Kubernetes DNS example

This is a toy example demonstrating how to use kubernetes DNS.

Step Zero: Prerequisites

This example assumes that you have forked the repository and turned up a Kubernetes cluster. Make sure DNS is enabled in your setup, see DNS doc.

```
$ cd kubernetes
$ hack/dev-build-and-up.sh
```

Step One: Create two namespaces

We'll see how cluster DNS works across multiple namespaces, first we need to create two namespaces:

```
$ kubectl create -f examples/cluster-dns/namespace-dev.yaml
$ kubectl create -f examples/cluster-dns/namespace-prod.yaml
```

Now list all namespaces:

```
$ kubectl get namespaces
```

```
NAME          LABELS              STATUS
default       <none>              Active
development   name=development    Active
production    name=production     Active
```

For kubectl client to work with each namespace, we define two contexts:

```
$ kubectl config set-context dev --namespace=development
--cluster=${CLUSTER_NAME} --user=${USER_NAME}
$ kubectl config set-context prod --namespace=production
--cluster=${CLUSTER_NAME} --user=${USER_NAME}
```

You can view your cluster name and user name in kubernetes config at ~/.kube/config.

Step Two: Create backend replication controller in each namespace

Use the file `examples/cluster-dns/dns-backend-rc.yaml` to create a backend server replication controller in each namespace.

```
$ kubectl config use-context dev
$ kubectl create -f examples/cluster-dns/dns-backend-rc.y
aml
```

Once that’s up you can list the pod in the cluster:

```
$ kubectl get rc
CONTROLLER     CONTAINER(S)   IMAGE(S)                    SELECT
OR             REPLICAS
dns-backend    dns-backend    ddysher/dns-backend    name=d
ns-backend     1
```

Now repeat the above commands to create a replication controller in prod namespace:

```
$ kubectl config use-context prod
$ kubectl create -f examples/cluster-dns/dns-backend-rc.y
aml
$ kubectl get rc
CONTROLLER     CONTAINER(S)   IMAGE(S)                    SELECT
OR             REPLICAS
dns-backend    dns-backend    ddysher/dns-backend    name=d
ns-backend     1
```

Step Three: Create backend service

Use the file `examples/cluster-dns/dns-backend-service.yaml` to create
a service for the backend server.

```
$ kubectl config use-context dev
$ kubectl create -f examples/cluster-dns/dns-backend-serv
ice.yaml
```

Once that’s up you can list the service in the cluster:

```
$ kubectl get service dns-backend
NAME          CLUSTER_IP        EXTERNAL_IP        PORT(S)
               SELECTOR           AGE
dns-backend   10.0.2.3          <none>             8000/TCP
               name=dns-backend   1d
```

Again, repeat the same process for prod namespace:

```
$ kubectl config use-context prod
$ kubectl create -f examples/cluster-dns/dns-backend-serv
ice.yaml
$ kubectl get service dns-backend
NAME          CLUSTER_IP        EXTERNAL_IP        PORT(S)
               SELECTOR           AGE
dns-backend   10.0.2.4          <none>             8000/TCP
               name=dns-backend   1d
```

Step Four: Create client pod in one namespace

Use the file `examples/cluster-dns/dns-frontend-pod.yaml` to create a client pod in dev namespace. The client pod will make a connection to backend and exit. Specifically, it tries to connect to address `http://dns-backend.development.cluster.local:8000`.

```
$ kubectl config use-context dev
$ kubectl create -f examples/cluster-dns/dns-frontend-pod
.yaml
```

Once that's up you can list the pod in the cluster:

```
$ kubectl get pods dns-frontend
NAME               READY     STATUS        RESTARTS   AGE
dns-frontend       0/1       ExitCode:0    0          1m
```

Wait until the pod succeeds, then we can see the output from the client pod:

```
$ kubectl logs dns-frontend
2015-05-07T20:13:54.147664936Z 10.0.236.129
2015-05-07T20:13:54.147721290Z Send request to: http://dn
s-backend.development.cluster.local:8000
2015-05-07T20:13:54.147733438Z <Response [200]>
2015-05-07T20:13:54.147738295Z Hello World!
```

Please refer to the source code about the log. First line prints out the ip address associated with the service in dev namespace; remaining lines print out our request and server response.

If we switch to prod namespace with the same pod config, we'll see the same result, i.e. dns will resolve across namespace.

```
$ kubectl config use-context prod
$ kubectl create -f examples/cluster-dns/dns-frontend-pod
.yaml
$ kubectl logs dns-frontend
2015-05-07T20:13:54.147664936Z 10.0.236.129
2015-05-07T20:13:54.147721290Z Send request to: http://dn
s-backend.development.cluster.local:8000
2015-05-07T20:13:54.147733438Z <Response [200]>
2015-05-07T20:13:54.147738295Z Hello World!
```

Note about default namespace

If you prefer not using namespace, then all your services can be addressed using `default` namespace, e.g. `http://dns-backend.default.svc.cluster.local:8000`, or shorthand version `http://dns-backend:8000`

tl; dr;

For those of you who are impatient, here is the summary of the commands we ran in this tutorial. Remember to set first `$CLUSTER_NAME` and `$USER_NAME` to the values found in

`~/.kube/config`.

```
# create dev and prod namespaces
kubectl create -f examples/cluster-dns/namespace-dev.yaml
kubectl create -f examples/cluster-dns/namespace-prod.yaml

# create two contexts
kubectl config set-context dev --namespace=development --cluster=${CLUSTER_NAME} --user=${USER_NAME}
kubectl config set-context prod --namespace=production --cluster=${CLUSTER_NAME} --user=${USER_NAME}

# create two backend replication controllers
kubectl config use-context dev
kubectl create -f examples/cluster-dns/dns-backend-rc.yaml
kubectl config use-context prod
kubectl create -f examples/cluster-dns/dns-backend-rc.yaml

# create backend services
kubectl config use-context dev
kubectl create -f examples/cluster-dns/dns-backend-service.yaml
kubectl config use-context prod
kubectl create -f examples/cluster-dns/dns-backend-service.yaml
```

```

# create a pod in each namespace and get its output
kubectl config use-context dev
kubectl create -f examples/cluster-dns/dns-frontend-pod.y
aml
kubectl logs dns-frontend

kubectl config use-context prod
kubectl create -f examples/cluster-dns/dns-frontend-pod.y
aml
kubectl logs dns-frontend
```

<!-- BEGIN MUNGE: GENERATED_ANALYTICS -->

<!-- END MUNGE: GENERATED_ANALYTICS -->