

Kubeflow Test-infra on AWS



Author: Jiaxin Shan(Jeffwan@) shjiaxin@amazon.com
Yao Xiao(PatrickXYS@) xyshow@amazon.com
Date: 09/21/2020

This doc describes the goal, the approach, and the architecture of AWS Kubeflow Test-infra. Doing so, we should be able to run Kubeflow tests against AWS test-infra.

Background

Currently, Kubeflow test and release infra is built with GCP services such as GKE, GCR, GCS, etc. This testing infra is managed by Googlers. In addition, few community members have access to the cluster, which means debugging upstream tests is not easy for Non-Googlers. [testing/issues/737](#) describes major issues to maintain test/release infrastructure.

At the beginning, AWS considered running e2e Kubeflow tests periodically for internal testing purposes because of increasing needs, AWS also needs more specific test cases the community doesn't cover. see [testing/issues/748](#). As the community has more discussion on long term testing solutions recently, AWS feels it would be great to provide and share test-infra with the community as an alternative provider. Thus, AWS spent some time on existing testing code refactor and made them work on AWS.

Goal

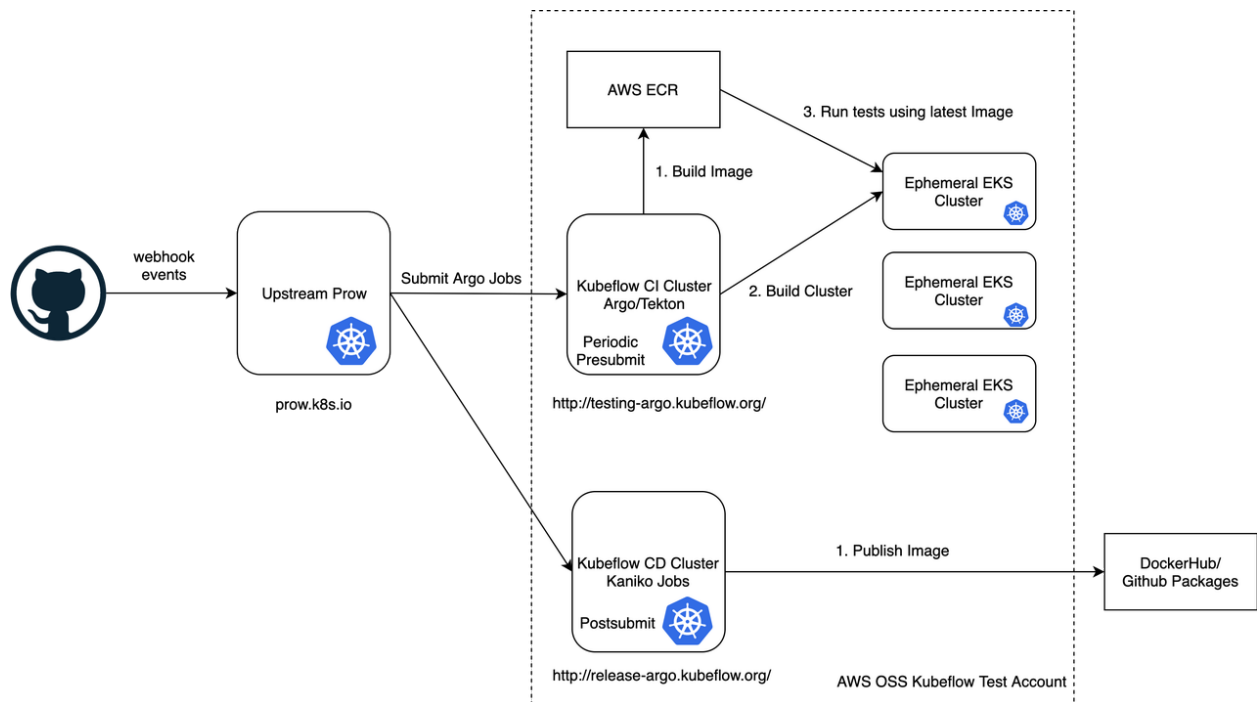
- Provides test infrastructure for Kubeflow community. WGs like Deployment, Training, Auto ML, Serving can choose to run their tests on by leveraging it.

- Rewrite or create separate e2e test workflows which can run on AWS.
- Come up with a scalable approach and share responsibility model to maintain the infrastructure.

High Level Infrastructure

In the short or mid term, the community probably can still make a deal with sig-testing to leverage existing prow to receive webhook events from Github just like what Kubeflow is doing now, see [here](#) for existing Kubeflow prow jobs. To move forward with this approach,

1. Apply credits for Kubeflow project and create a new AWS testing account for kubeflow project
2. Give credentials to sig-testing and ask them to create preset **preset-aws-credential-aws-kubeflow-testing: "true"**
3. The credentials will be used to **DescribeCluster** and authenticate ProwJob to submit Argo/Tekton workflow to AWS EKS CI cluster. Kubeflow ephemeral test clusters will be created in Argo Pipeline which is hosted in Kubeflow CI cluster in AWS account, privilege given to them is minimum.
4. Stop all existing jobs send to existing [Kubeflow](#) cluster

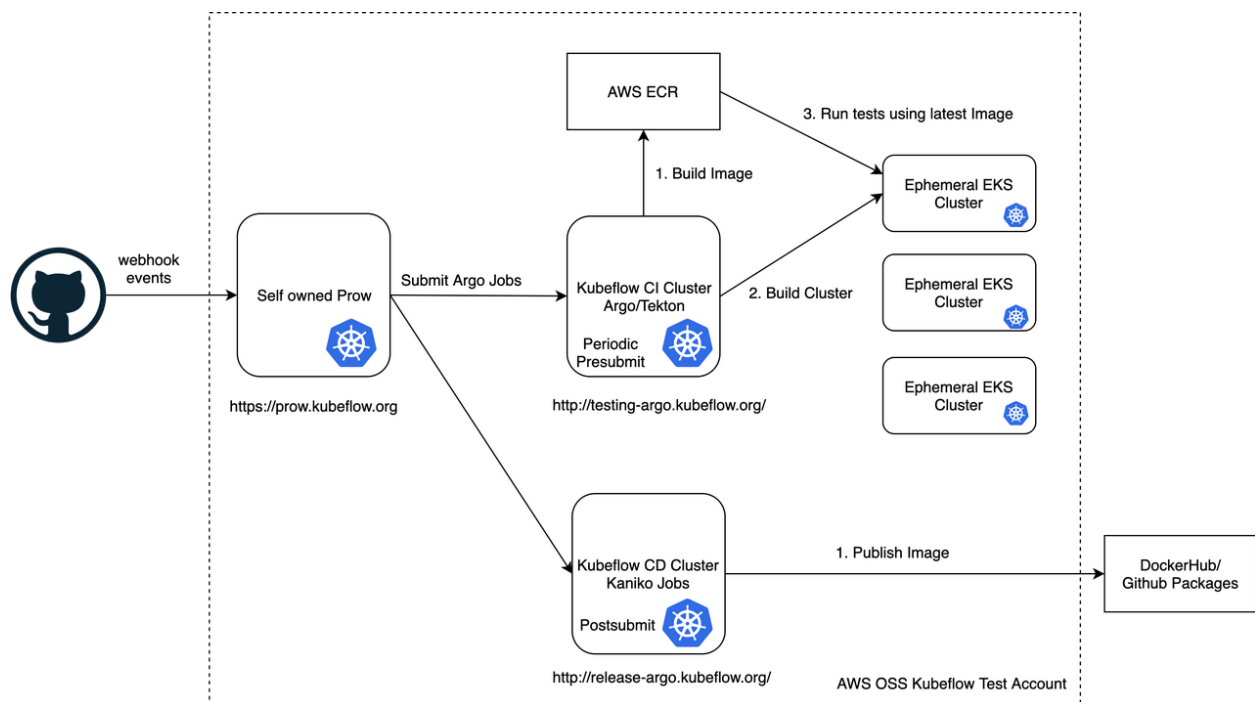


Notes:

1. The reason we propose to use Dockerhub or Github Packages is because ECR doesn't provide public container images. Once ECR adds public image feature, we could consider publishing images in ECR. (target for Q4 2020)
2. CI and CD will be hosted in two separate clusters for security reasons, they will use different container registries. Images built by CI cluster should only be used for testing and don't have to be exposed to the public.
3. Cloud native and open source solutions will be adopted rather AWS managed services. [Kaniko](#) will be used to build docker images and push to ECR in EKS. [Argo](#) and [Tekton](#) will be used to orchestrate testing steps and build test workflow. Kubeflow community is moving from Argo to Tekton for many reasons. See Appendix A for more details.
4. The reason Prow is still being used is because Kubeflow is an organization and prow is designed for this purpose. plugins like /lgtn /approve and integration with Github repo members makes it easy to manage all the projects in a fine grain way. Even though all these can be implemented using individual Github plugins, Prow is much easier.

In the long term, sig-testing would like to split CNCF projects and OSS projects into different prow clusters, see [issue](#). [GoogleCloudPlatform/oss-test-infra](#) will be used to host and serve GCP projects. The transition is going on, see [issue](#). If community will stick to current testing infra solution, then a separate Prow needs to be setup by community.

Here's a Kubeflow testing infra architecture on AWS.



In this case, the community needs to manage a separate prow. Engineer side, we need to do a few things

1. Create a separate EKS cluster in OSS Kubeflow test account for prow jobs. (it can be deployed in the same cluster as CI if there's no security concerns)
2. Create a robot account and add to Kubeflow projects as collaborators.
3. Update Kubeflow projects webhook endpoint to prow.kubeflow.org which points to Prow's public endpoint.

Security

This sections will go over all the permission needed by different components and potential security problems

Components	Policies	Comments
Prow	S3 credentials	Write artifacts to S3
	hmac token	Used by Github to validate webhooks
	github token	Access public repository and commit status
ProwJobs	aws credential	Describe kubeflow-ci cluster and submit workflow to that cluster
Argo	S3 credentials (optional)	Persist logs to S3
Pipeline:Kaniko	ECR policy	Upload docker images to existing ECR registry
Pipeline:create_cluster	eksctl policy	Create an EKS cluster
Pipeline:test_steps	aws credential	Deploy manifests to test cluster
CD pipeline	Dockerhub robot	Push images to remote Dockerhub

Sharing permissions

Kubeflow community needs a more sustainable and scalable approach to owning and maintaining testing infrastructure. Grant permission to some community members to help reduce maintenance efforts on AWS side and it's more scalable. While, security becomes critical and there will be a tradeoff.

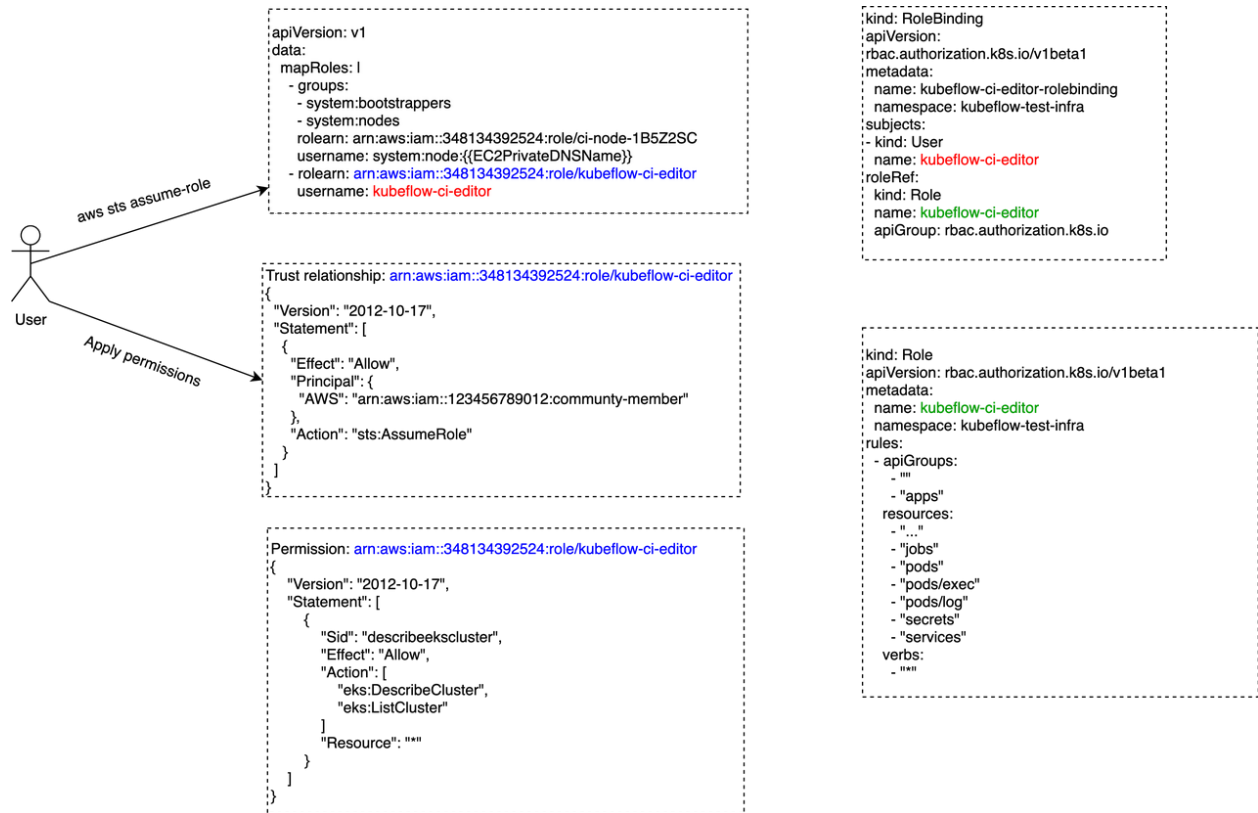
We will talk about the native AWS solutions to share permissions to external users.

EKS doesn't support OIDC on the server side and we can not easily bind an external email with EKS Kubernetes users. This feature will come in Q4 2020. Instead, we can leverage the IAM role or OIDC provider to manage external users authentication. For external users who needs access to EKS cluster, I think in general there are two options

1. Users need to assume a given role from AWS and get access to the EKS testing cluster.
2. EKS can configure OIDC providers and users can authenticate with their familiar IDP solutions like Google.

Option 1. Create an IAM role that can be assumed by some members

Roles	Rules	Kubeflow Users
admin	system:masters	AWS maintainers
editor	Prow/Argo data plane editor	WG leads and maintainers. Someone who work on testing stories.
viewer	Prow: test-pods Argo: kubeflow-test-infra	Active contributors



Option 2: EKS OIDC with Google authentication

Waiting for [feature](#) .

Maintenance efforts

Community needs a more sustainable and scalable approach to owning and maintaining test and release infrastructure for Kubeflow projects. One major problem is community lack a sufficient number of individuals with the willingness and ability to meet the increasing demands of Kubeflow as it grows. Two options are being discussion in issue [kubeflow/testing/issues/737](#)

1. Staff a horizontal team supporting test and release infra across Kubeflow
2. Make each WG responsible for their own test infra.

This still needs further discussion. From our point of view, I think we can do the following things to reduce maintenance overall efforts.

1. AWS will help update some test cases to show how to move the e2e test pipeline on EKS, [example](#).
2. Review the doc with community WG owners to help them understand shared responsibility.
3. AWS team members will help on quota issues, cluster upgrade or grant permissions to org members.
4. Provides persisted Prow and Argo Logs to the public to help debug test cases.
5. Use GitOps way to manage AWS resources like new ECR registry, EKS cluster template. AWS doesn't have that mature tools to manage AWS resources through Kubernetes but this is unnecessary.
 1. AWS offering <https://github.com/aws/aws-controllers-k8s>
 2. GCP offering <https://github.com/GoogleCloudPlatform/k8s-config-connector>

Milestones

Phase 1: POC

1. Setup test-infra in AWS OSS Kubeflow account
 1. Point prow.kubeflow.org and argo.kubeflow.org to AWS application load balancer
 2. Add Github Bot account [eks-bot](#) to repos
 3. Set up a Prow cluster used to receive Github events and kick off ProwJobs, then ProwJob will send requests to Argo Cluster to create an Argo based workflow test.
 4. Set up Argo cluster
2. Add support in kubeflow/testing to provide AWS as an alternative platform to run tests. PR [kubeflow/testing/pull/755](#)
3. WGs who like to run E2E on AWS can migrate test case
 1. Add bot as repo collaborator
 2. Add webhook to ship Github events to Kubeflow Prow hosted in AWS.
4. Apply credits for AWS accounts.

Phase 2: Cut over to AWS OSS Kubeflow account

1. Switch webhook to a different AWS account with approved credits
2. Write playbooks and infrastructure details to onboard community members.

Phase 3: Share clusters with community

1. Gradually open test-infra to Kubeflow users with limited permissions
2. Manage AWS resources in GitOps way to auto apply infrastructure changes by all community members.

Kubeflow Container Image Release CD

Option 1: PR's Postsubmit Job can use Kaniko to build and push container image to public registry. [Kubeflow CD](#) will utilize Tekton Pipeline to submit [PR](#) for updating new-pushed images in manifests.

Option 2: After PR merged, Kubeflow CD detects the branch is updated, then build & push image to public registry, cuts a PR to update new-pushed image in manifest.

Appendix A: Moving CI/CD pipeline from Argo to Tekton

The major benefits from Tekton is the pipeline can be split into reusable tasks. Community could have many common tasks for things like creating/deleting clusters, deploying Kubeflow and then combining these tasks into E2E tests.

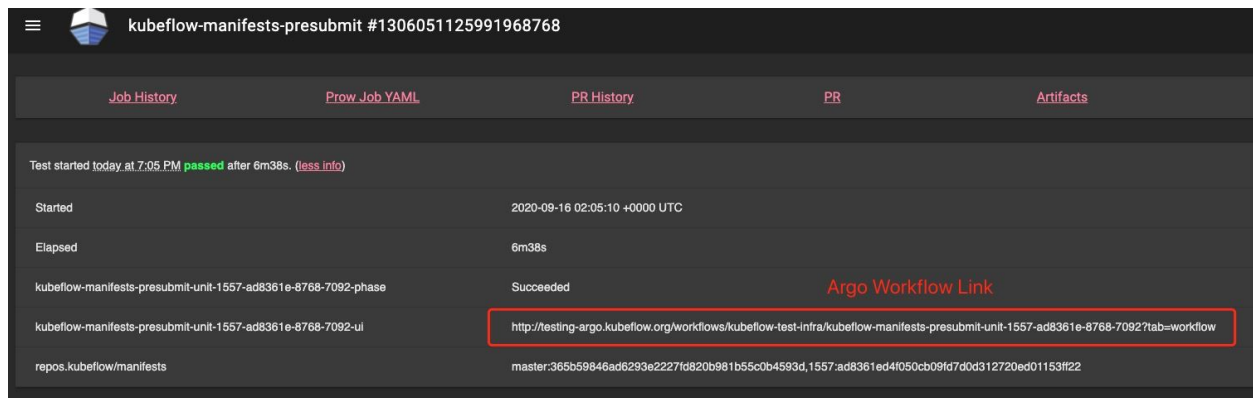
Appendix B: Kubeflow Projects which needs e2e

[kubeflow](#) has 42 repos now. Not all the repos have e2e tests. Here's a list of repos currently have e2e testing running.

Working Group	Repos	Comments
Control Plane	https://github.com/kubeflow/kubeflow	
	https://github.com/kubeflow/kfctl	
	https://github.com/kubeflow/manifests	
Training	https://github.com/kubeflow/tf-operator	
	https://github.com/kubeflow/pytorch-operator	
	https://github.com/kubeflow/mxnet-operator	
	https://github.com/kubeflow/xgboost-operator	
	https://github.com/kubeflow/fate-operator	
	https://github.com/kubeflow/common	

AutoML	https://github.com/kubeflow/katib	
Serving	https://github.com/kubeflow/kfserving	
Tools	https://github.com/kubeflow/fairing	
Examples	https://github.com/kubeflow/examples	
Tools	https://github.com/kubeflow/testing	

Appendix C: Prow skyglass setting in Kubeflow



The screenshot shows the Prow UI for a job named 'kubeflow-manifests-presubmit #1306051125991968768'. The job status is 'passed' after 6m38s. The 'Argo Workflow Link' is highlighted in red and points to <http://testing-argo.kubeflow.org/workflows/kubeflow-test-infra/kubeflow-manifests-presubmit-unit-1557-ad8361e-8768-7092?tab=workflow>.

Job History	Prow Job YAML	PR History	PR	Artifacts
Test started today at 7:05 PM passed after 6m38s. (less info)				
Started	2020-09-16 02:05:10 +0000 UTC			
Elapsed	6m38s			
kubeflow-manifests-presubmit-unit-1557-ad8361e-8768-7092-phase	Succeeded			
kubeflow-manifests-presubmit-unit-1557-ad8361e-8768-7092-ui	http://testing-argo.kubeflow.org/workflows/kubeflow-test-infra/kubeflow-manifests-presubmit-unit-1557-ad8361e-8768-7092?tab=workflow			
repos.kubeflow/manifests	master:365b59846ad6293e2227d820b981b55c0b4593d,1557:ad8361ed4f050cb09fd7d0d312720ed01153ff22			

Appendix D: Kubeflow E2E test cases examples. Kfctl unit test, Katib, KFServing

Argo workflow written by jsonnet:

https://github.com/kubeflow/kfctl/blob/master/testing/workflows/components/kubeflow_workflow.libsonnet

