

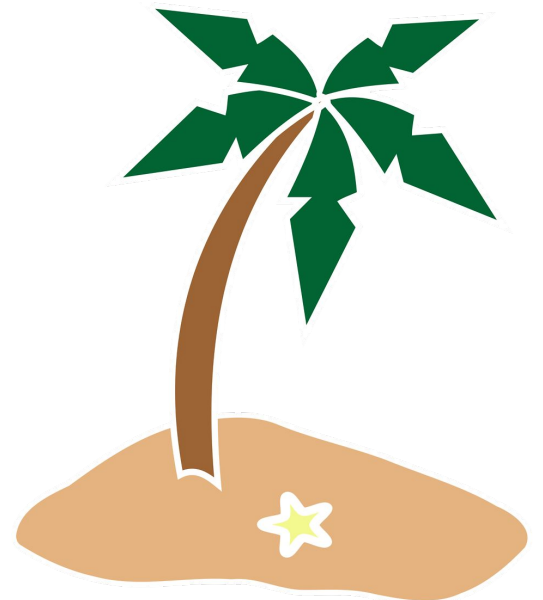
SIG-Service Catalog Introduction

Jonathan Berkahn - jaberkha@us.ibm.com - [@jberkhahn](https://twitter.com/jberkhahn)



Applications are rarely islands

- Often applications leverage ancillary "Services"
 - E.g. Application stores data in database
- Critical to application's success
 - But developers shouldn't spend their time managing them



Services - an overloaded term

- **Kubernetes “Services”**
 - Applications running in the cluster accessible via DNS discovery
- **Platform managed/hosted Services**
 - e.g. Object Storage
- **External Services - 3rd Party Services**
 - e.g. Twillio



Access to services can be challenging



- Creating and managing services is non-trivial
 - Duplication of effort across teams
 - Ops team manages it for you on their schedule
 - Managing credentials could be problematic
 - Sent via email, sticky-notes, etc...
 - Where are they stored? Plain text in config files?
 - Each service has its own set of provisioning APIs
- Let's shift the burden to the Platform via self-service model
 - "Tell us what you need and we'll manage it for you"
 - Service Credentials are protected and provided at runtime

What if ...?



```
$ svcat marketplace
```

CLASS	PLANS	DESCRIPTION
+-----+-----+-----+		
mysql	free	Simple SQL
	basic	
	enterprise	
mongodb	free	No-SQL DB

```
$ svcat provision myDB --class mysql --plan free
```

```
$ svcat bind myDB
```

Credentials (and connection info) in “myDB” secret

The magic



Cluster Admin:

- **Service Brokers** are registered with Kubernetes
 - Each Broker manages one or more **Services**
 - Each Service offers a set of variant-QoSs/**Plans**
- Services are available via a “**Marketplace**” in Kubernetes

\$ `svcat marketplace`

Developer:

- Chooses a **Service** from the **Marketplace**
- Kubernetes talks to owning **Broker** to provision it and obtain the credentials
- **Secret** (credentials, connection info) is available to the app

\$ `svcat provision myDB...`

\$ `svcat bind myDB`

Making it all possible



- API between Kubernetes (or CF) and a Service Broker
 - get list of services / provision / deprovision / bind / unbind
- Abstracts the Service Lifecycle APIs
- Service Brokers
 - Manage all aspects of Service's lifecycle
 - User Initiated: Create, Delete, Provide Credentials
 - Automatic: Auto-Scale, Backup, Recovery, QoS, ...
 - Hosted anywhere – in or out of the Platform
 - Application is usually unaware

Why?



- Application Developers
 - Can focus on their business logic
 - Services managed by the experts
 - Self-service model **speeds up** CI/CD timelines
 - Platforms can do more for you - e.g. sharing of services across clusters & platforms
- Service Providers
 - Low barrier or entry for new Service Providers
 - **Interop**: easily integrated into environments that supports the API
 - Kube, CloudFoundry, custom platforms (e.g. IBM Cloud, SAP)
 - With ease of access to services, an increase in their usage (\$)

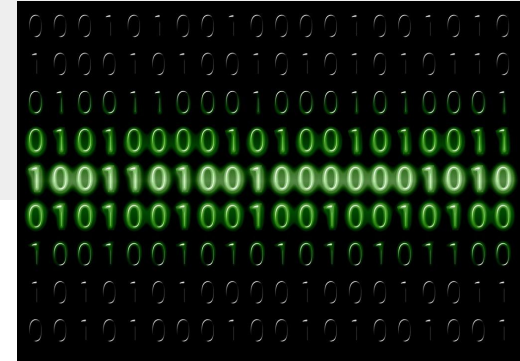
Demo

YAML all the things

```
apiVersion: servicecatalog.k8s.io/v1beta1
kind: ServiceInstance
metadata:
  name: myDB
spec:
  serviceClassName: mysql
  planName: free
```

```
apiVersion: servicecatalog.k8s.io/v1beta1
kind: ServiceBinding
metadata:
  name: myDB
spec:
  instanceRef:
    name: myDB
```

Credentials and connection info in “myDB” secret



Service Catalog Summary



Why?

- Help developers discover and connect to 3rd party services
- Allowing them to focus on their business logic
 - Ask for the service - connection information provided at runtime

Status

- Kubernetes incubator project
- Can be deployed into any Kubernetes cluster via a Helm chart
- Beta

One last thing about Services

- A service can be just about anything
- Data & Analytics – e.g. DBs, ElasticSearch
- Integration – e.g. Box, Twitter, SendGrid
- Utilities – e.g. conversions, speech to text
- Infrastructure – networks, volumes, routing
- DevOps – monitoring, metrics, auto-scaling

Questions



More information:

- <https://svc-cat.io>
- <https://github.com/kubernetes-incubator/service-catalog>
- <https://www.openservicebrokerapi.org/>
- Deep Dive session: Wednesday, May 22nd, 14:00 - 14:35 (Hall 8.1 G3)
- If you're interested in contributing, we'll be hosting weekly SIG meetings at 1 PM PST or 7 AM PST on the first Monday of every month @ <https://zoom.us/j/7201225346>

SIG-Service Catalog Deep-Dive

Jonathan Berkahn - jaberkha@us.ibm.com - [@jberkhahn](https://twitter.com/jberkhahn)



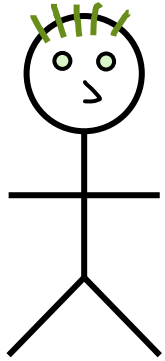


- API for the automated deployment, management, and use of services
 - Cloud-native apps require resources such as stable storage, etc
 - App developers shouldn't have to care about how the service is managed
 - OSB API abstracts all of this away
- Client side implemented by Service Catalog
 - managed through custom resource types
- Server side implemented by service provider as a 'broker'
 - get catalog endpoint
 - provision service endpoint
 - bind service endpoint

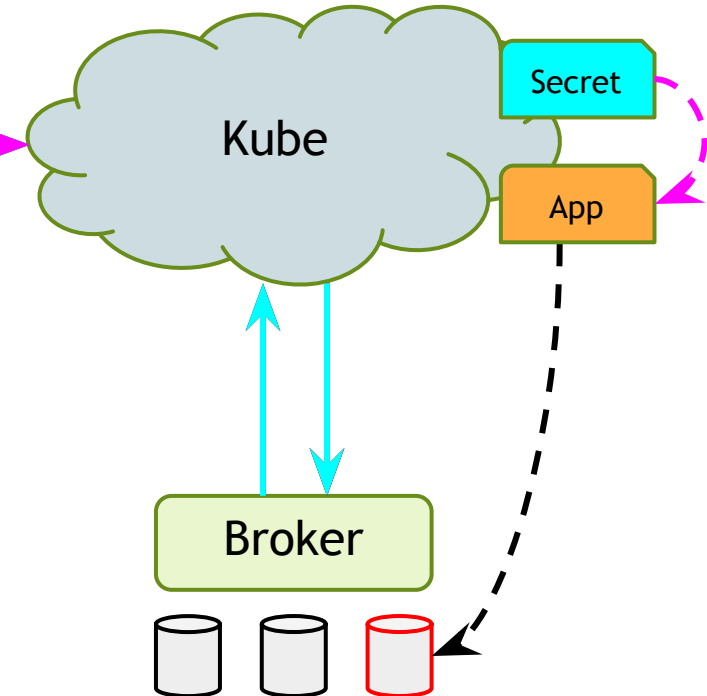
Service Catalog Resource Types

- **ClusterServiceBroker** (and namespaced Servicebroker)
 - A server running somewhere that offers various services, e.g. MySQL Broker
- **ClusterServiceClass** (and namespaced ServiceClass)
 - A category of services offered by a Broker, e.g. MySQL Databases
- **ClusterServicePlan** (and namespaced ServicePlan)
 - A specific type of a Service that a Broker offers, e.g. 100 MB MySQL Databases
- **ServiceInstance**
 - A single instantiation of a Service/Plan, e.g. Jonathan's 100 MB MySQL Database
- **ServiceBinding**
 - A unique set of creds to access a specific Instance, e.g. username/password for Jonathan's 100 MB MySQL Database

The Magic



1. Register Service Broker
2. Retrieve the Catalog of Services
3. Create a new Service Instance
 - Platform asks Brokers for Instance
4. Deploy Application
5. Bind Instance to an Application
 - Platform asks for new Binding/Creds
6. Access Service from Application
 - Using Creds from Binding Secret



Design Issues - API Aggregation

- CRDs didn't exist yet, TPRs were buggy
- Didn't want Service Catalog to have access to the main etcd in Kube for security reasons
- Solution: implement our own apiserver, use API aggregation to hook it in
- Allows normal interaction, i.e. ``kubectl create -f serviceinstance.yml``

Design Issues - GUIDs vs. Names

- Kube names are fixed
- OSB API resources have mutable names, and fixed GUIDs
- Service Catalog types use OSB API GUID as the name, and have a mutable ExternalName field
- svcat cli tool alleviates this pain by referencing human-readable ExternalNames as much as possible

```
metadata:
```

```
  name: 12kbac-adad12kbasd // from the broker; immutable
```

```
  uid: affd6f9b-defe-11e8-87bb-0242ac110007 // generated by Kube
```

```
spec:
```

```
  externalName: mysql // from the broker; can change
```

```
  externalID: 12kbac-adad12kbasd // same as metadata.name
```

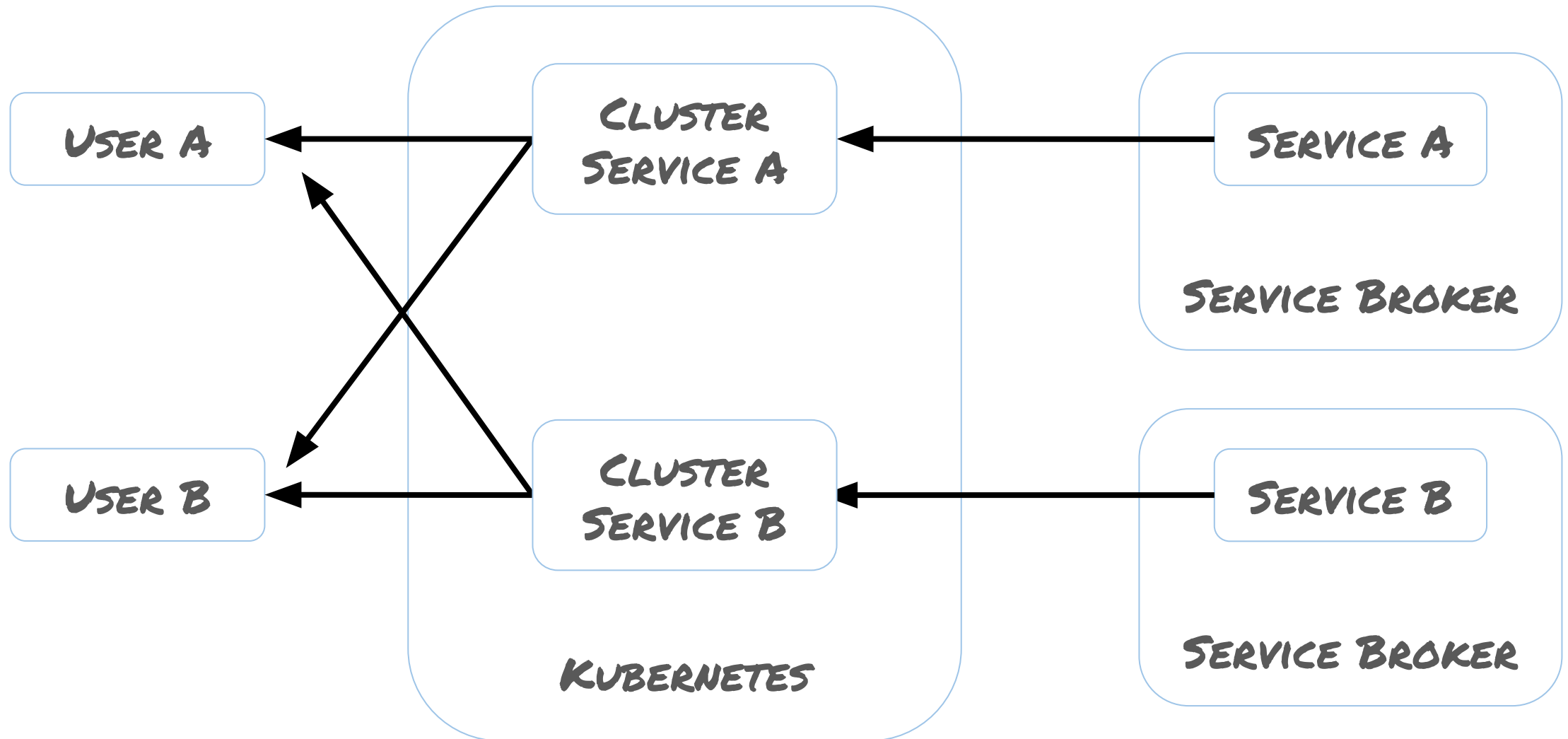
Design Issues - Broker Synchronization

- Kube isn't the sole source of truth
- Declarative control flow allows users to manipulate Service Catalog resources at-will
- Broker can still reject these changes
- Ongoing work to fix these sync issues

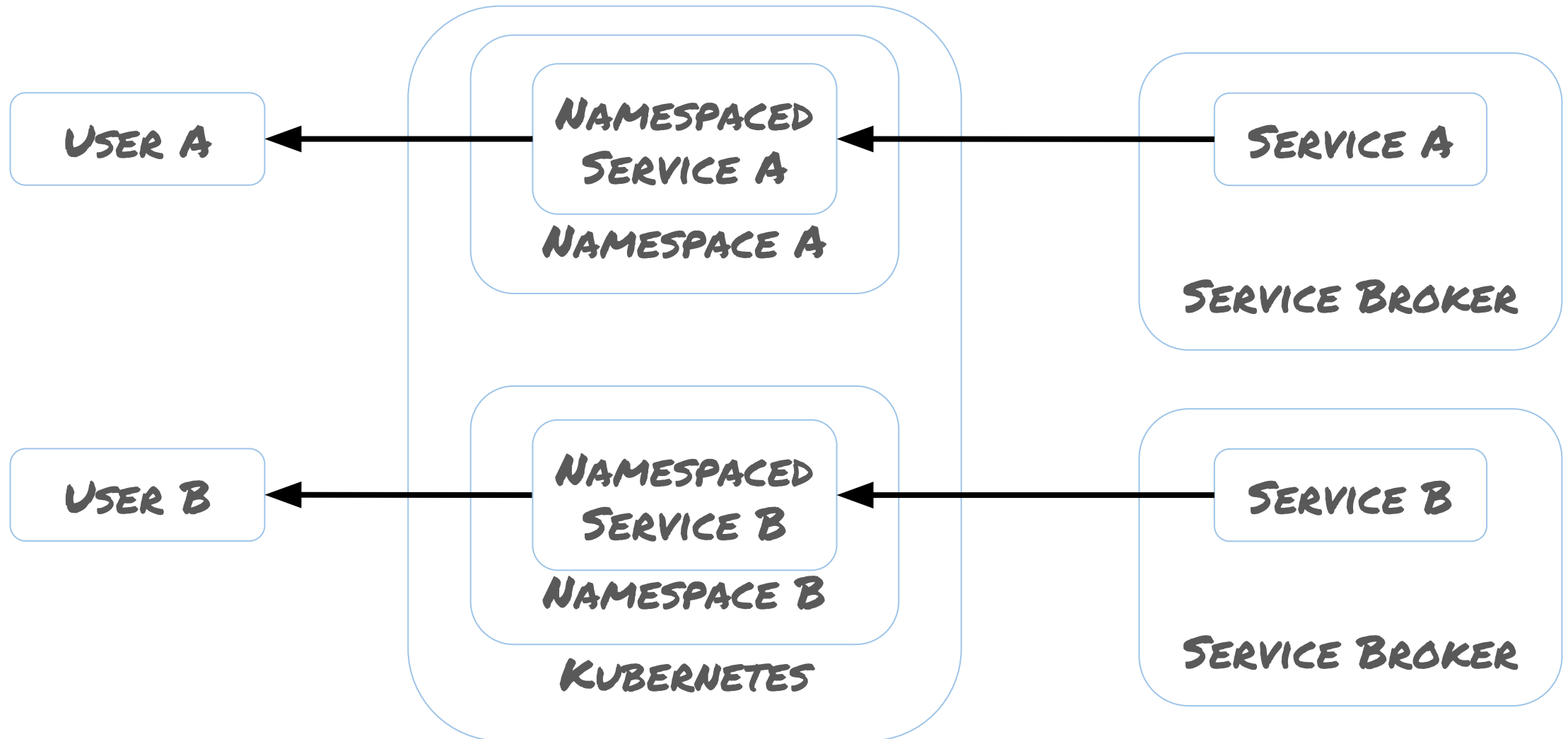
Release 0.2.0

- Official release of namespaced resources
 - Allows operators to make specific services available to specific users
 - Allow developers to manage their own brokers
 - svcat updated to work with both cluster and namespaced resources
- Allow Kube operators and users to grant selective access to service brokers/classes/plans
 - Namespaced brokers
 - Catalog Restrictions

Cluster Service Brokers



Namespaced Service Brokers



CRDs

- Transitioning from an API Server to CRDs
- Adding a mutating webhook server to replicate minor functionality from our old API Server
- Still a work-in-progress

Future Plans

- CRDs
- Improve synchronization between Kube and brokers
- User-Provided-Services to allow use of legacy services with service catalog
- Pod presets
- Docs improvements
- Coming up on 1.0.0

Questions

More information:

- <https://svc-cat.io>
- <https://github.com/kubernetes-incubator/service-catalog>
- <https://www.openservicebrokerapi.org/>
- If you're interested in contributing, we'll be hosting weekly SIG meetings at 1 PM PST or 7 AM PST on the first Monday of every month @ <https://zoom.us/j/7201225346> and agenda doc [here](#)