

Arrikto

Enabling Multi-user Machine Learning Workflows for Kubeflow Pipelines

A story about auth

Yannis Zarkadas, Arrikto
Yuan Gong, Google Cloud



KubeCon



CloudNativeCon

Europe 2020

Virtual

Enabling Multi-user Machine Learning Workflows for Kubeflow Pipelines

ARRIKTO



Yannis Zarkadas
Software Engineer
Arrikto



Yuan Gong
Software Engineer
Google Cloud

What You'll Learn In This Session

How multiple users can work on Kubeflow Pipelines in a secure, isolated manner.

Why is this important?

- ✓ **Simplify** user onboarding with an intuitive UX
- ✓ **Accelerate** pipeline development by writing pipelines as python code
- ✓ **Collaborate** in a secure and isolated manner



Don't forget, you can grab the slides right now at arrik.to/kubeconAMS as well as enter the draw to win a fabulous prize



Get your questions answered live on Twitter and LinkedIn using the three hashtags [#kubecon](#) [#ml](#) [#arrikto](#)



The Kubeflow project is dedicated to making deployments of machine learning (ML) workflows on Kubernetes: simple, portable and scalable.

Platform for Machine Learning Pipelines:

- UI
- Python SDK to define Pipelines
- Visualizations
- Lineage Tracking
- ...

The screenshot displays the Kubeflow Pipelines interface. On the left is a dark blue sidebar with navigation options: Home, Notebooks, Volumes, Snapshots, Pipelines, Experiments (with sub-items: Pipelines, HP Tuning, Runs/Trials), Pipelines (highlighted), Artifacts, and Steps/Executions. At the bottom of the sidebar are links for GitHub and Documentation, and footer text: Privacy • Usage Reporting, build version v1beta1.

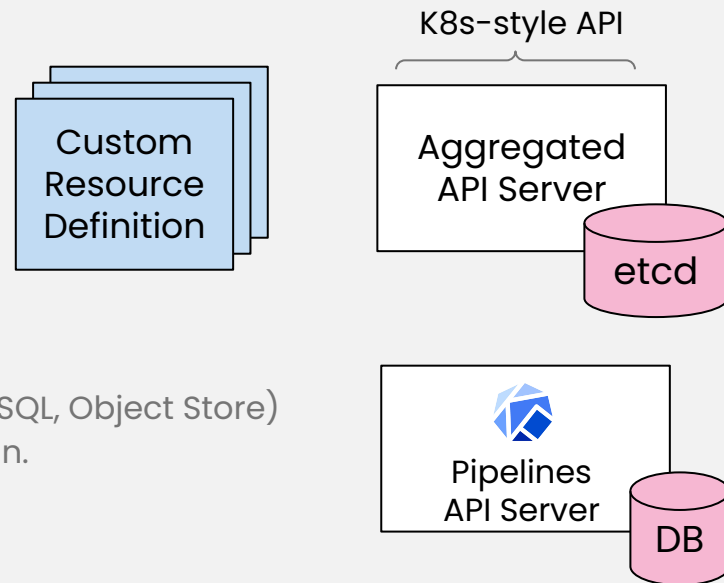
The main content area shows the user 'kubeflow-user' and the experiment 'Titanic'. The selected pipeline is 'titanic-ml-wv1k0_run-zspaj'. The 'Graph' tab is active, showing a 'Simplify Graph' button. The runtime execution graph consists of the following steps, all marked with a green checkmark to indicate completion:

- create-volume-2
- create-volume-1
- Loaddata
- Datapreprocessing
- Featureengineering
- Svm
- Naivebayes
- Logisticregression
- Randomforest
- Decisiontree
- Results

Arrows indicate the flow of data between steps. A legend at the bottom states: 'Runtime execution graph. Only steps that are currently running or have already completed are shown.'

The Problem

- All Pipeline Runs happen in the same namespace
 - No way to isolate secrets
 - No way to isolate data
- Kubeflow Pipelines is NOT a K8s-native API
 - Does NOT use CRDs or Aggregated API-Server
- Kubeflow Pipelines has its own API Server and Persistence (MySQL, Object Store)
 - No isolation/authentication/authorization in initial design.
 - How do we extend it?

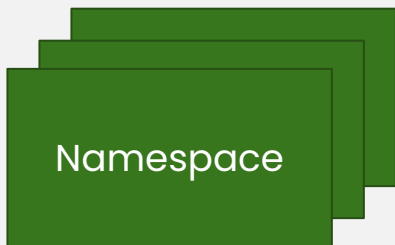


Three takeaways:

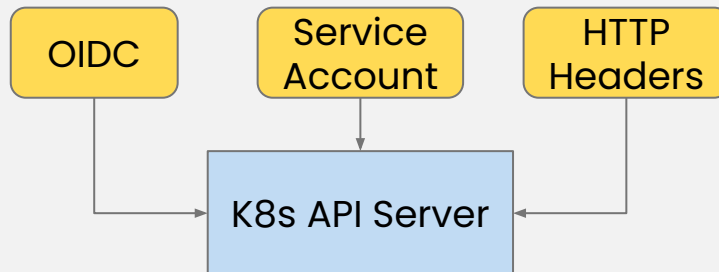
1. **Isolation** using namespaces
2. **Authentication** with multiple options (OIDC, ServiceAccount, ...)
3. **Authorization** with Role Based Access Control (RBAC)

VERB /apis/GROUP/VERSION/namespaces/**NAMESPACE**/**RESOURCETYPE**/**NAME**

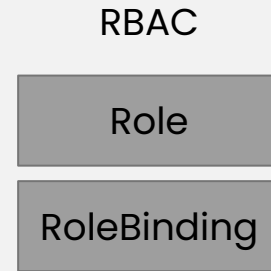
Isolation



Authentication



Authorization



Kubernetes APIs:

- **Built-in Resources:** Pod, Deployment, PersistentVolume, etc.
- **Custom Resource Definitions:** Jupyter Notebook, Argo Workflow, etc.

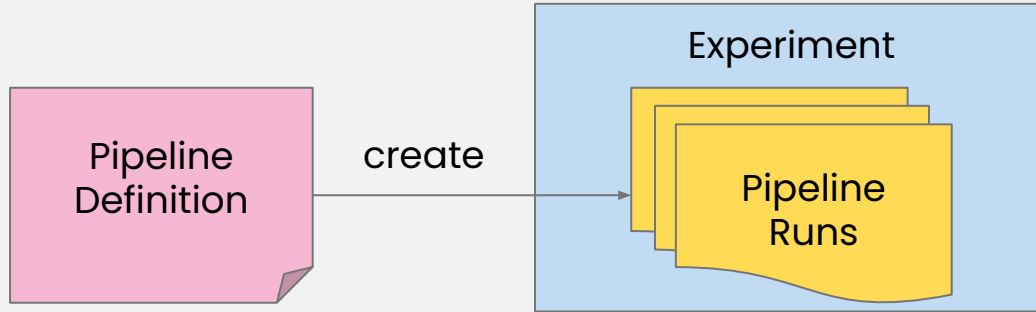
VERB /apis/GROUP/VERSION/namespaces/**NAMESPACE**/**RESOURCE**/**NAME**
ACTION on **RESOURCE** in **NAMESPACE**

GET /apis/apps/v1/namespaces/**istio-system**/**deployments**/**istiod**

Can **user sarah** **GET deployment istiod** in namespace **istio-system**?

POST /apis/kubeflow.org/v1/namespaces/**kubeflow-user**/**notebooks**/

Can **serviceaccount user-jupyter-notebook** **CREATE notebook** in namespace **kubeflow-user**?



Pipeline APIs have no isolation primitive:

- Add namespace to experiments
- Runs find their namespace from the experiment they are in

BEFORE:

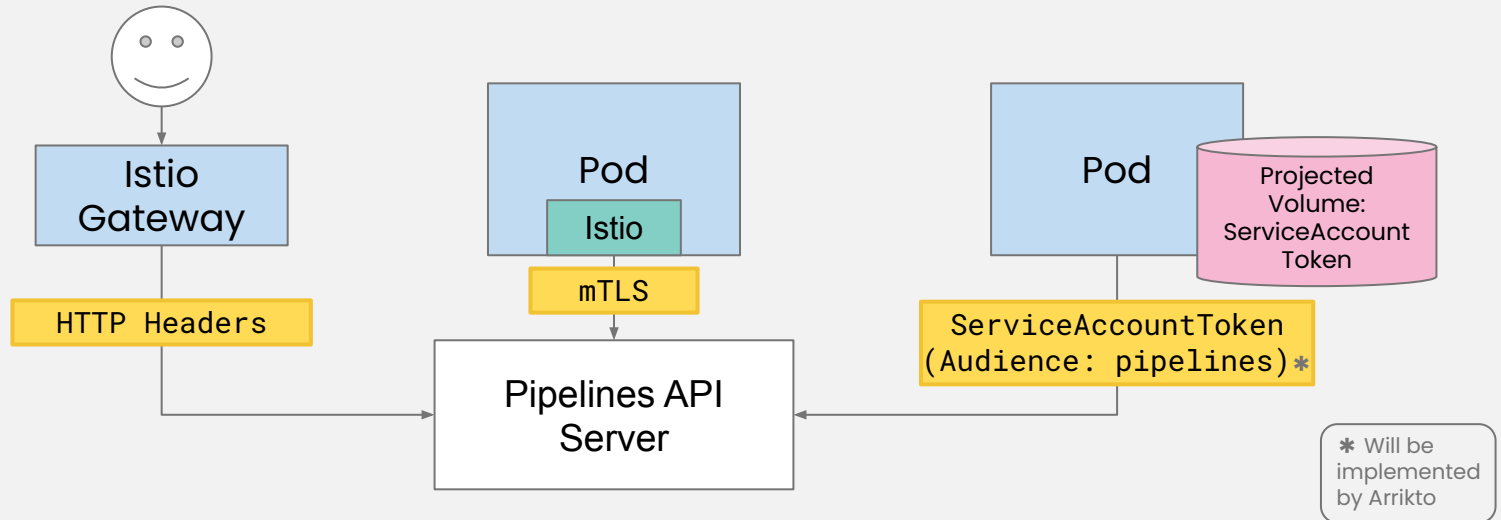
`/apis/v1beta1/experiments/{experiment_id}`


AFTER:


`/apis/v1beta1/experiments/{experiment_id}?resource_reference_key.type=NAMESPACE&resource_reference_key.id=ns1`


Auth for Kubeflow Pipelines – Authentication

- HTTP Headers
 - Kubeflow comes with a trusted Istio Ingress-Gateway proxy
- Istio mTLS
 - Difficult to use Argo with sidecar
- ServiceAccountTokens
 - TokenRequest/TokenReview API
 - WARNING! Use custom audience specifically for Pipelines





Joe Beda 
@jbeda

[Follow](#) 

Replying to [@wallyqs](#) [@pires_oss](#)

This is a horrible horrible idea. You should never hand a k8s service account token to anything except the API server.

[@mikedanese](#) is working to add APIs so that you can ask for a jwt with a different aud field. But before that is ready don't do this.

Auth for Kubeflow Pipelines – Authorization

Can USER do ACTION on RESOURCE in NAMESPACE?

Kubeflow Access Management Service (KFAM)

- View, Edit, Admin ACL per namespace
- Awkward way of assigning permissions
- Coarse-grained abstraction on top of K8s RBAC
- Deprecated

Pipelines API Server Endpoints	KFAM
	Namespace ACL
GET /apis/v1beta1/runs/{id}	view
GET /apis/v1beta1/runs?querystring	view
POST /apis/v1beta1/runs	edit
DELETE /apis/v1beta1/runs/{id}	admin
GET /apis/v1beta1/pipelines/{id}	view

Auth for Kubeflow Pipelines – Authorization

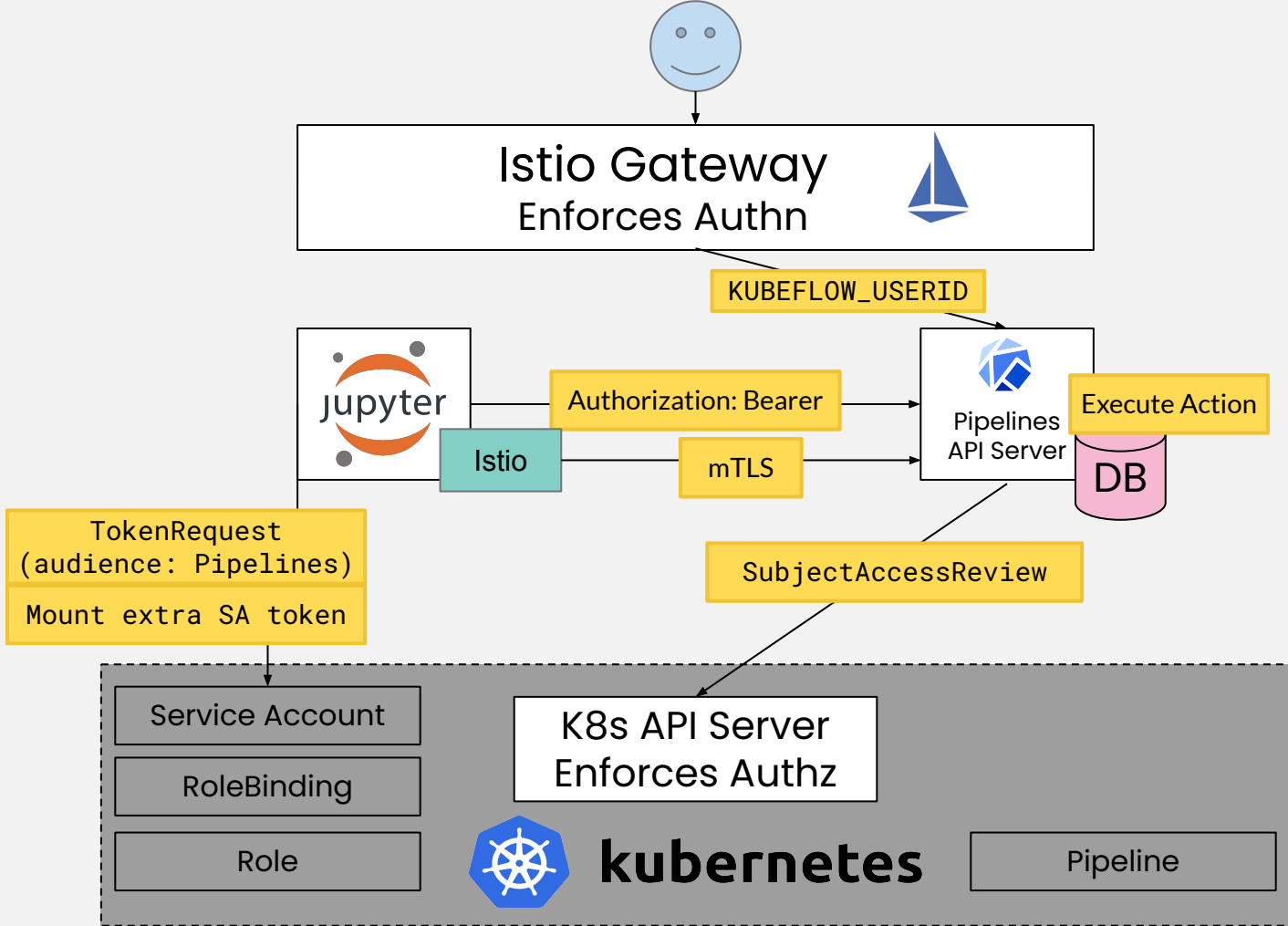
Can USER do ACTION on RESOURCE in NAMESPACE?

Role Based Access Control (RBAC)

- Map each API endpoint to an RBAC permission
- Use SubjectAccessReview to make authorization decisions
- Use standard Roles/RoleBindings for assigning permissions

Pipelines API Server Endpoints	RBAC	
	Resources	Verbs
GET /apis/v1beta1/runs/{id}	Runs	GET
GET /apis/v1beta1/runs?querystring	Runs	LIST
POST /apis/v1beta1/runs	Runs	CREATE
DELETE /apis/v1beta1/runs/{id}	Runs	DELETE
GET /apis/v1beta1/pipelines/{id}	Experiments	GET

* Will be implemented by Arrikto



Securing in-cluster traffic



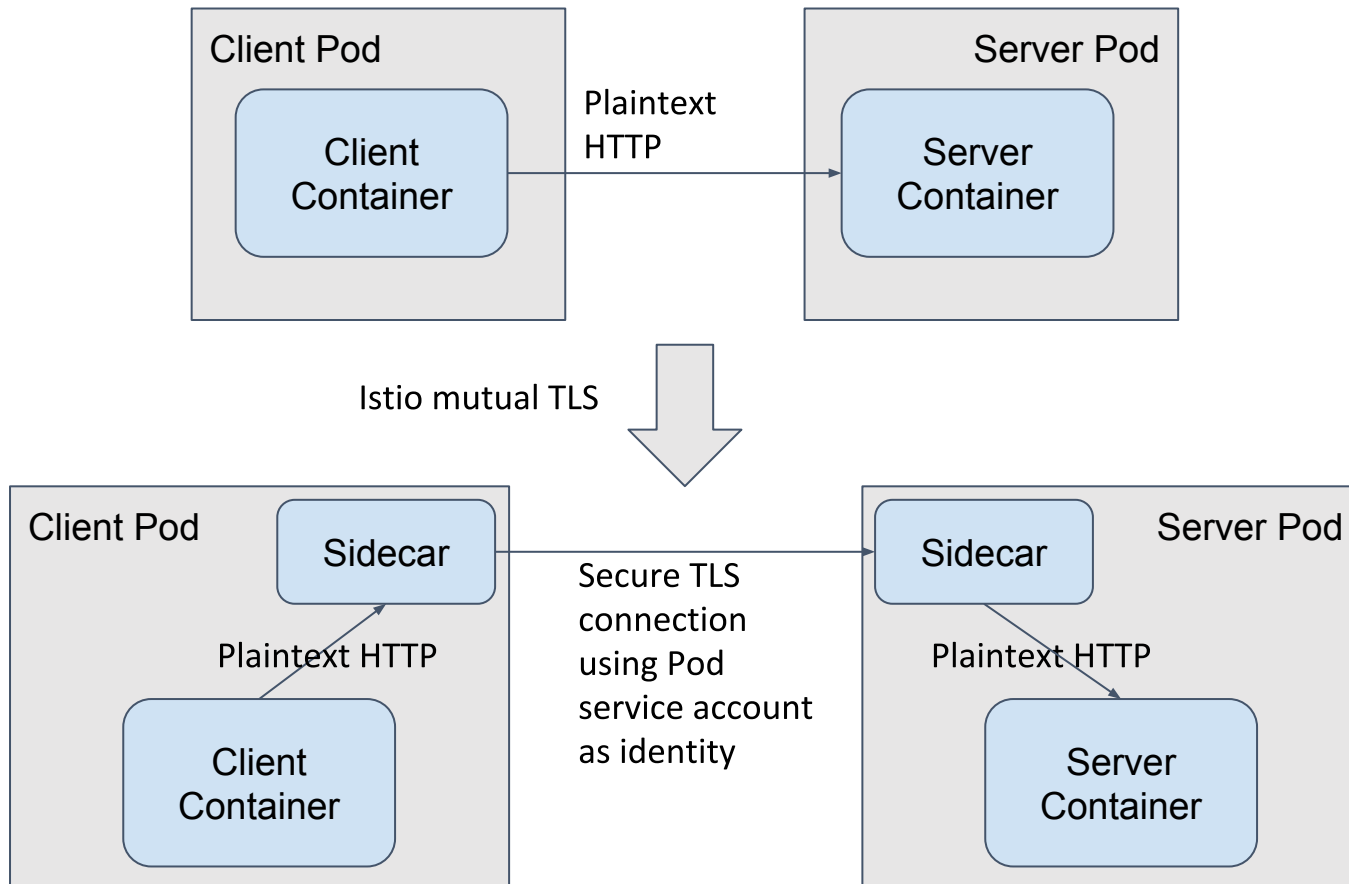
KubeCon



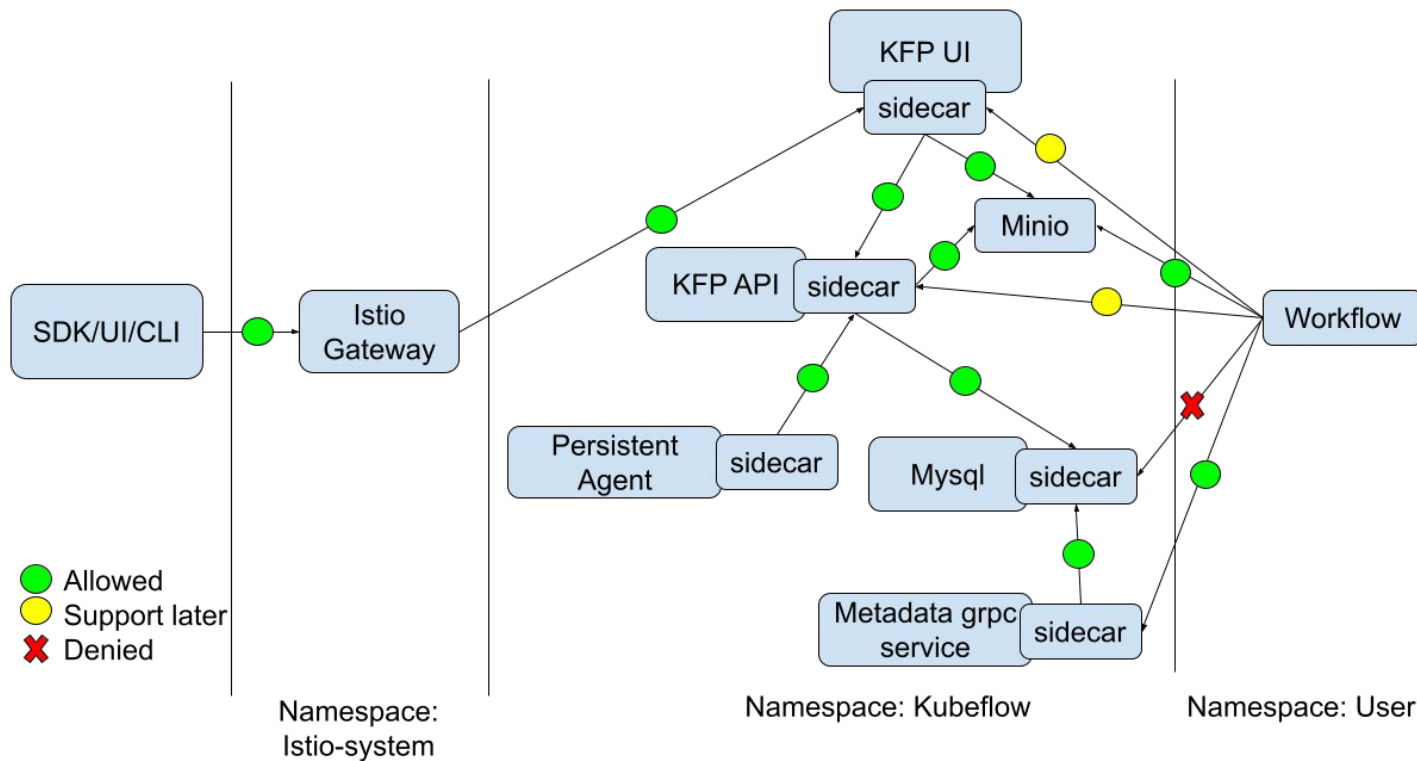
CloudNativeCon

Europe 2020

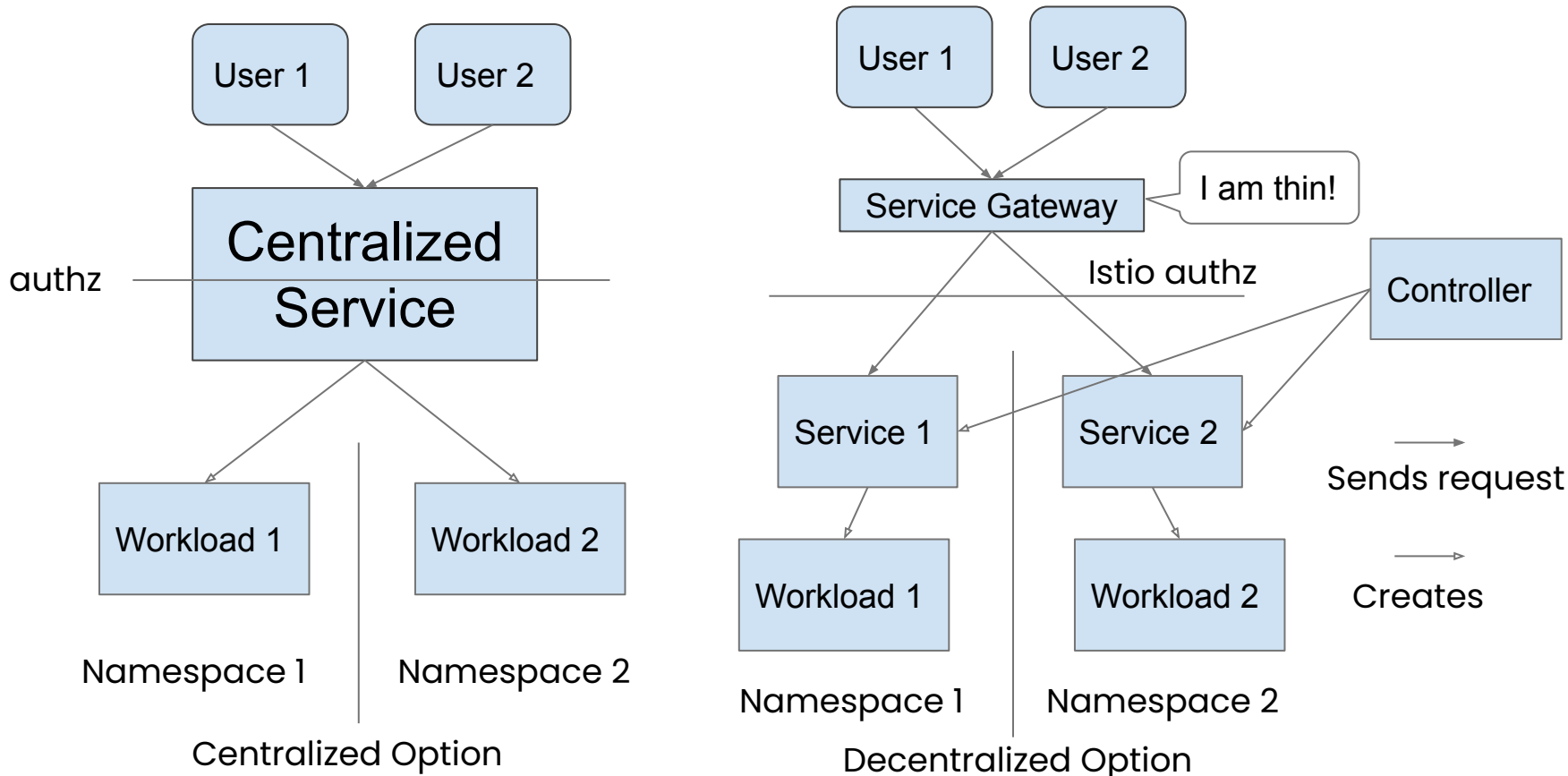
Virtual



Securing in-cluster traffic



Multi user support for KFP



Centralized API Server



KubeCon



CloudNativeCon

Europe 2020

Virtual

Pros

- Lower operational and computational cost
- Possibility of building cross-namespace features like sharing

Cons

- A lot of code changes throughout all APIs

Artifacts



KubeCon



CloudNativeCon

Europe 2020

Virtual

Run output Config

Analyzer ✓

Transformer ✓

Trainer ✓

Predictor ✓

ROC curve ✓

Confusion matrix ✓

dataproc_delete_cl... ✓

xgboost-trainer-w

Input/Output **Visualizations** ML Metadata Volumes Logs Pod

ROC Curve

tpr

fpr

Visualization Creator

[create visualizations manually](#)

Decentralized UI artifact servers



KubeCon



CloudNativeCon

Europe 2020

Virtual

Pros

- Minimal code changes
- Flexible to customize, e.g. mount volumes in user namespaces
- No operational cost increase

Cons

- Limitedly higher computational cost

Design - what is missing?



KubeCon



CloudNativeCon

Europe 2020

Virtual

- How do we integrate with other two authentication methods securely?
- Pipeline definitions are shared -- Code is shared.
- How do we isolate Object Store?
- How do we isolate Machine Learning Metadata DB (MLMD)?



The istio lessons I learned,
the HARD way.

* Using Istio 1.1
some content may be outdated

Gradual migration path enabling istio sidecar auto injection in a shared namespace.

A few ways to configure auto injection

- Default injection policy in the istio-sidecar-injector configmap in istio-system namespace (cluster scoped)
- “istio-injection: enabled” namespace label (required, but not default)
- `sidecar.istio.io/inject: true/false` pod annotation

Workaround: I had to add pod annotations to every pod in the shared namespace.

Implementation - istio



KubeCon



CloudNativeCon

Europe 2020

Virtual

mTLS is required for namespace/service account in authorization rules

Implementation - istio



KubeCon



CloudNativeCon

Europe 2020

Virtual

Information needed by authorization rules need to be populated first

- Service port name should start with its protocol, e.g. http for http payload related information



Takeaways as an istio learner

- Do not skip the basics
- Most authorization problems I met can be solved with just the “ensuring proxies enforced policies correctly” section in the huge “Common Problems / Security Problems” doc

Implementation - metacontroller



KubeCon



CloudNativeCon

Europe 2020

Virtual

- The entire KFP controller fits into one single 281-line python file.
- Mounted as a configmap on a python image.

This means anyone can easily come, read and customize KFP controller code to accommodate for their special requirements on namespace setup.



This couldn't happen without these amazing people:

- Ning Gao, previously at Google Cloud
- Chen Sun, Google Cloud
- Yang Pan, Google Cloud (Design only)
- Yannis Zarkadas, Arrikto (Design only)

Demo