Kubernetes Operators in Java

What are Custom Resource Definitions and how to use them

Donato Wolfisberg

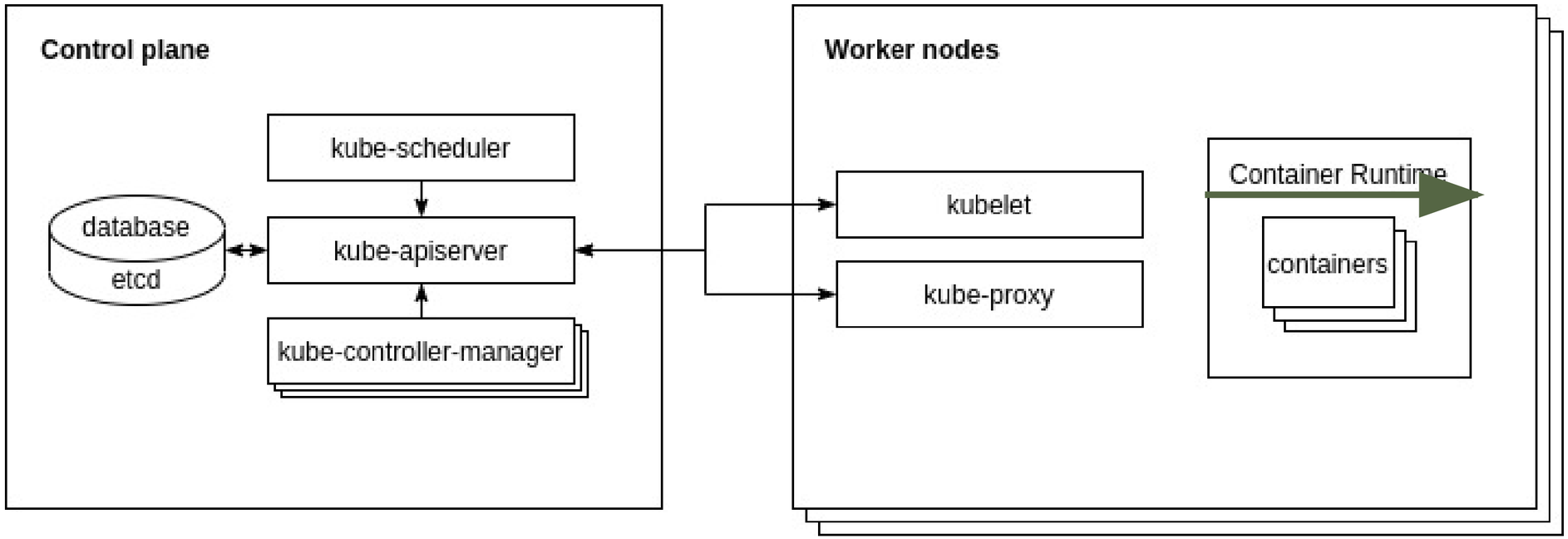
SirCremefresh wo_donato

Inhalt

- Kubernetes
- Resourcen
- Use Case
- Custom Resource
- Custom Controller
- Implementation
- Demo

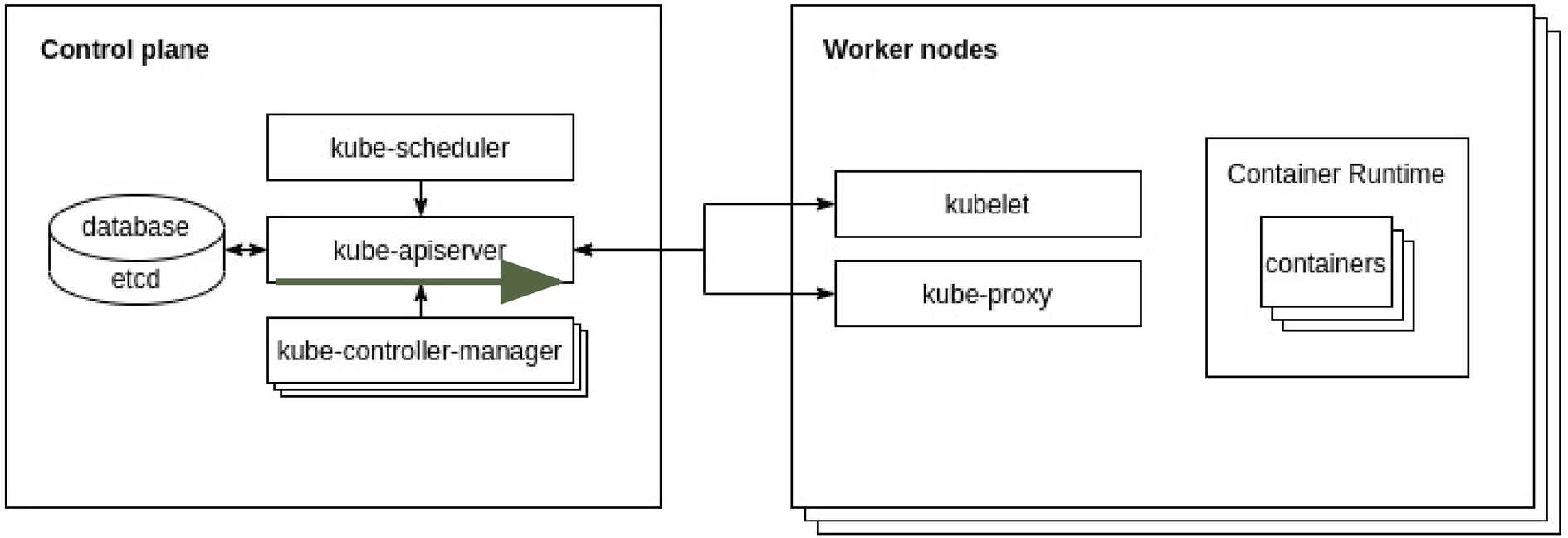
Kubernetes

System für deklaratives verwalten von Containern und Resourcen



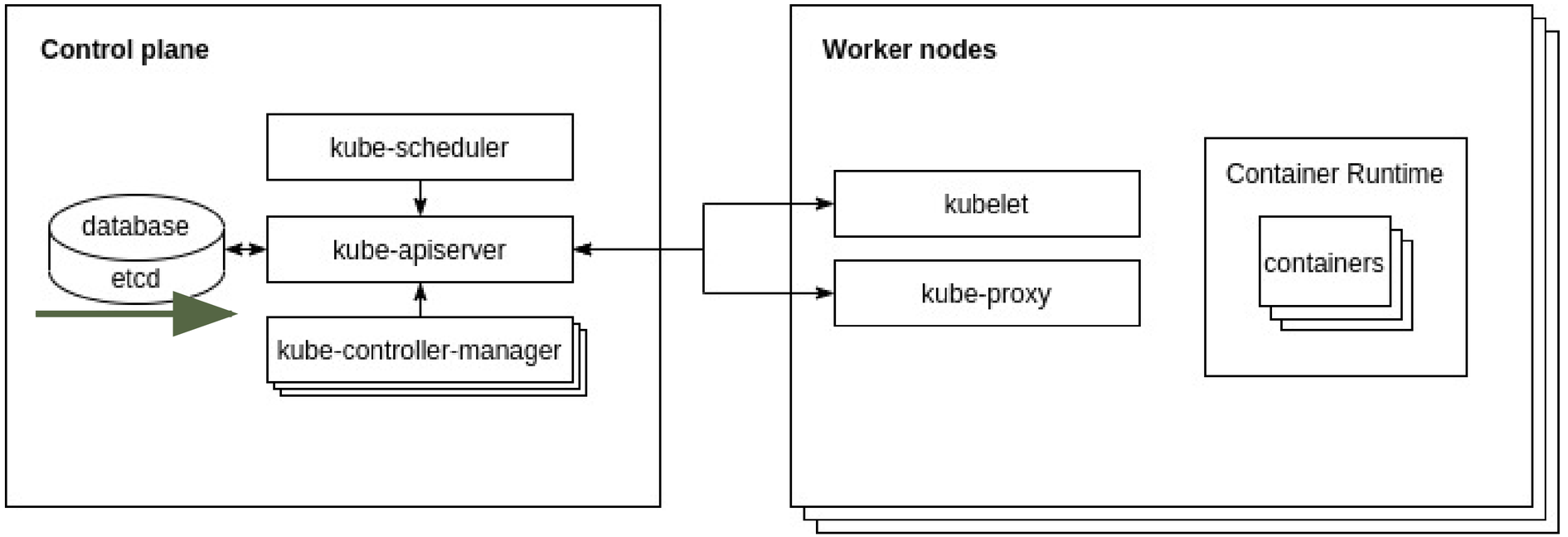
Kubernetes

System für deklaratives verwalten von Containern und Resourcen



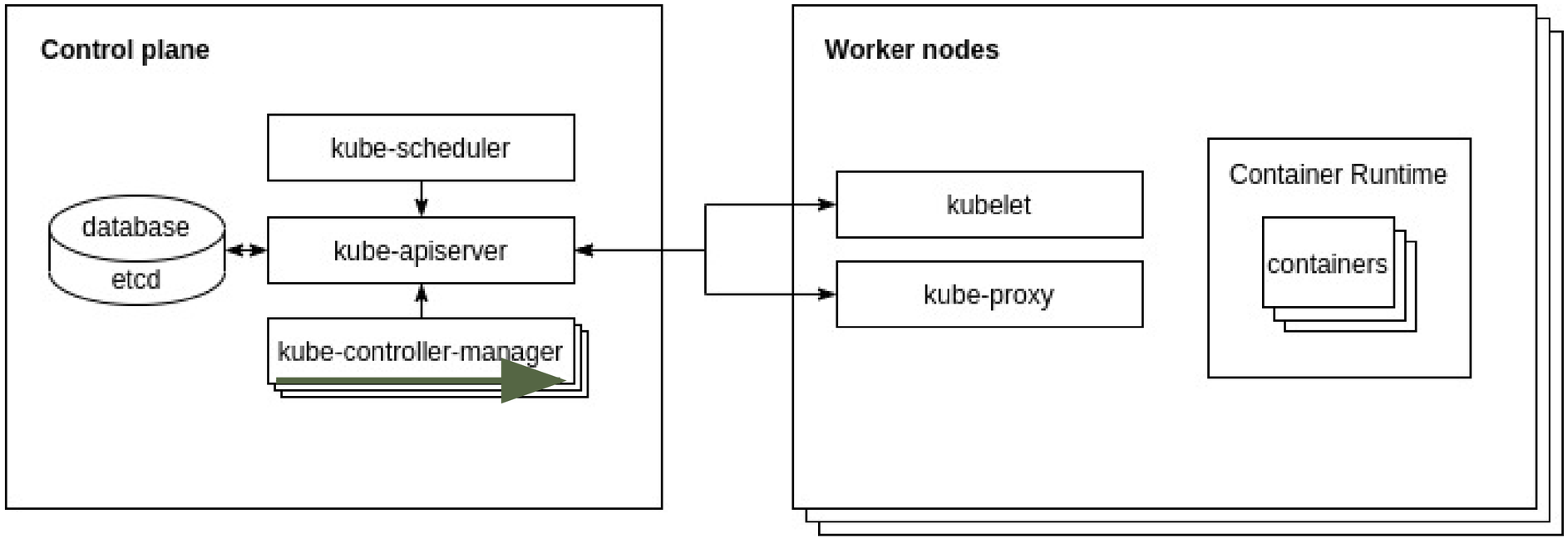
Kubernetes

System für deklaratives verwalten von Containern und Resourcen



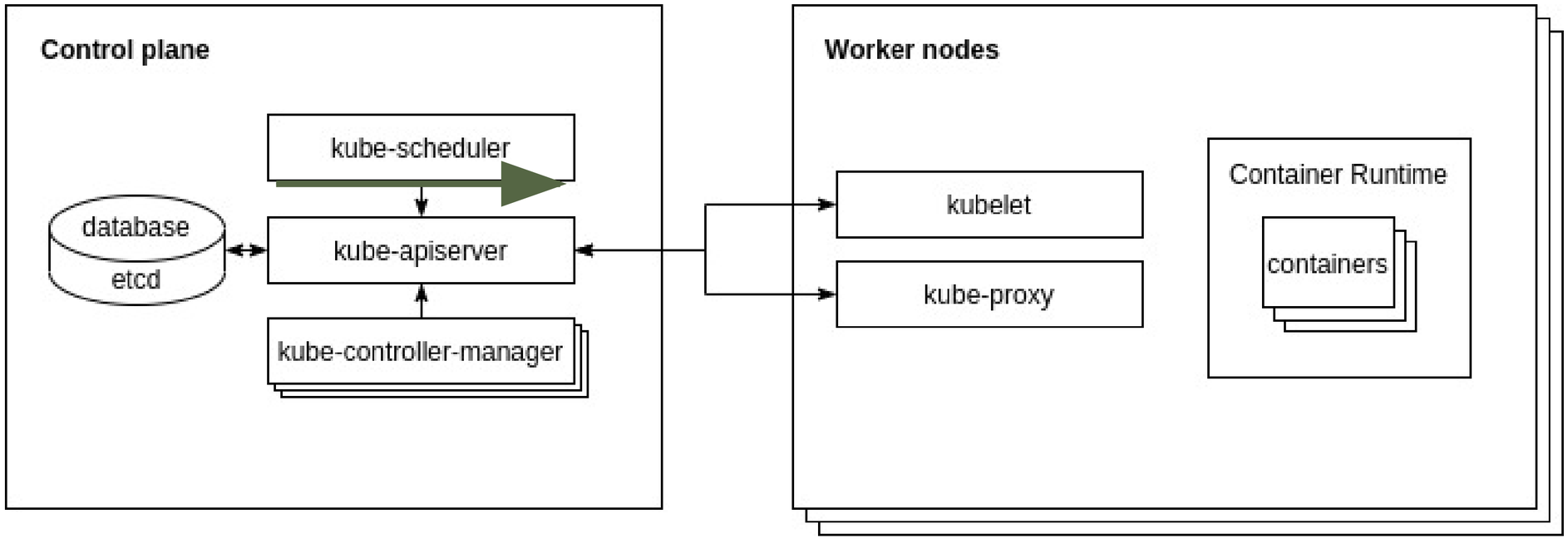
Kubernetes

System für deklaratives verwalten von Containern und Resourcen



Kubernetes

System für deklaratives verwalten von Containern und Resourcen

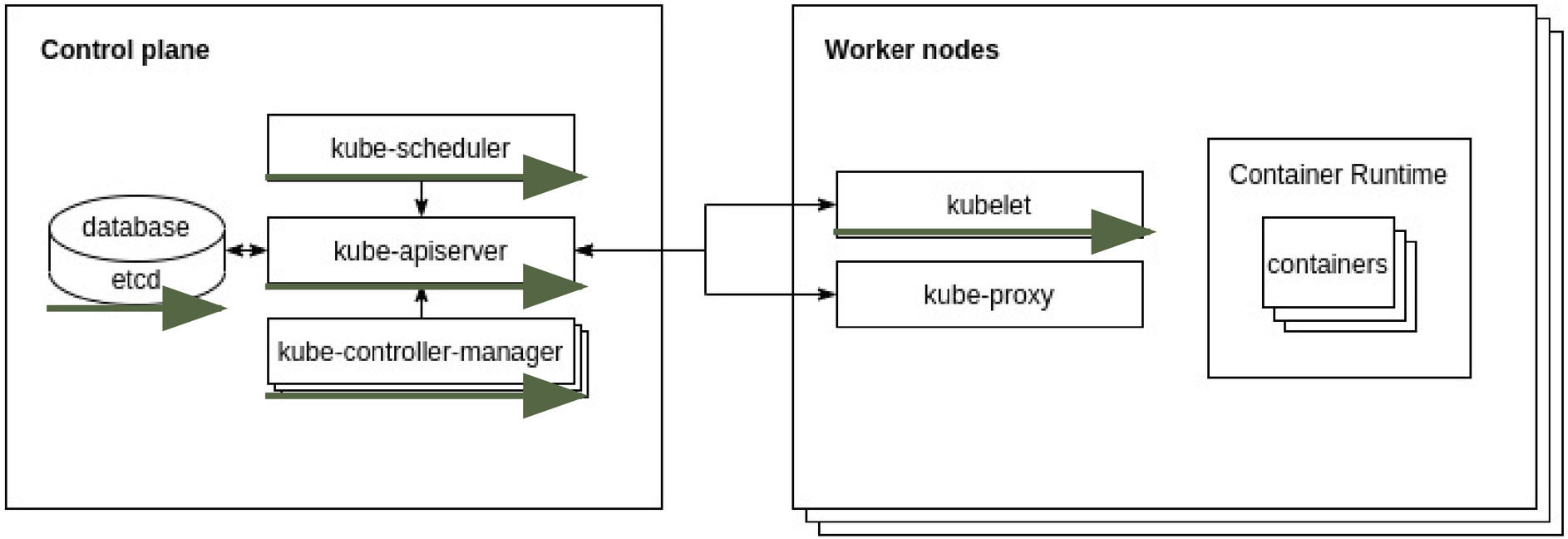


Pod Resource

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: example-webserver
  namespace: example-namespace
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

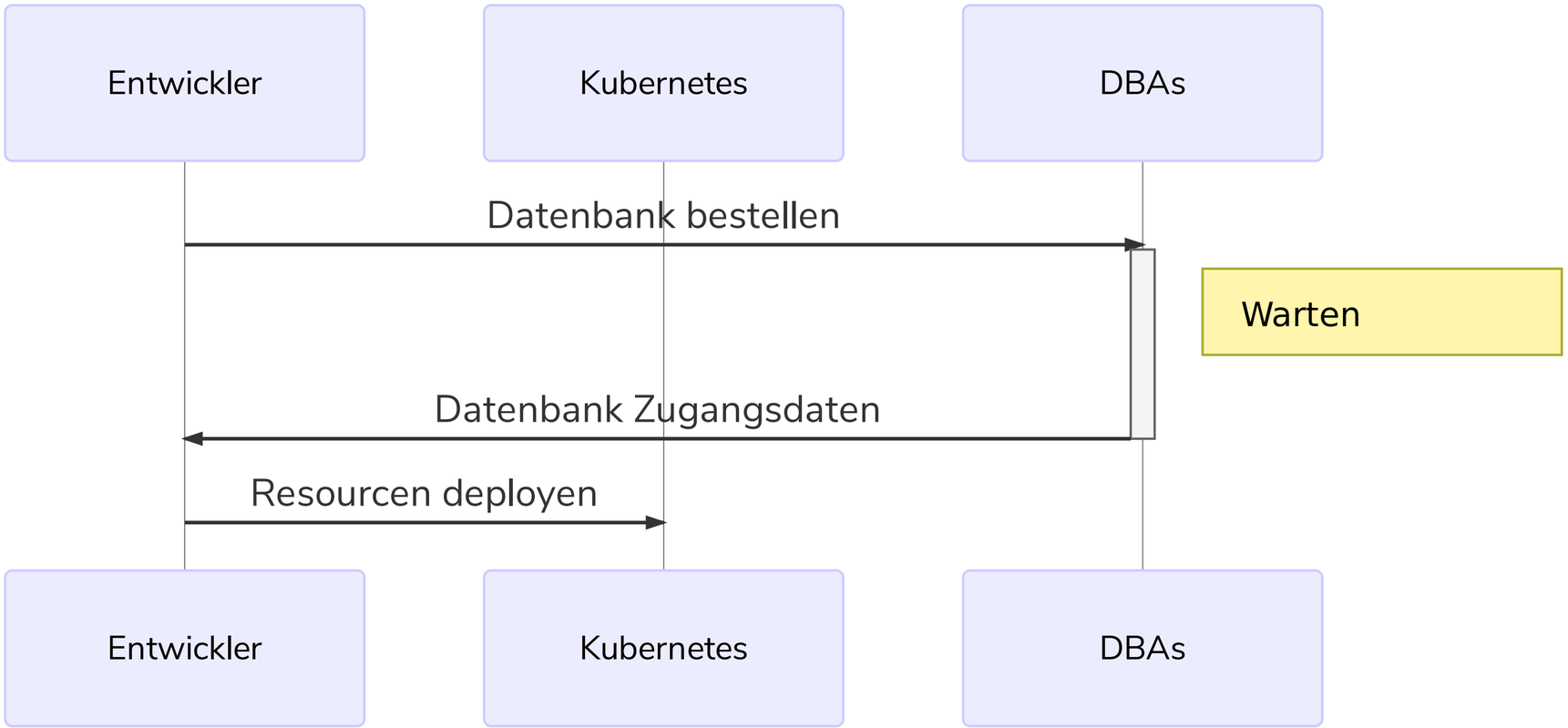
```
kubectl apply --file pod.yaml
```

Pod Resource

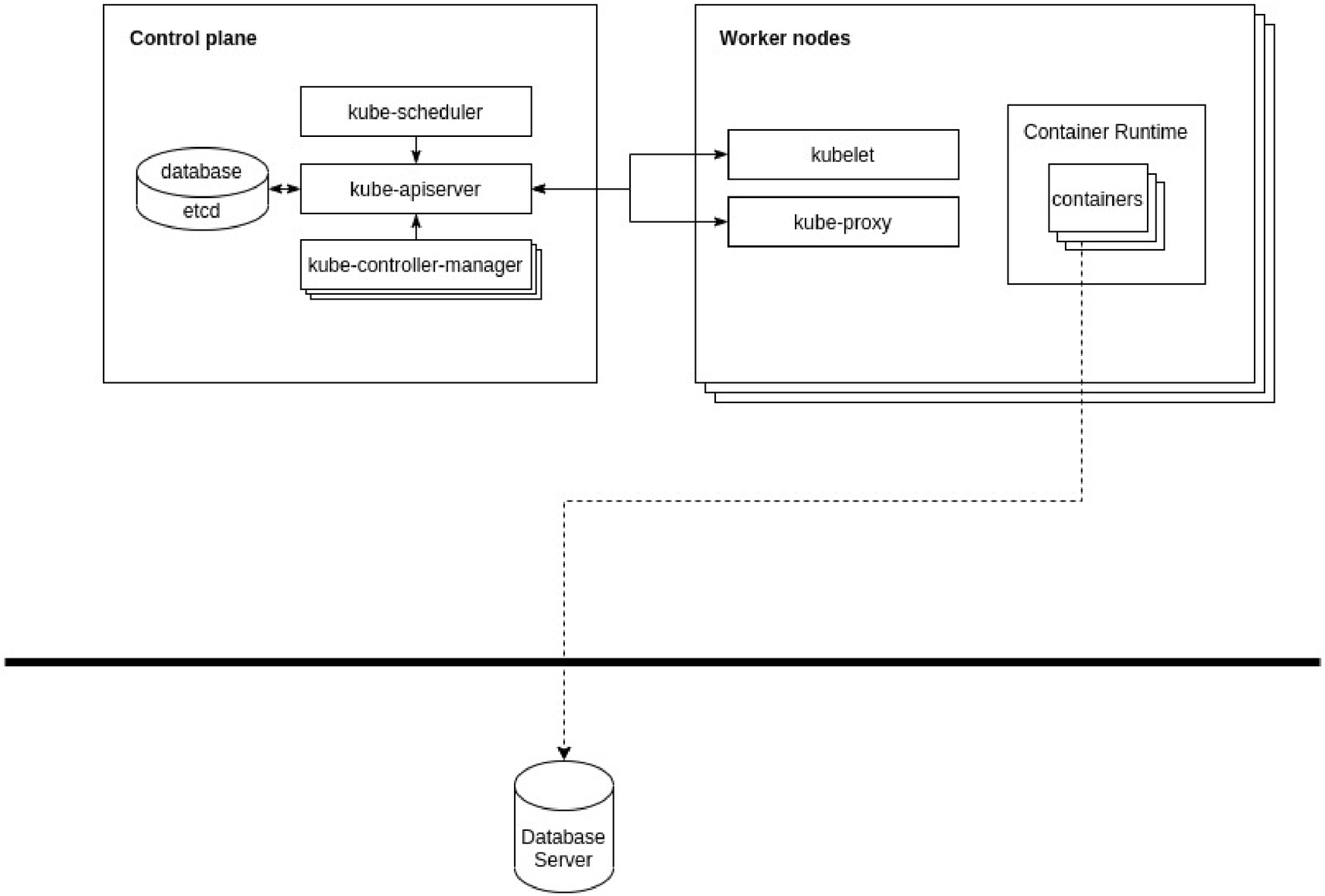


Use Case - IST

Entwickler will eine Applikation deployen die eine zentrale Datenbank braucht

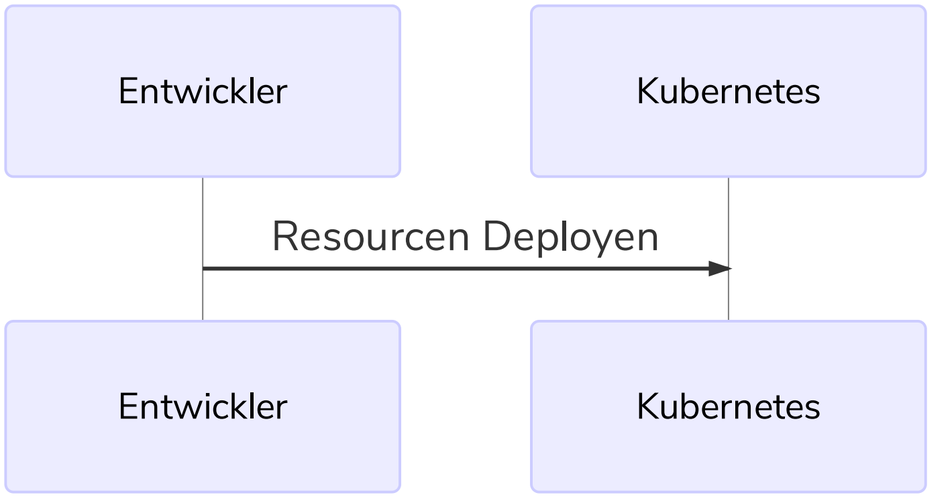


Use Case - IST


Control plane
kube-scheduler
database
etcd
kube-apiserver
kube-controller-manager
Worker nodes
kubelet
kube-proxy
Container Runtime
containers
Database
Server

Use Case - SOLL

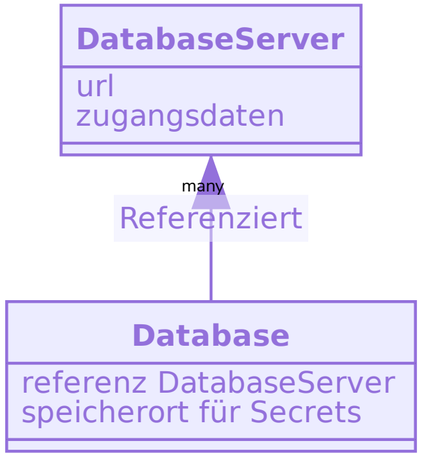
Entwickler will eine Applikation deployen die eine zentrale Datenbank braucht



Use Case - 2 neue Resourcen

- **DatabaseServer Custom Resource**: von Infrastruktur einmalig erstellt und beinhaltet Zugangsdaten zur Datenbank

- **Database Custom Resource**: von Entwickler erstellt und referenziert auf ein **DatabaseServer**. Anhand dieser Resource sollen Datenbanken und User erstellt werden. Die Zugangsdaten sollen dann als kubernetes Resource zur Verfügung gestellt werden.



Use Case - DatabaseServer

```yaml
apiVersion: autodba.sircremefresh.dev/v1alpha1
kind: ClusterDatabaseServer
metadata:
  name: database-server-prd
spec:
  host: 'instance.database-prd'
  port: '5432'
  authSecretRef:
    name: 'database-server-secret'
  databaseType: 'PostgreSQL'
```

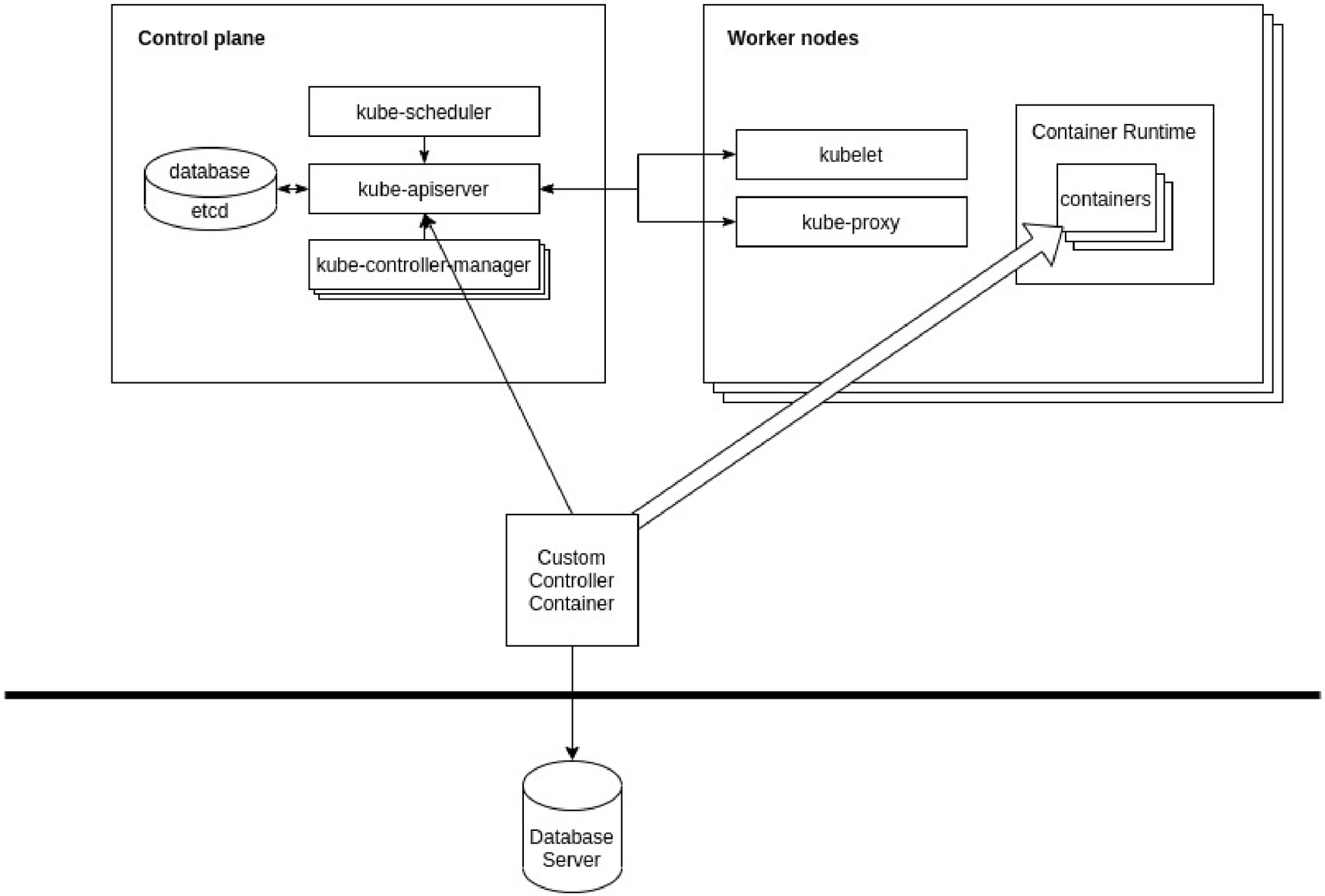
Use Case - Database

```yaml
apiVersion: autodba.sircremefresh.dev/v1alpha1
kind: Database
metadata:
  namespace: voting-prd
  name: voting-database
spec:
  secretName: 'voting-db-secret'
  serverRef:
    name: 'database-server-prd'
```

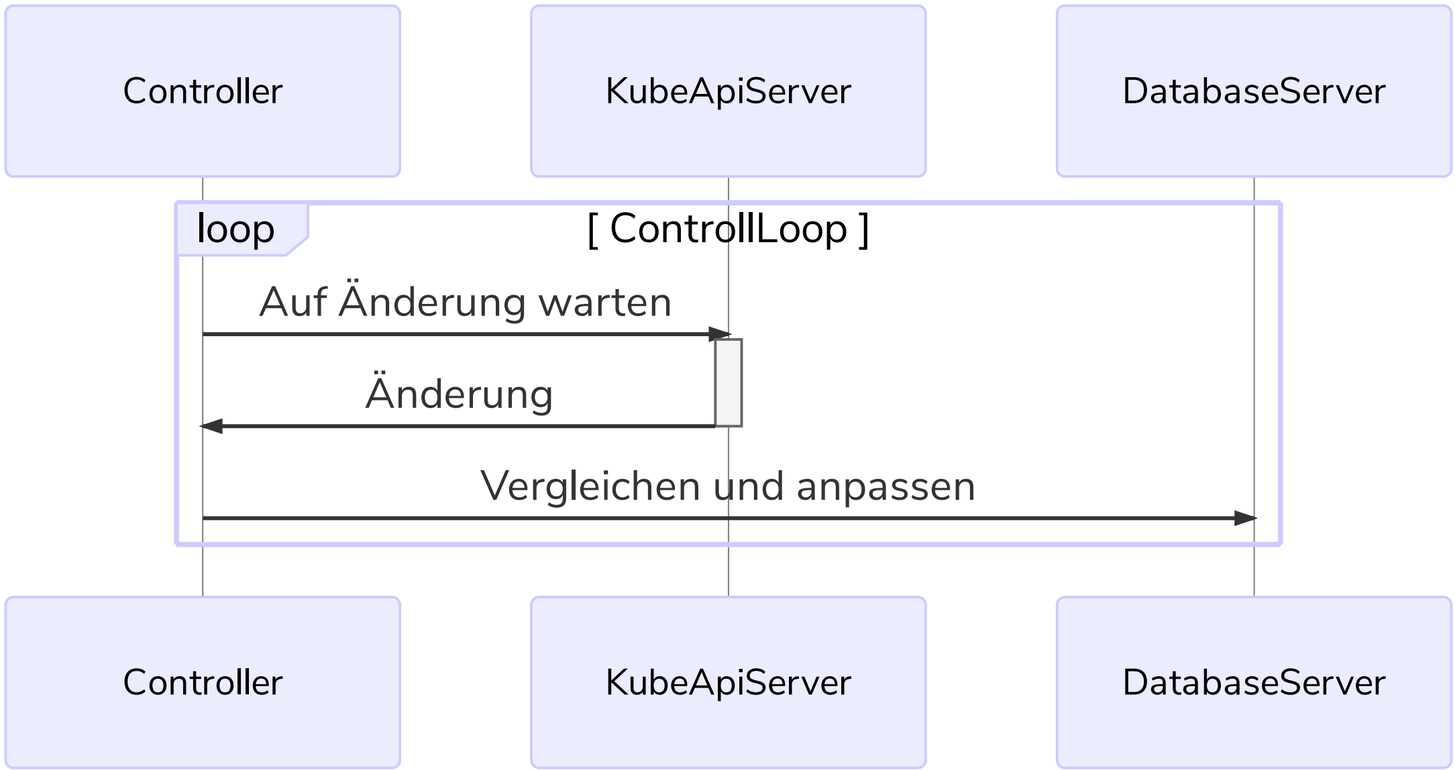
Custom Resource

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: databases.autodba.sircremefresh.dev
spec:
  group: autodba.sircremefresh.dev
  names:
    kind: Database
    listKind: DatabaseList
    plural: databases
    singular: database
  scope: Namespaced
  versions:
    - name: v1alpha1
      schema:
        openAPIV3Schema:
          ...
```

Custom Controller



Custom Controller



Impl - Library

fabric8

Impl - Database

```
@Version("v1alpha1")
@Group("autodba.sircremefresh.dev")
@Kind("Database")
@Plural("databases")
@Singular("database")
public class Database extends CustomResource<DatabaseSpec, DatabaseStatus> implements Namespaced {
}

public class DatabaseSpec {
    private String secretName;
    private DatabaseServerRef serverRef;
}
```

Impl - Events

```
var databaseInformer =
  informerFactory.sharedIndexInformerForCustomResource(Database.class, DatabaseList.class, RESYNC_PERIOD_MILLIS);

databaseInformer.addEventHandler(new ResourceEventHandler<>() {
...
            @Override
            public void onUpdate(Database oldDatabase, Database newDatabase) {
                logger.info("Database {} updated", Cache.metaNamespaceKeyFunc(newDatabase));
                enqueueDatabase(newDatabase);
            }
...
        });
```

Impl - WorkQueue

```
public void run() {
...
  while (!Thread.currentThread().isInterrupted()) {
    String key = workQueue.take();
...
    String namespace = key.split("/")[0];
    String name = key.split("/")[1];
    Database database = databaseLister
        .namespace(namespace)
        .get(name);
...
    databaseReconciler.reconcile(database, databaseServer, secret);
...
    }
}
```

Impl - Secret

```
var secret = new SecretBuilder()
    .withNewMetadata()
    .withName(secretName)
...
    .endMetadata()
    .addToStringData(DATABASE_KEY, databaseKey)
    .addToStringData(USERNAME_KEY, databaseKey)
    .addToStringData(PASSWORD_KEY, password)
    .addToStringData(HOST_KEY, databaseServer.getSpec().getHost())
    .addToStringData(PORT_KEY, databaseServer.getSpec().getPort())
    .addToStringData(URL_KEY, getDatabaseUrl(databaseServer, databaseKey, password))
    .build();

client.secrets().createOrReplace(secret);
```