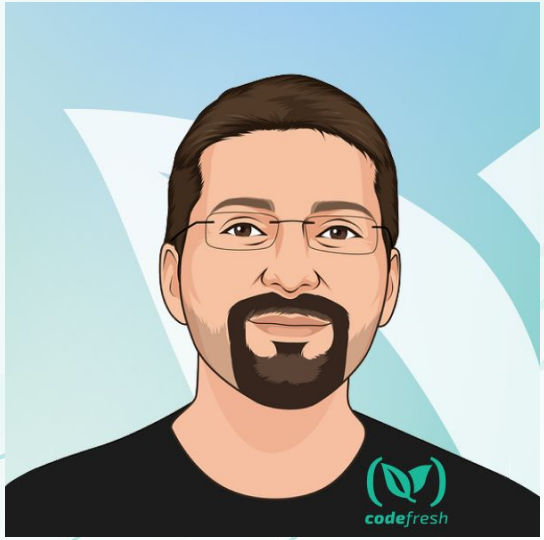
codefresh

How to Preview and Diff Your Argo CD Deployments

Argo Con 2023

Kostis Kapelonis

- Developer Advocate at Codefresh and Argo contributor
- Codefresh is the Enterprise Platform for Argo
- Co-author of GitOps certification with Argo -> http://learning.codefresh.io

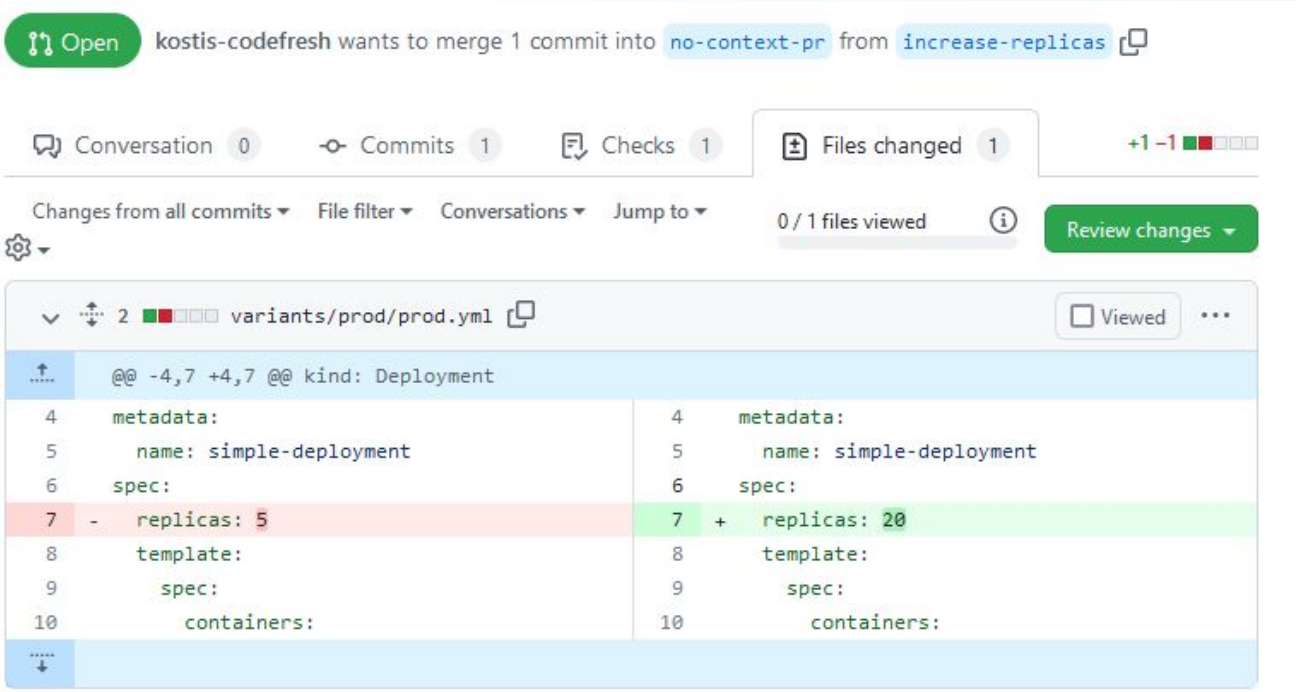


codefresh

Let's set the stage

Problem statement

No Context Diff



https://github.com/kostis-codefresh/argocd-preview-diff/pull/3

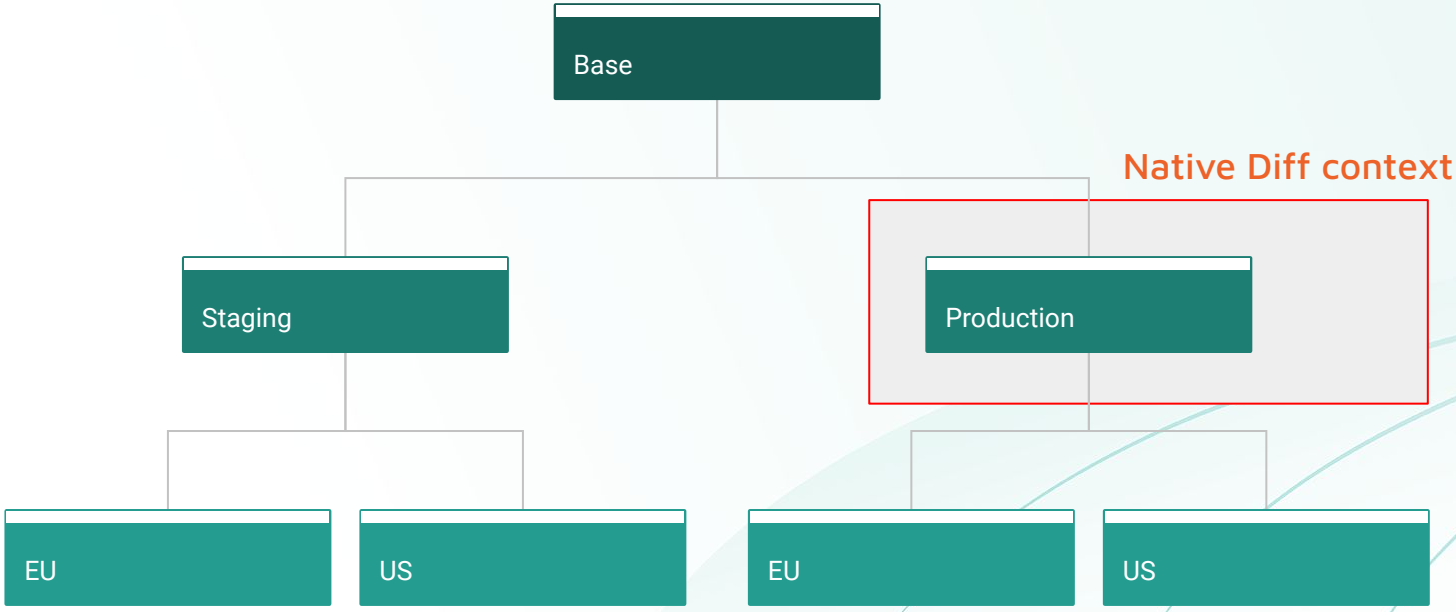
codefresh

Diffs without context

- You merge this and ….nothing happens
- Typical scenario of many Kustomize overlays
- Same issue with hierarchy of Helm values

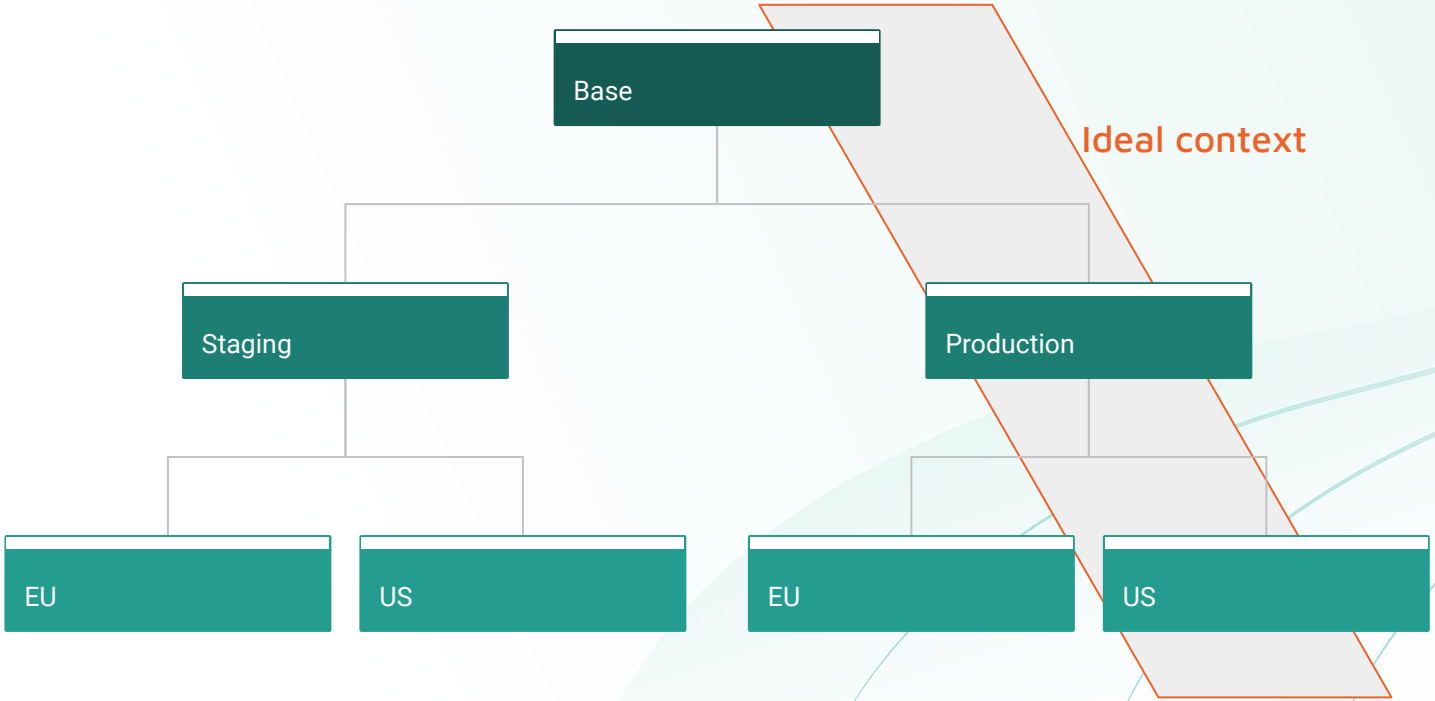
codefresh

Hierarchy of Overlays/value files



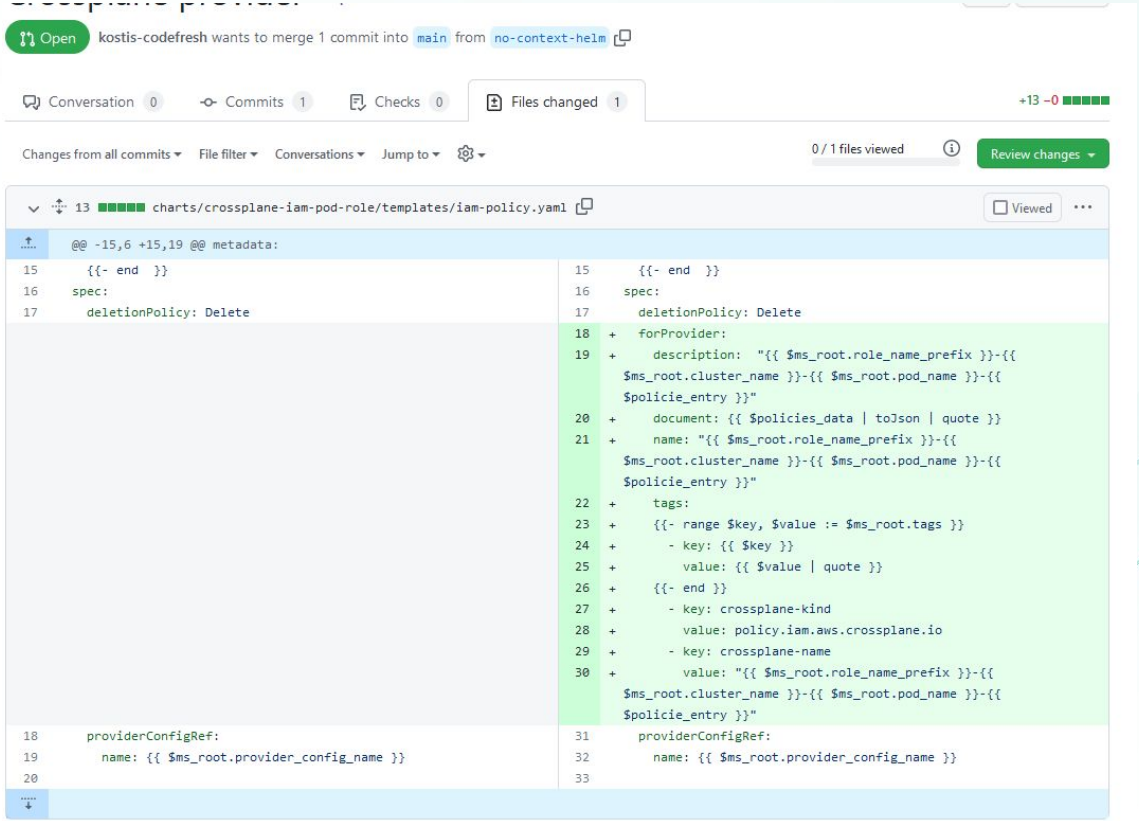
codefresh

Hierarchy of Overlays/value files



codefresh

No Context Helm Diff



Open kostis-codefresh wants to merge 1 commit into main from no-context-helm

Conversation 0 · Commits 1 · Checks 0 · Files changed 1 · +13 −0

Changes from all commits · File filter · Conversations · Jump to · 0 / 1 files viewed · Review changes

13 charts/crossplane-iam-pod-role/templates/iam-policy.yaml · Viewed

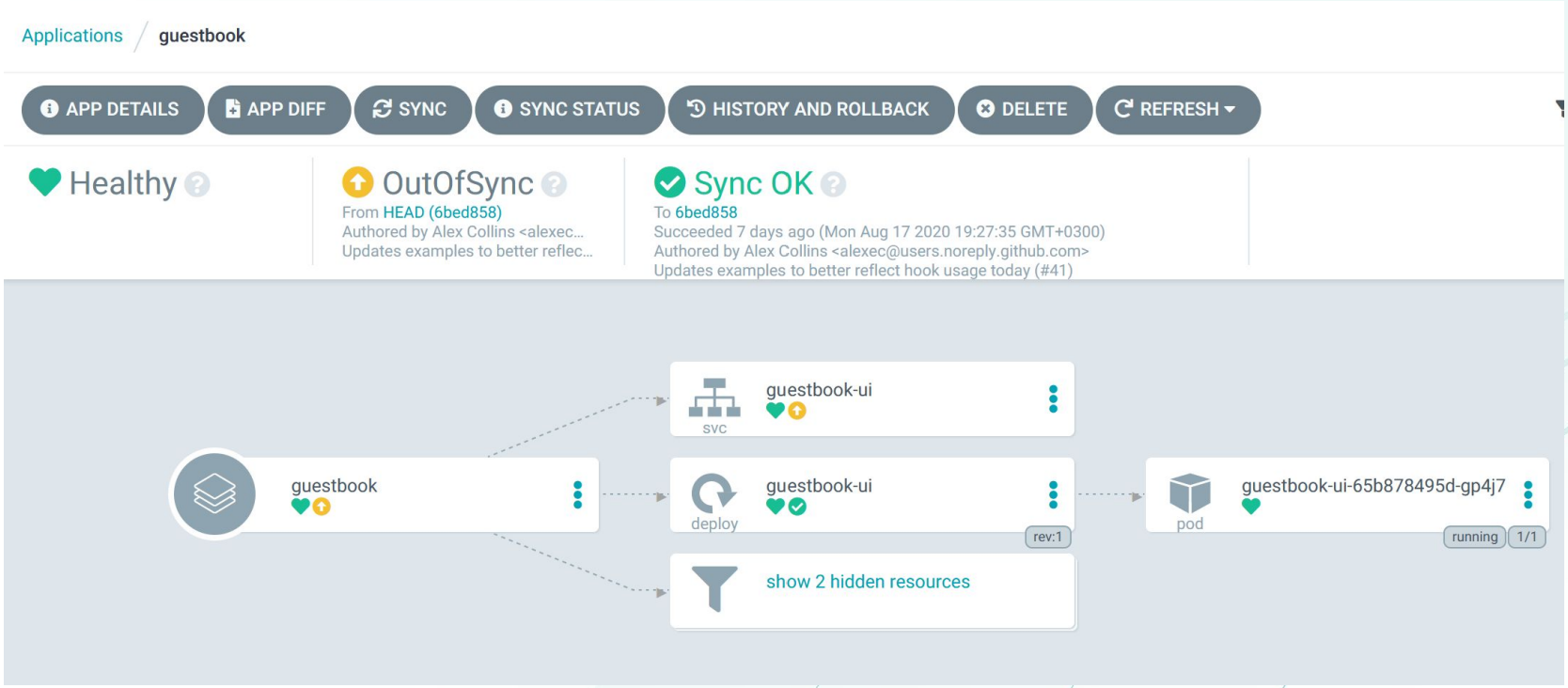
```
@@ -15,6 +15,19 @@ metadata:
    {{- end  }}
  spec:
    deletionPolicy: Delete
+   forProvider:
+     description:  "{{ $ms_root.role_name_prefix }}-{{ $ms_root.cluster_name }}-{{ $ms_root.pod_name }}-{{ $policie_entry }}"
+     document: {{ $policies_data | toJson | quote }}
+     name: "{{ $ms_root.role_name_prefix }}-{{ $ms_root.cluster_name }}-{{ $ms_root.pod_name }}-{{ $policie_entry }}"
+     tags:
+     {{- range $key, $value := $ms_root.tags }}
+       - key: {{ $key }}
+         value: {{ $value | quote }}
+     {{- end }}
+       - key: crossplane-kind
+         value: policy.iam.aws.crossplane.io
+       - key: crossplane-name
+         value: "{{ $ms_root.role_name_prefix }}-{{ $ms_root.cluster_name }}-{{ $ms_root.pod_name }}-{{ $policie_entry }}"
    providerConfigRef:
      name: {{ $ms_root.provider_config_name }}
```

codefresh

Possible solutions

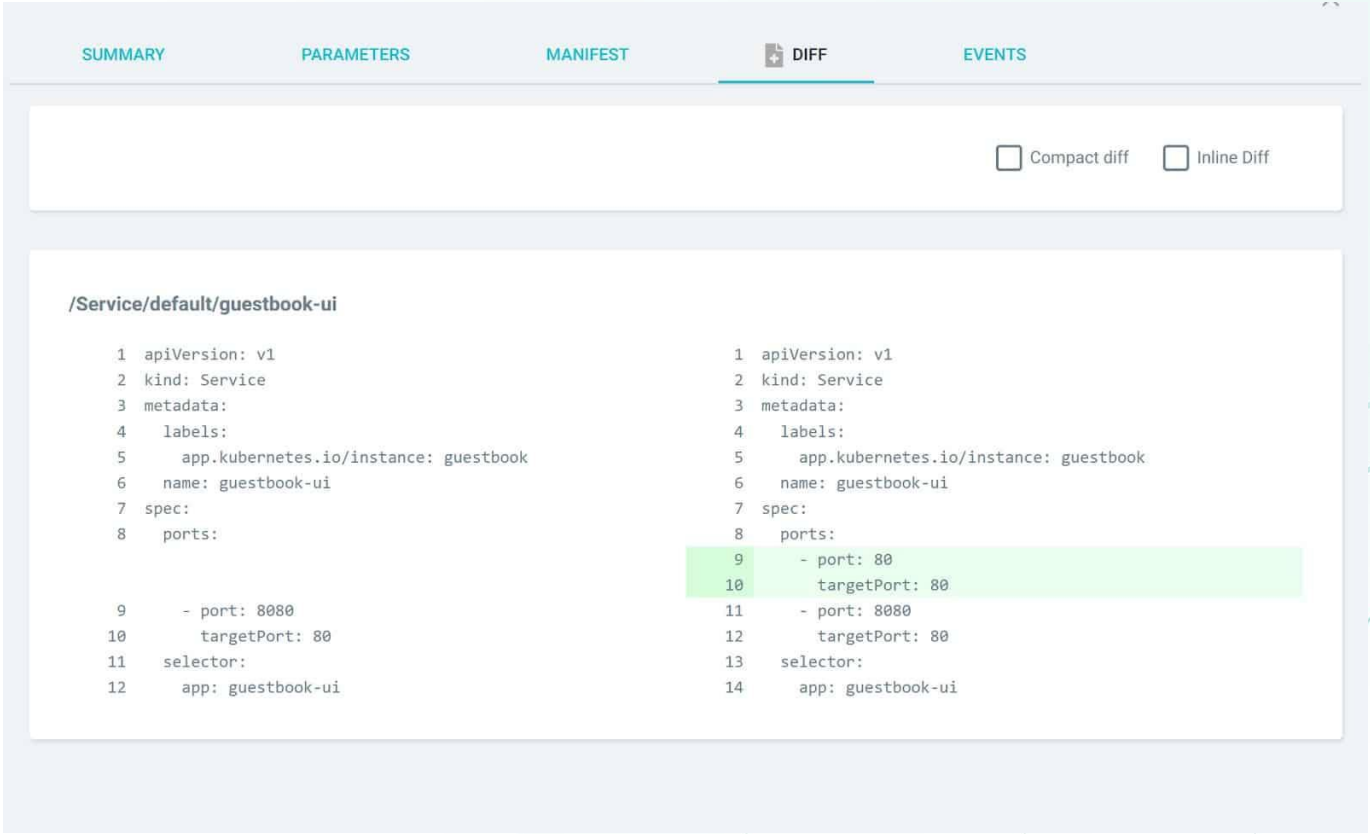
2. Use Argo CD diff in UI

Argo CD UI Diff



codefresh

Argo CD UI Diff



codefresh

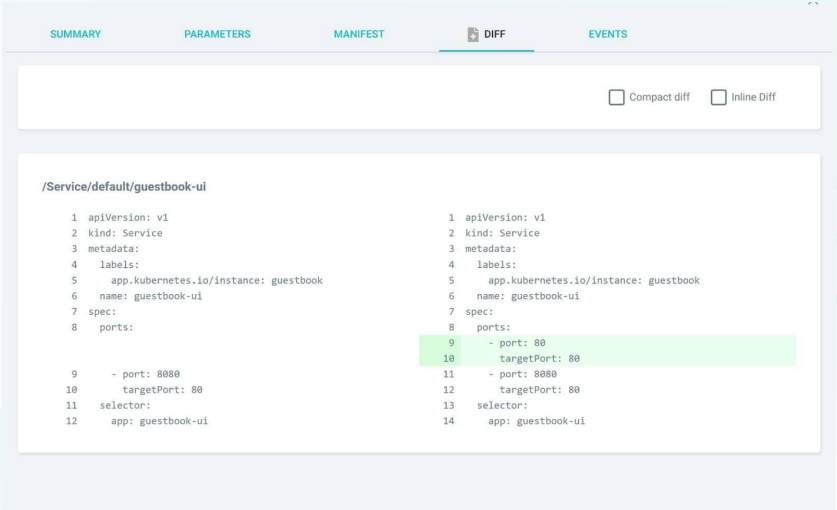
Argo CD UI Diff

✅ - Preview with full context
✅ - Built into Argo CD
✅ - Zero setup/maintenance
✅ - Support for Kustomize/Helm
❌ - Doesn’t work with auto-sync
❌ - Only show changes after push
❌ - Diff is shown too late in the process

codefresh

Argo CD UI Diff Recommendation

Could be used for sanity checking in production environments



codefresh

Possible solutions

3. Use Argo CD Local diff

Argo CD local diff

- The Argo CD CLI allows you to diff an application between the cluster and local manifests
- Native support for Kubernetes/Helm

codefresh

Argo CD local diff

```
root@kubernetes-vm:~/workdir/gitops-cert-level-2-examples/environment-promotion# argocd app diff qa --local envs/qa
INFO[0000] kustomize build envs/qa                       dir= execID=08403
INFO[0000] Trace                                         args="[kustomize build envs/qa]" dir= operation_name="exec

===== /Service qa/qa-simple-service ======
60c60
<     nodePort: 31000
---
>     nodePort: 31001

===== apps/Deployment qa/qa-simple-deployment ======
153c153
<   replicas: 3
---
>   replicas: 5
190c190
<           value: non-prod-user
---
>           value: new-non-prod-user
192c192
<           value: non-prod-password
---
>           value: new-non-prod-password
```

codefresh

Argo CD local diff in CI

1. Open a Pull request with suggested changes
2. Have the CI system checkout the PR
3. Run in a CI pipeline “argocd diff –local” against the cluster where the PR is destined
4. Present the diff to the user in order to view the full context

codefresh

Argo CD Local Diff

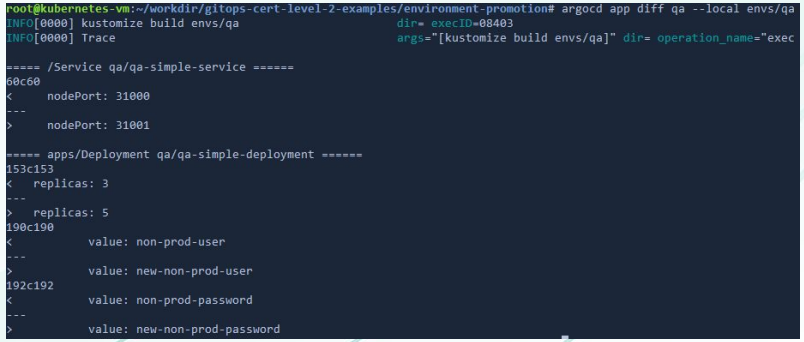
✅ - Built into Argo CD
✅ - Zero setup/maintenance
✅ - Support for Kustomize/Helm
✅ - Preview with full context
✅ - No need to commit/push anything
❌ - Need direct access to Argo CD cluster
❌ - Needs network setup rules
❌ - Not scalable for multiple clusters
❌ - Problematic for Edge/Remote setups

codefresh

Argo CD Local Diff Recommendation

Great for local experimentation and quick ad-hoc checks.

Not recommended for other uses



codefresh

Possible solutions

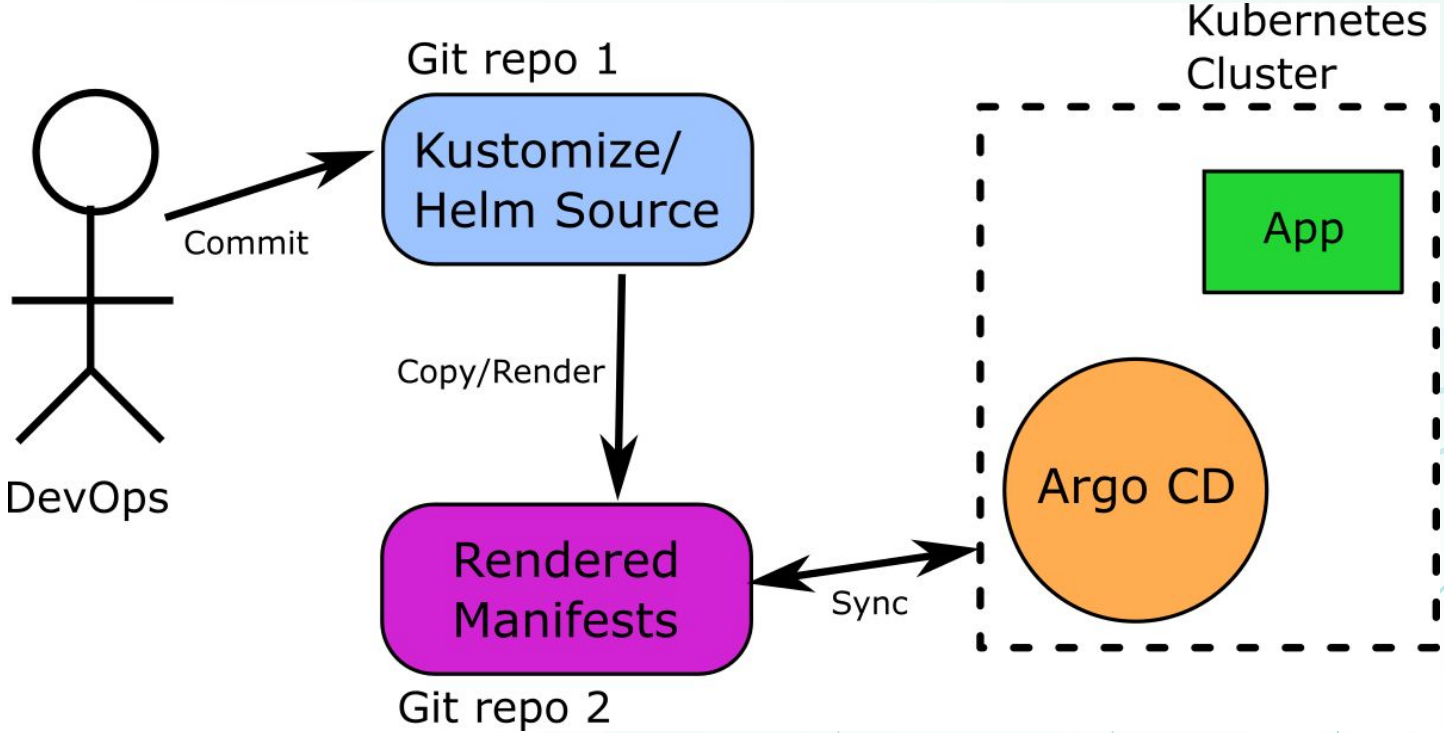
4. Pre-render Manifests

Pre-Render manifests in second Git repo

1. Use 2 GitOps repositories for each application/cluster
2. First Git repository has unprocessed files (e.g. charts/overlays)
3. Second Git repository has final rendered manifests
4. Argo CD is pointed at the second Git repository
5. An automated process renders manifests and commits them to the second repository

codefresh

Pre-Render manifests in second Git repo

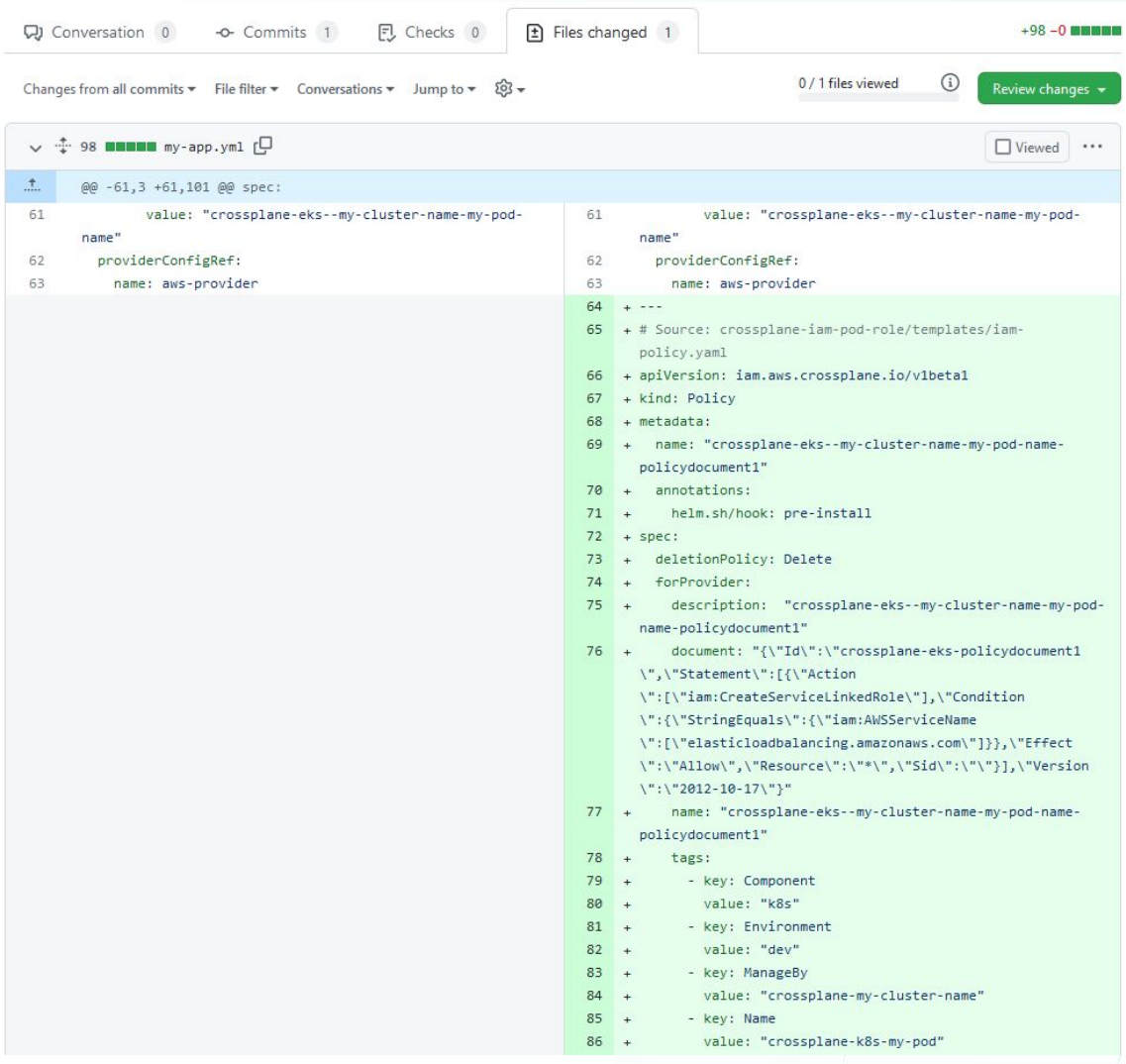


codefresh

Pre-Render manifests in second Git repo

1. A human creates a PR on “source” Git repo
2. A “copy” process takes content and renders manifests (using Helm/Kustomize)
3. A second PR is opened automatically in the “Rendered” Git repo
4. A human sees diff in the second repo
5. If PR is approved it is merged in both repos
6. Argo CD always monitors the “Rendered” Git repo

codefresh



Get full context of everything
Great for Helm apps

codefresh

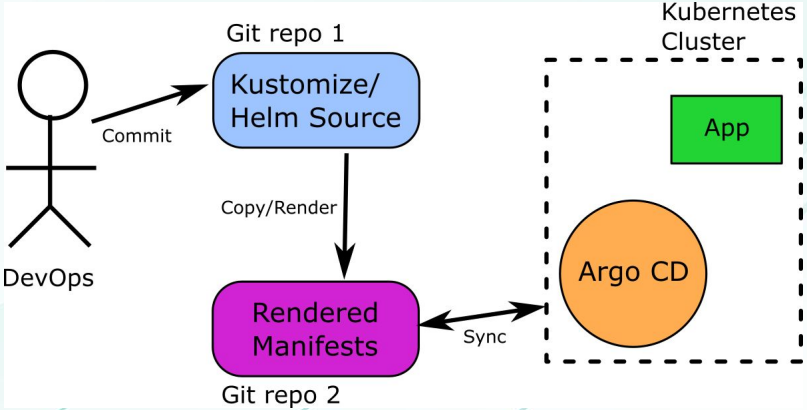
Pre-Render manifests in second Git repo

✅ - Preview with full context
✅ - Use native Git diff
✅ - Perfect fit for Helm apps
❌ - Needs setup/maintenance
❌ - More moving parts/Harder to debug
❌ - CI becomes a point of failure
❌ - Bypasses Argo CD support for Helm/Kustomize
❌ - Doubles number of Git repositories
❌ - Caution not to commit final manifests

codefresh

Pre-Render manifests Recommendation

Several companies use this with success. Needs well disciplined teams and has more moving parts



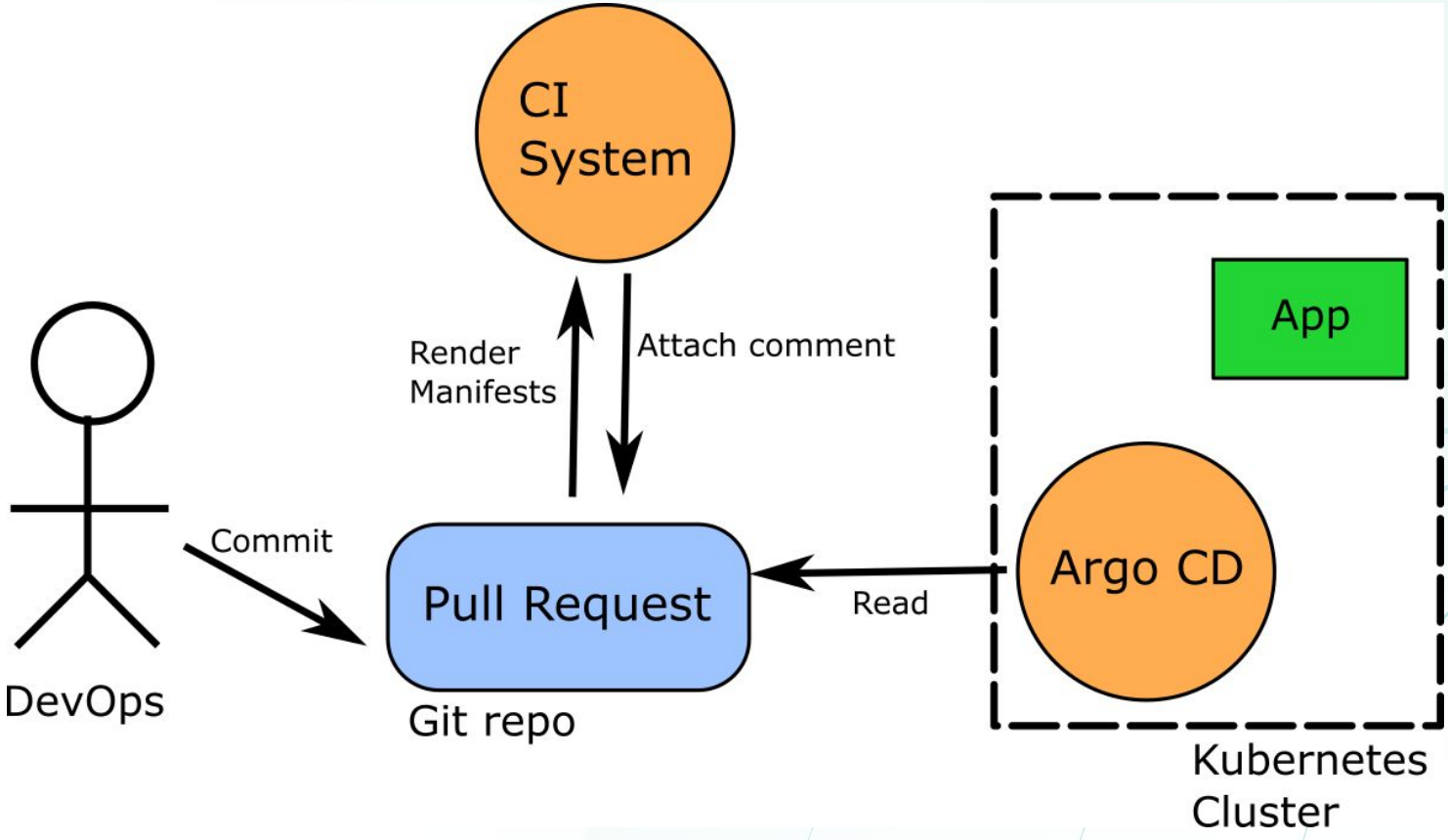
codefresh

Render manifests on the fly

Render manifests on the fly

- Current Git state has what is in the cluster
- PR state has proposed changes to the cluster
- Run a diff between those two

Render manifests on the fly

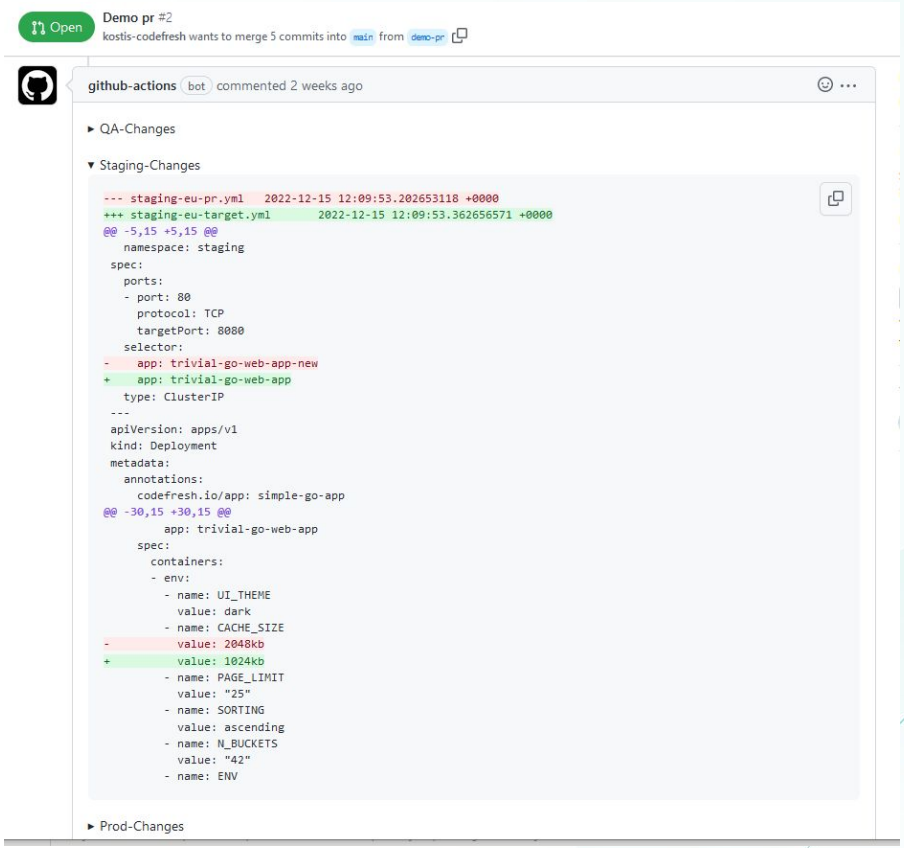


codefresh

Render manifests on the fly

1. A human opens a PR on the manifest repo
2. We checkout out the PR and run Kustomize/Helm
3. We checkout the target of the PR and run Kustomize/Helm again
4. We run a diff between the final rendered manifests
5. We present the diff to the human

Render manifests on the fly



Open Demo pr #2

kostis-codefresh wants to merge 5 commits into main from demo-pr

github-actions bot commented 2 weeks ago

▸ QA-Changes

▾ Staging-Changes

```
--- staging-eu-pr.yml   2022-12-15 12:09:53.202653118 +0000
+++ staging-eu-target.yml       2022-12-15 12:09:53.362656571 +0000
@@ -5,15 +5,15 @@
   namespace: staging
 spec:
   ports:
   - port: 80
     protocol: TCP
     targetPort: 8080
   selector:
-    app: trivial-go-web-app-new
+    app: trivial-go-web-app
   type: ClusterIP
 ---
 apiVersion: apps/v1
 kind: Deployment
 metadata:
   annotations:
     codefresh.io/app: simple-go-app
@@ -30,15 +30,15 @@
         app: trivial-go-web-app
     spec:
       containers:
       - env:
         - name: UI_THEME
           value: dark
         - name: CACHE_SIZE
-          value: 2048kb
+          value: 1024kb
         - name: PAGE_LIMIT
           value: "25"
         - name: SORTING
           value: ascending
         - name: N_BUCKETS
           value: "42"
         - name: ENV
```

▸ Prod-Changes

▾ Prod-Changes

```
--- prod-us-pr.yml      2022-12-15 12:09:52.898646558 +0000
+++ prod-us-target.yml  2022-12-15 12:09:53.050649838 +0000
@@ -5,26 +5,26 @@
   namespace: prod
 spec:
   ports:
   - port: 80
     protocol: TCP
     targetPort: 8080
   selector:
-    app: trivial-go-web-app-new
+    app: trivial-go-web-app
   type: ClusterIP
 ---
 apiVersion: apps/v1
 kind: Deployment
 metadata:
   annotations:
     codefresh.io/app: simple-go-app
   name: prod-us-simple-deployment
   namespace: prod
 spec:
-  replicas: 12
+  replicas: 10
   selector:
     matchLabels:
       app: trivial-go-web-app
   template:
     metadata:
       labels:
         app: trivial-go-web-app
```

https://github.com/kostis-codefresh/argocd-preview-diff/pull/2

codefresh

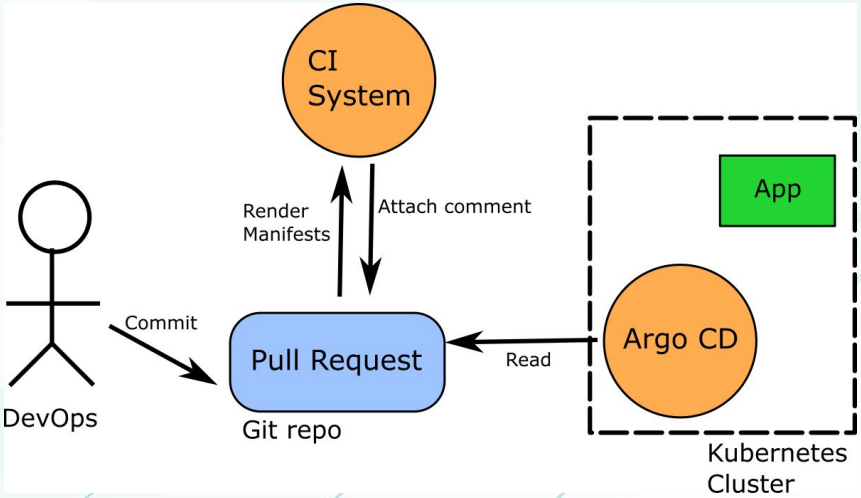
Render manifests on the fly

✅ - Preview with full context
✅ - Based only on Git files
✅ - No access needed to Argo CD API
✅ - Works with any topology
✅ - Uses Argo CD support for Helm/Kustomize
✅ - No confusion over raw/final manifests
✅ - GitOps tool agnostic (could work with Flux)
❌ - Needs setup/maintenance
❌ - Might miss some corner cases

codefresh

Render manifests on the fly recommendation

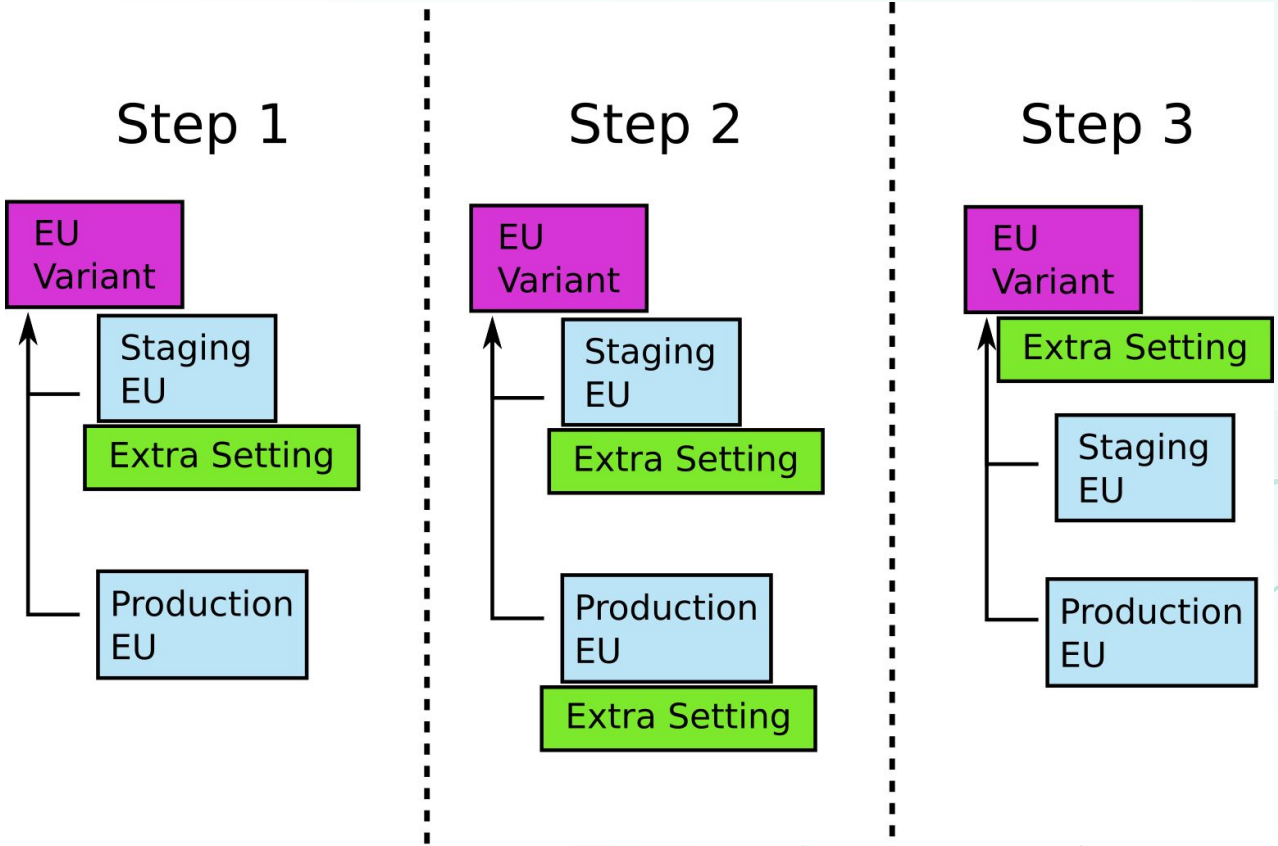
It is simple, robust and scalable. Use it!



codefresh

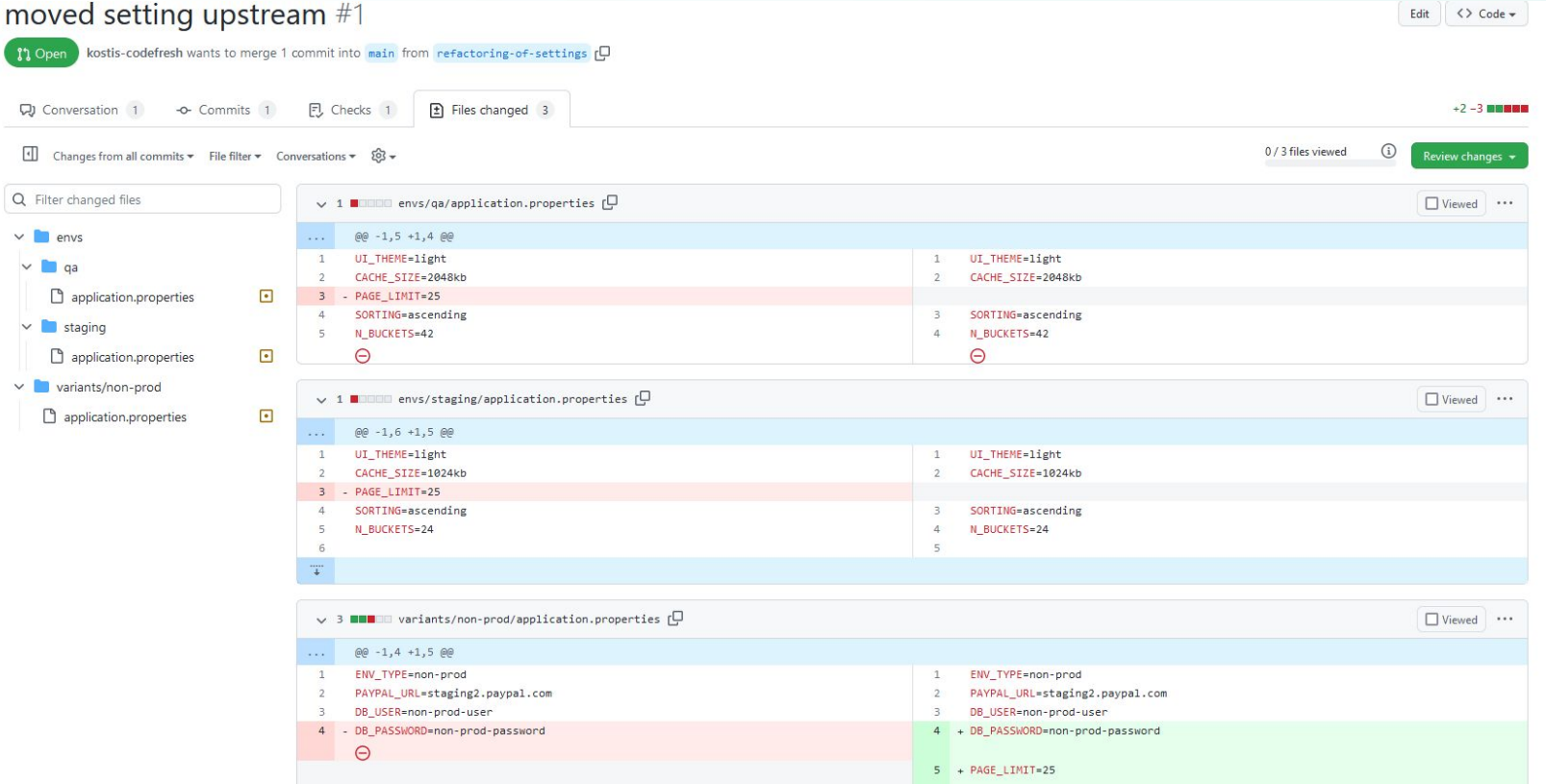
Bonus: Enforce zero changes

Kustomize refactoring



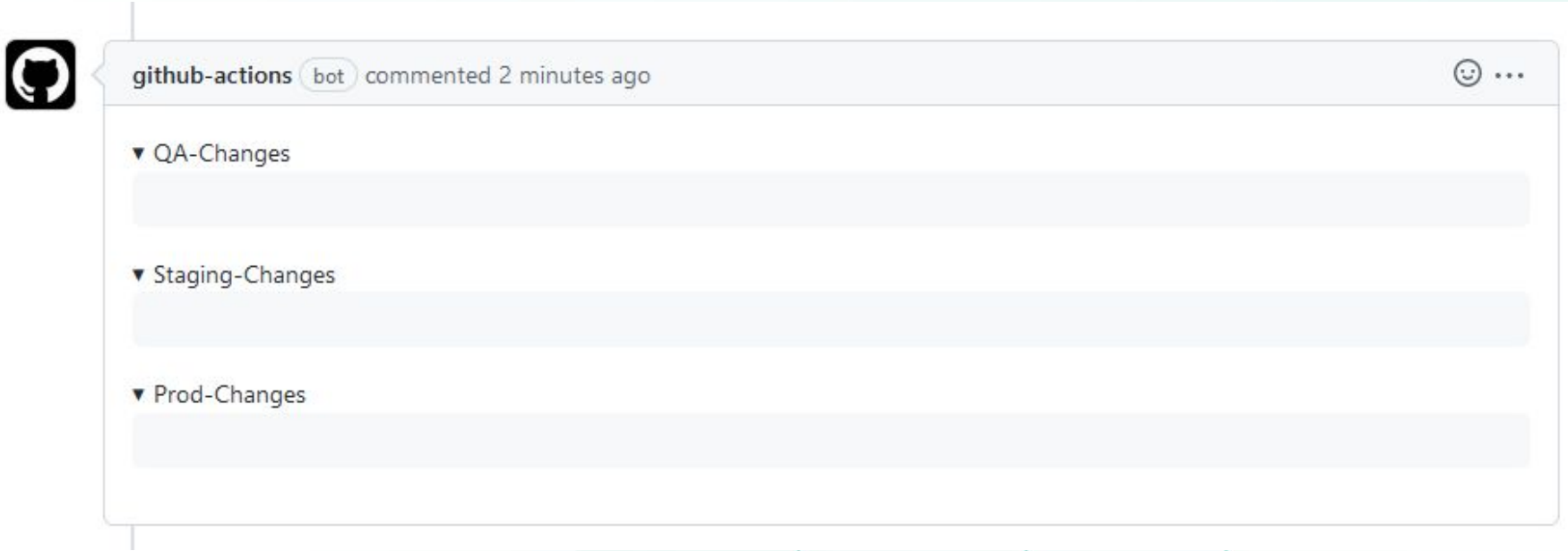
codefresh

Kustomize refactoring



codefresh

Argo CD will not do anything at all



https://github.com/kostis-codefresh/manifest-refactoring/pull/1

codefresh

5 ways to understand the context of Manifest changes

1. Native Git diff
2. Argo CD UI Diff
3. Argo CD CLI Local Diff
4. Pre-render manifests on second Git repo
5. Render manifests on the fly 

Resources

- https://codefresh.io/blog/argo-cd-preview-diff/
- https://www.runatlantis.io/
- http://learning.codefresh.io (Argo CD certification)

Intermission: Terraform plans

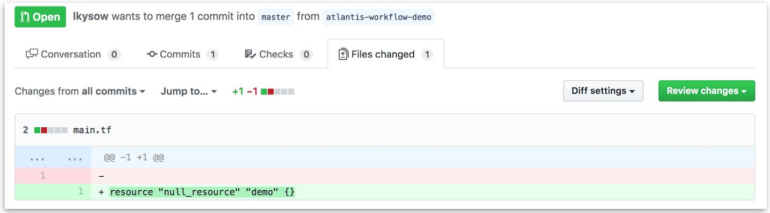
Terraform plan

Get a preview on what will change

```
team@Azure:~/infrastructure$ terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  + azurerm_resource_group.production-america
      id:       <computed>
      location: "westus2"
      name:     "production-america"
      tags.%:   <computed>

  + azurerm_resource_group.production-europe
      id:       <computed>
      location: "westeurope"
      name:     "production-europe"
      tags.%:   <computed>


Plan: 2 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------

Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.

team@Azure:~/infrastructure$
```

codefresh

STEP 1 Open a Terraform pull request



STEP 2 Atlantis automatically runs `terraform plan` and comments back on the pull request

atlantisbot commented

Ran Plan in dir: `.` workspace: `default`

```
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

+ null_resource.demo
      id: <computed>
Plan: 1 to add, 0 to change, 0 to destroy.
```

- To **apply** this plan, comment:
  - `atlantis apply -d .`
- To **delete** this plan click here
- To **plan** this project again, comment:
  - `atlantis plan -d .`

- To **apply** all unapplied plans, comment:
  - `atlantis apply`

runatlantis.io

codefresh