate' for templates.
+        execWfSpec.Templates = nil
+
+        return template.Replace(&execWfSpec, woc.globalParams, true)
```

@sarabala1979 like this?

sarabala1979 on Aug 2, 2021 Member

yes

sarabala1979 on Aug 2, 2021 Member

return template.Replace(&execWfSpec, woc.globalParams, true)

👍 1

alexec commented on Aug 2, 2021 Collaborator Author

I'm closing this PR. I think it is too risky.

alexec closed this on Aug 2, 2021

crenshaw-dev commented on Aug 2, 2021 Contributor

@alexec should I continue working on #6285?

alexec commented on Aug 2, 2021 Collaborator Author

I really don't know what the best solution is with templates. Bugs are a game of whack-a-mole.

crenshaw-dev commented on Aug 2, 2021 Contributor

Given:

1. expression templates introduce a security vulnerability and
2. the bug also eliminates a major (the primary?) use case for expression templates (the possibility to generate JSON lists containing strings to use with `withParam`)

I feel like it's worth either playing a bit more whack-a-mole or removing the expression template feature.

Is there any set of tests/validations we could perform on #6285 which would provide sufficient confidence that it is safe? Would be happy to schedule a call to discuss what that would look like.

alexec linked an issue on Aug 2, 2021 that may be closed by this pull request

workflow re-write vulnerability using input parameter #6441 Closed

alexec commented on Aug 2, 2021 Collaborator Author

It was my understanding from you email that the vulnerability (#6441) exists for any template. It sound like you think it only exists for *expression tag templates*. Users can remove the vulnerability by starting the controller using `EXPRESSION_TEMPLATES=false` , but that is not helpful if you are already using that feature.

Can you please double-confirm if this is only a problem with expression templates?

That given, if this is a new issue, or an existing issue, the we should have a plan to fix it. That doesn't need to be in v3.1, but should probably be in v3.2 (releases take 3 months to rollout).

This PR (#6442) is a big change, and therefore risky, so I'd want to do more testing. I'm not sure that'd make it in time for v3.2

Your PR (#6285) isolates changes to the expression code. So this is low risk and suitable for back-porting to v3.1.

crenshaw-dev commented on Aug 2, 2021 • edited Contributor

> It sound like you think it only exists for expression tag templates. Users can remove the vulnerability by starting the controller using EXPRESSION_TEMPLATES=false, but that is not helpful if you are already using that feature.

That is all correct. I should have made the email more clear. I believe the *non-expression tag template* code avoids the vulnerability by running the evaluated expression value through `strconv.Quote` , ensuring the value *cannot* escape its JSON string container. (As far as I can tell, injecting an unescaped `"` is the only way to escape the containing string and modify the JSON-encoded workflow. `strconv.Quote` escapes all double-quotes.)

> This PR (#6442) is a big change, and therefore risky, so I'd want to do more testing. I'm not sure that'd make it in time for v3.2

Yep. I *like* this PR more because it clearly isolates expressions (and their evaluated values) to the strings in which they're defined. But I agree, it's riskier.

> Your PR (#6285) isolates changes to the expression code. So this is low risk and suitable for back-porting to v3.1.

In that case, I'll spend some time validating the metrics template behavior (I'm just slightly uneasy about that code) and then un-mark it as a draft.

Still happy to hop on a call if anything here could use clarification or you have any guidance on additional work you'd like to see done for #6285.

`Merge branch 'master' into map-tmpl` … e7b7598

alexec commented on Aug 2, 2021 Collaborator Author

Lets go with your PR for v3.1, and maybe in v3.2 or v3.3 we can consider this more riskly change.

`revert` … ✓ 5b96b80

alexec mentioned this pull request on Sep 8, 2021

Expression results are printed incorrectly at random #6673

Closed

alexec reopened this on Sep 22, 2021

alexec changed the title ~~feat!: Rewrite templating. BREAKING CHANGE~~ **feat!: Rewrite templating.** on Sep 22, 2021

alexec added 2 commits last year

`Merge branch 'master' into map-tmpl` … ✗ `ee6b6b3`

`ok` … ✓ `f1d58e6`

alexec closed this on Sep 22, 2021

alexec deleted the `map-tmpl` branch 7 months ago

Reviewers

crenshaw-dev

sarabala1979

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

workflow re-write vulnerability using input parameter

3 participants