Expose Pod Information to Containers Through Environment Variables

This page shows how a Pod can use environment variables to expose information about itself to Containers running in the Pod. Environment variables can expose Pod fields and Container fields.

Before you begin

You need to have a Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster. If you do not already have a cluster, you can create one by using minikube or you can use one of these Kubernetes playgrounds:

- Katacoda
- Play with Kubernetes

To check the version, enter `kubectl version`.

The Downward API

There are two ways to expose Pod and Container fields to a running Container:

- Environment variables
- Volume Files

Together, these two ways of exposing Pod and Container fields are called the *Downward API*.

Use Pod fields as values for environment variables

In this exercise, you create a Pod that has one Container. Here is the configuration file for the Pod:

pods/inject/dapi-envars-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: dapi-envars-fieldref
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "sh", "-c"]
      args:
      - while true; do
          echo -en '\n';
          printenv MY_NODE_NAME MY_POD_NAME MY_POD_NAMESPACE;
          printenv MY_POD_IP MY_POD_SERVICE_ACCOUNT;
          sleep 10;
        done;
```

```
    env:
      - name: MY_NODE_NAME
        valueFrom:
          fieldRef:
            fieldPath: spec.nodeName
      - name: MY_POD_NAME
        valueFrom:
          fieldRef:
            fieldPath: metadata.name
      - name: MY_POD_NAMESPACE
        valueFrom:
          fieldRef:
            fieldPath: metadata.namespace
      - name: MY_POD_IP
        valueFrom:
          fieldRef:
            fieldPath: status.podIP
      - name: MY_POD_SERVICE_ACCOUNT
        valueFrom:
          fieldRef:
            fieldPath: spec.serviceAccountName
  restartPolicy: Never
```

In the configuration file, you can see five environment variables. The `env` field is an array of EnvVars. The first element in the array specifies that the `MY_NODE_NAME` environment variable gets its value from the Pod's `spec.nodeName` field. Similarly, the other environment variables get their names from Pod fields.

> **Note:** The fields in this example are Pod fields. They are not fields of the Container in the Pod.

Create the Pod:

```
kubectl apply -f https://k8s.io/examples/pods/inject/dapi-envars-pod.yaml
```

Verify that the Container in the Pod is running:

```
kubectl get pods
```

View the Container's logs:

```
kubectl logs dapi-envars-fieldref
```

The output shows the values of selected environment variables:

```
minikube
dapi-envars-fieldref
default
172.17.0.4
default
```

To see why these values are in the log, look at the `command` and `args` fields in the configuration file. When the Container starts, it writes the values of five environment variables to stdout. It repeats this every ten seconds.

Next, get a shell into the Container that is running in your Pod:

```
kubectl exec -it dapi-envars-fieldref -- sh
```

In your shell, view the environment variables:

```
/# printenv
```

The output shows that certain environment variables have been assigned the values of Pod fields:

```
MY_POD_SERVICE_ACCOUNT=default
...
MY_POD_NAMESPACE=default
MY_POD_IP=172.17.0.4
...
MY_NODE_NAME=minikube
...
MY_POD_NAME=dapi-envars-fieldref
```

Use Container fields as values for environment variables

In the preceding exercise, you used Pod fields as the values for environment variables. In this next exercise, you use Container fields as the values for environment variables. Here is the configuration file for a Pod that has one container:

pods/inject/dapi-envars-container.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: dapi-envars-resourcefieldref
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox:1.24
      command: [ "sh", "-c"]
      args:
      - while true; do
          echo -en '\n';
          printenv MY_CPU_REQUEST MY_CPU_LIMIT;
          printenv MY_MEM_REQUEST MY_MEM_LIMIT;
          sleep 10;
        done;
      resources:
        requests:
          memory: "32Mi"
          cpu: "125m"
        limits:
          memory: "64Mi"
          cpu: "250m"
      env:
        - name: MY_CPU_REQUEST
          valueFrom:
            resourceFieldRef:
              containerName: test-container
              resource: requests.cpu
        - name: MY_CPU_LIMIT
          valueFrom:
            resourceFieldRef:
              containerName: test-container
```

```
            resource: limits.cpu
      - name: MY_MEM_REQUEST
        valueFrom:
          resourceFieldRef:
            containerName: test-container
            resource: requests.memory
      - name: MY_MEM_LIMIT
        valueFrom:
          resourceFieldRef:
            containerName: test-container
            resource: limits.memory
  restartPolicy: Never
```

In the configuration file, you can see four environment variables. The `env` field is an array of EnvVars. The first element in the array specifies that the `MY_CPU_REQUEST` environment variable gets its value from the `requests.cpu` field of a Container named `test-container`. Similarly, the other environment variables get their values from Container fields.

Create the Pod:

```
kubectl apply -f https://k8s.io/examples/pods/inject/dapi-envars-container.yaml
```

Verify that the Container in the Pod is running:

```
kubectl get pods
```

View the Container's logs:

```
kubectl logs dapi-envars-resourcefieldref
```

The output shows the values of selected environment variables:

```
1
1
33554432
67108864
```

What's next

- Defining Environment Variables for a Container
- PodSpec
- Container
- EnvVar
- EnvVarSource
- ObjectFieldSelector
- ResourceFieldSelector

Feedback

Was this page helpful?

Yes No

Last modified May 30, 2020 at 3:10 PM PST: add en pages (ecc27bbbe)