How we preview Kubernetes changes at Chime

By Nicolas Richard, Chime Software Engineer

Talent at Chime · Follow

Published in Life at Chime · 7 min read · 6 hours ago

69

Open in app ↗ Sign up Sign in

Medium Search Write

At Chime, we're committed to GitOps, and all of our application and infrastructure changes are carried out with reviewed and approved pull requests. We’re also committed to safety: every change has the potential for unanticipated consequences, and we need to reduce the likelihood of an availability incident when making changes.

Our commitment as the Infrastructure Engineering team at Chime to prevent outages is deeply rooted in our member-obsessed culture. We recognize that our work has a direct impact on the experiences of our members, and we strive to ensure that our infrastructure is reliable and stable.

In this article, we’ll share a tool we’ve built to easily preview changes to our Kubernetes (k8s) resources before they are applied.

This tool enables Chime to merge changes confidently, resulting in an average of a thousand deployments a day.

Making changes to k8s resources in the dark

All of Chime’s k8s resources are managed via Helm charts that are applied by Argo CD on our EKS clusters. The Helm charts are centralized in a dedicated repository. To make changes to services, Chime engineers find themselves editing both charts and override files, and submitting PRs, and when the changes are approved and merged, they are picked up and automatically reconciled by ArgoCD.

In this scenario, there is a heavy responsibility on both the author and the PR reviewer to make sure they understand and can accurately foresee what the changes will actually accomplish. This sometimes requires detailed knowledge of Kubernetes, Helm, and the implementation of our charts, making PR reviews time-consuming and risky. For example, a PR can be risky if engineers deploy a change that results in a Chime service becoming unavailable, which may or may not result in members losing access to certain features and will almost certainly disrupt Chime’s internal operations.

In other words, this is reasonable for simple changes like, say, incrementing a replicaCount. When it comes to making broad changes, like updates to the shared application charts that are used by many Chime services, engineers have to be certain that they are not introducing unexpected changes that may have consequences.

This problem is made particularly acute by the fact that some of the charts we use have grown over time to include a large number of branching and parameterization.

Some of this heavy parameterization is directly tied to the lack of visibility into the changes an engineer might attempt to make. When the stakes are high, and it’s difficult to preview the “net result” of the changes one is making, it’s only natural that the engineers act cautiously by making use of feature flags to isolate and minimize the blast radius of their changes. As you may have guessed, this impacts the team’s velocity, and large refractors are rendered practically impossible.

This comes in stark contrast to the workflow of another tool Chime’s infrastructure team relies on to make changes to services: Terraform. Indeed Terraform’s VCS-driven workflow automatically triggers `terraform plan` based on changes to our infrastructure-as-code repository. The result is visible directly from the pull request, making seeing and reviewing the “net result” of code changes straightforward.

As engineers got used to this convenient access to a “crystal ball” for their Terraform changes, it became clear that our workflow for k8s changes would be dramatically improved by having a similar feature.

We got to thinking: Wouldn’t it be great if there was a way to preview the outcome of changes before they were released? What if fully rendered manifests were automatically visible directly in the pull request? The author could know the exact outcome of their changes before even submitting the PR for review. It would also make the PR reviewer’s job immensely easier since there would be less guesswork involved.

Thanks to ‘mani-diffy’, any pull request that results in changes to Kubernetes resources is accompanied by a fully rendered manifest illustrating exactly what is changing.

Before we explain how "mani-diffy" works, it'll be helpful to review how our k8s resources are organized using the app of apps pattern.

AOA pattern primer and how we use it at Chime

If you are already familiar with the concepts of AOA and Argo CD, you may skip over this section. I'll be using the repo's example "demo" to showcase how the AOA pattern can be used.

The root and the tree of apps

The App of Apps Pattern lets us define a root ArgoCD Application (or bootstrap). Rather than point to an application manifest, the Root App points to the directory in the repository that contains Application manifests.

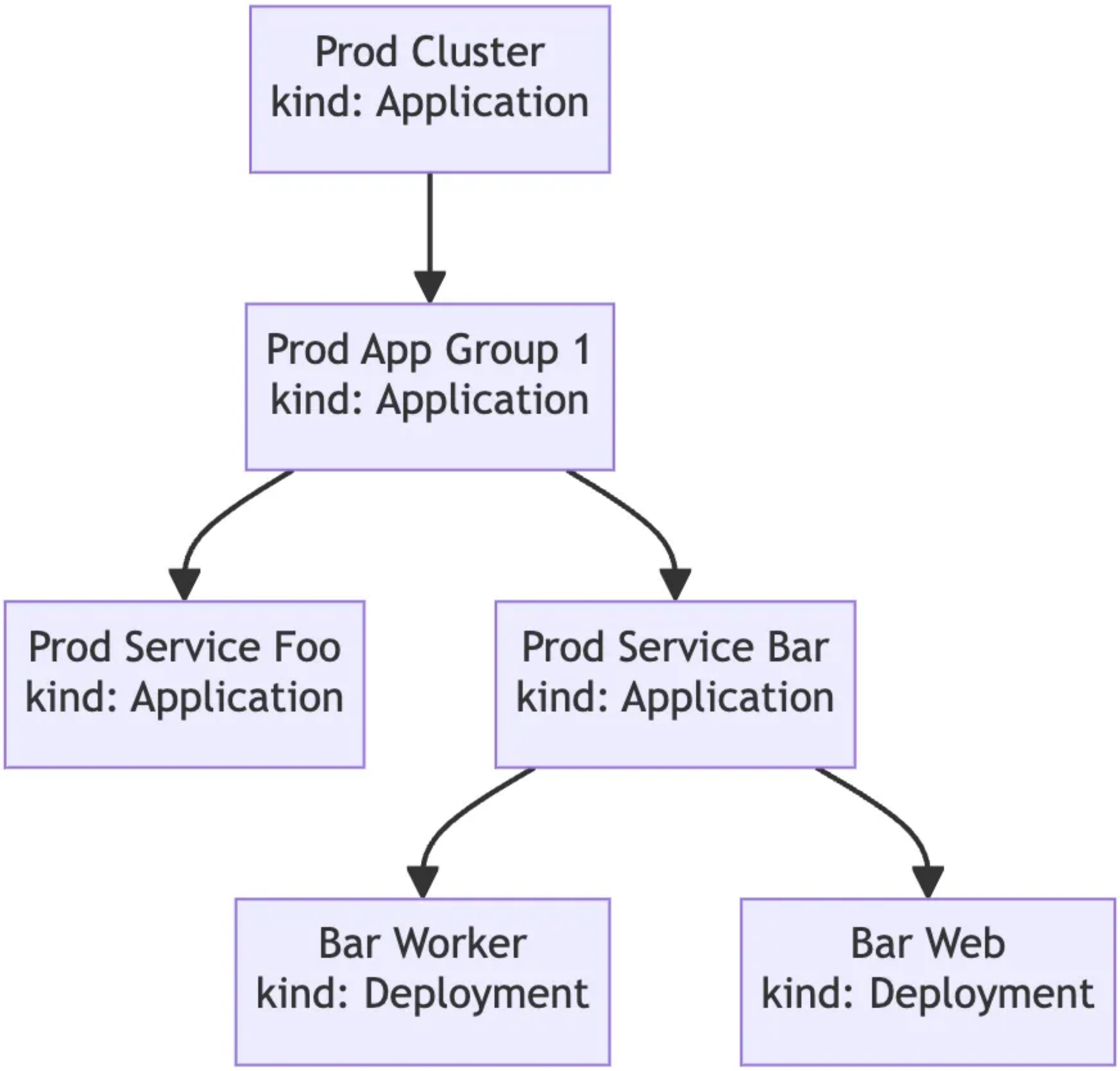
In this example the root app points to "/demo/bootstrap". In this directory, we have 2 Application manifests, a prod and a test cluster Ex : demo/bootstrap/prod-cluster.yaml

These applications point to the "app-of-apps" chart via the \`spec.source.path\` key.

When rendered this chart in turn will create more ArgoCD applications, which point to more charts.

In this example, we have only 2 charts, so it will be either to the "app-of-apps" chart or to the "service" chart which contains all the necessary k8s resources for a service to be created (ie. deployments, etc.).

So, from the root app all the way to the service, we have a tree of ArgoCD applications.



The AOA pattern and overrides

One of the strengths of the AOA pattern is that it makes it straightforward to use overrides.

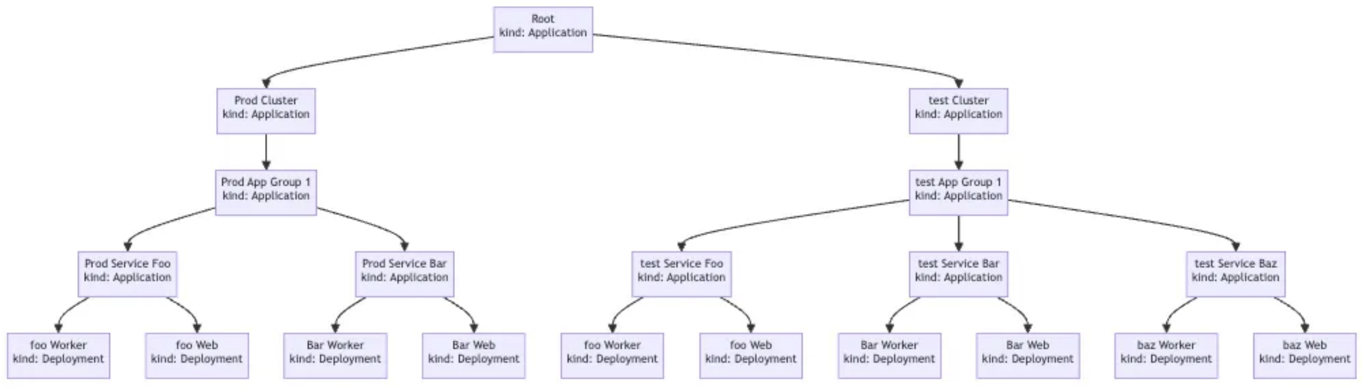
In the demo below, we have a prod and a test cluster with the same tree of ArgoCD applications, albeit configured with a few appropriate tweaks per environment.

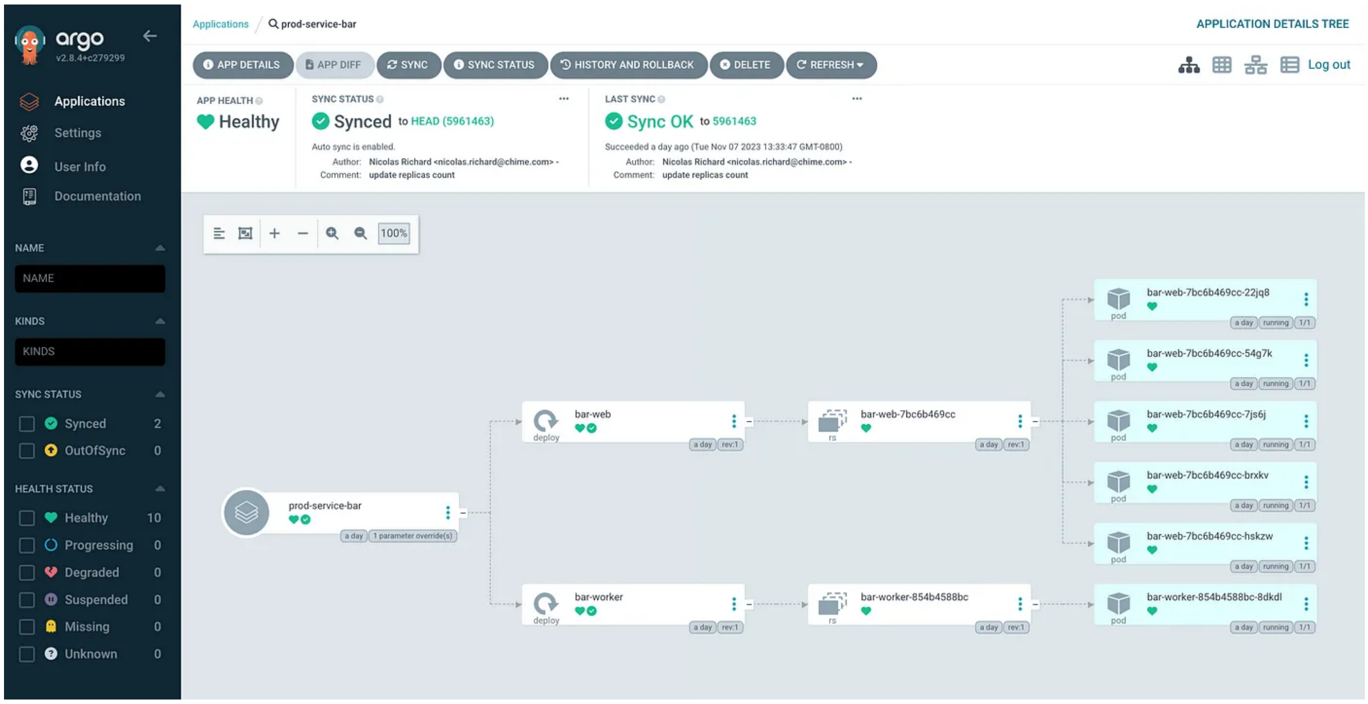
These overrides are passed to the charts via the “valueFiles” key.

The Helm charts contain dynamic logic to include the appropriate valueFiles for a service based on name and environment.

Ex: there is one extra service (baz) in the test cluster

Ex2: the replica count for the services in the test cluster is lower


Root kind: Application
Prod Cluster kind: Application
test Cluster kind: Application
Prod App Group 1 kind: Application
test App Group 1 kind: Application
Prod Service Foo kind: Application
Prod Service Bar kind: Application
test Service Foo kind: Application
test Service Bar kind: Application
test Service Baz kind: Application
foo Worker kind: Deployment
foo Web kind: Deployment
Bar Worker kind: Deployment
Bar Web kind: Deployment
foo Worker kind: Deployment
foo Web kind: Deployment
Bar Worker kind: Deployment
Bar Web kind: Deployment
baz Worker kind: Deployment
baz Web kind: Deployment



Making simple changes like incrementing a replica count is easily done with the overrides, and it's easy to anticipate what the change will be once deployed…

This becomes more difficult when we are talking about updating the charts that compose the tree, such as app-of-apps.

And this is where "mani-diffy" shines — by giving us the confidence to make such changes.

Introducing "mani-diffy"

Mani-diffy walks the hierarchy of Argo CD Application templates, renders the Kubernetes manifests from the input templates and commits the rendered files back for the user to review and validate.

It is designed to be called from a CI job within a pull request (we do it with a GitHub action (see generate-manifests-demos.yaml)), enabling the author to update templates and see the resulting manifests directly within the pull request before the changes are applied to the Kubernetes cluster.

The rendered manifests are kept within the repository, making diffs between revisions easy to parse, dramatically improving safety when updating complex application templates.

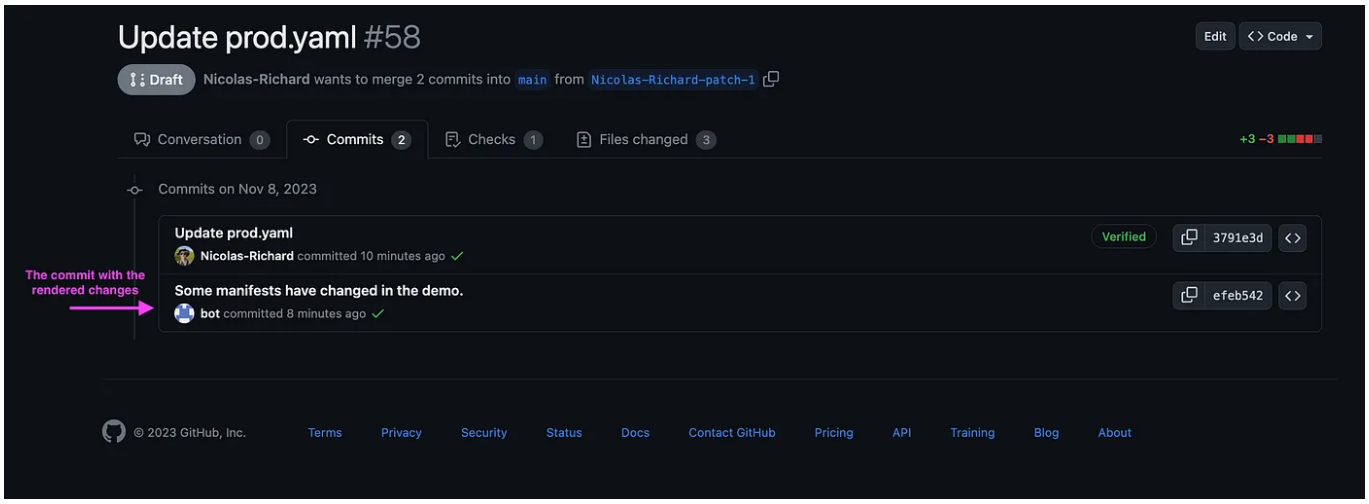
Chime is open-sourcing mani-diffy in the hope of helping the community of Argo users.

https://github.com/1debit/mani-diffy/

See it in action: Experience mani-diffy in a live demo

To try "mani-diffy" you simply need to submit a PR where you make a change to the overrides. You'll see the GitHub action add a commit to your PR with the resulting changes.

As you can see in these screenshots:



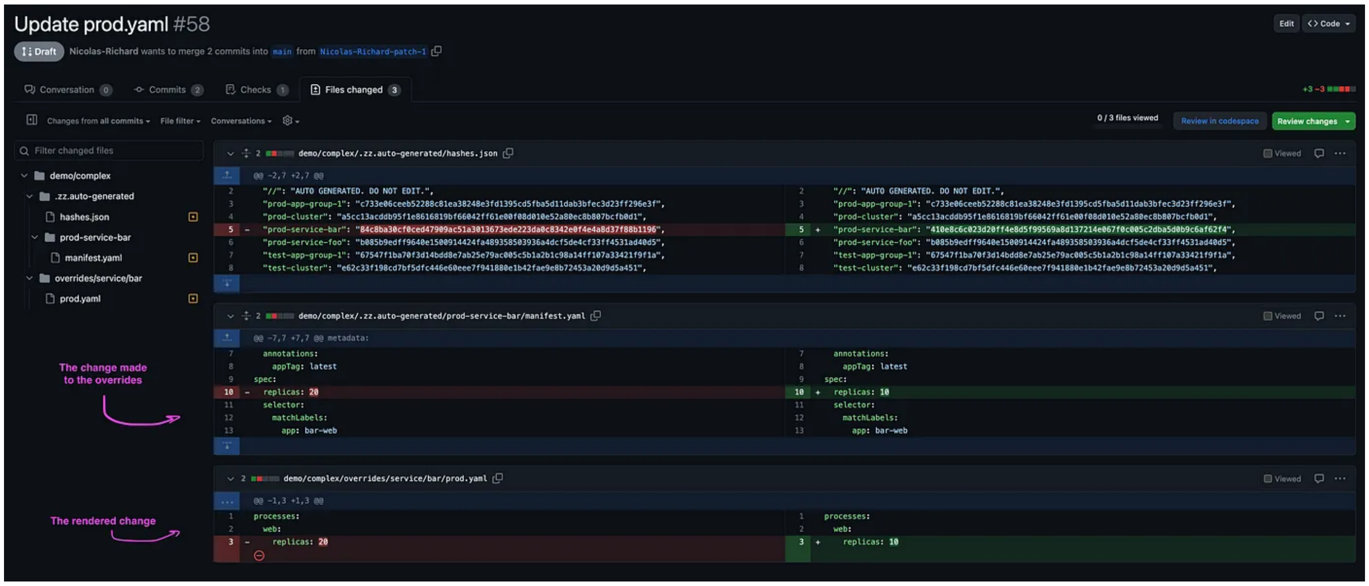
Update prod.yaml #58

Draft Nicolas-Richard wants to merge 2 commits into main from Nicolas-Richard-patch-1

Commits on Nov 8, 2023

Update prod.yaml
Nicolas-Richard committed 10 minutes ago — Verified — 3791e3d

Some manifests have changed in the demo.
bot committed 8 minutes ago — efeb542



Update prod.yaml #58

Nicolas-Richard wants to merge 2 commits into `main` from `Nicolas-Richard-patch-1`

2 demo/complex/.zz.auto-generated/hashes.json

```
@@ -2,7 +2,7 @@
   "//": "AUTO GENERATED. DO NOT EDIT.",
   "prod-app-group-1": "c733e06ceeb52288c81ea38248e3fd1395cd5fba5d11dab3bfec3d23ff296e3f",
   "prod-cluster": "a5cc13acddb95f1e8616819bf66042ff61e00f08d010e52a80ec8b807bcfb0d1",
-  "prod-service-bar": "84c8ba30cf0ced47909ac51a3013673ede223da0c8342e0f4e4a8d37f88b1196",
+  "prod-service-bar": "410e8c6c023d20ff4e8d5f99569a8d137214e067f0c005c2dba5d0b9c6af62f4",
   "prod-service-foo": "b085b9edff9640e1500914424fa489358503936a4dcf5de4cf33ff4531ad40d5",
   "test-app-group-1": "67547f1ba70f3d14bdd8e7ab25e79ac005c5b1a2b1c98a14ff107a33421f9f1a",
   "test-cluster": "e62c33f198cd7bf5dfc446e60eee7f941880e1b42fae9e8b72453a20d9d5a451",
```

2 demo/complex/.zz.auto-generated/prod-service-bar/manifest.yaml

```
@@ -7,7 +7,7 @@ metadata:
    annotations:
      appTag: latest
  spec:
-   replicas: 20
+   replicas: 10
    selector:
      matchLabels:
        app: bar-web
```

2 demo/complex/overrides/service/bar/prod.yaml

```
@@ -1,3 +1,3 @@
  processes:
    web:
-     replicas: 20
+     replicas: 10
```

Watch this video to see mani-diffy in action:

Conclusion

Accessible previews of Kubernetes changes directly in the pull requests make authoring and reviewing the deployments of our services an order of magnitude easier while also dramatically reducing the risk of causing an availability incident that would disrupt our members' financial peace of mind. 💚

Chime is proud to open-source mani-diffy and hopeful that it can help the community of Argo users.

Many thanks to Chime’s contributors to mani-diffy: Sean Marman, Eric Holmes, Andre Mercer, Andy Lindeman, Ethan Erchinger, Ben Whaley, Diego Alvarez, and Nicolas Richard.

A special shout-out goes to Sean Marman, who was the initiator of this project, for his significant contributions.

Engineering At Chime · Kubernetes · Kubernetes Cluster · Software Development