Persisting State through the emptyDir Volume Type

In this lesson, we will analyze the state of an updated Jenkins Deployment and discuss emptyDir Volume type.

WE'LL COVER THE FOLLOWING

Updating the Jenkins Deployment Definition

Let's take a look at a slightly updated YAML definition.

```
cat volume/jenkins-empty-dir.yml
```

The **output**, limited to the relevant parts, is as follows.

```
...
kind: Deployment
...
spec:
  ...
  template:
    ...
    spec:
      containers:
        ...
        volumeMounts:
        - mountPath: /var/jenkins_home
          name: jenkins-home
      volumes:
      - emptyDir: {}
        name: jenkins-home
...
```

We added a mount that references the `jenkins-home` Volume. The Volume type is, this time, `emptyDir`. We'll discuss the new Volume type soon. But, before we dive into explanations, we'll try to experience its effects dive into explanations, we'll try to experience its effects.

```
kubectl apply \
    -f volume/jenkins-empty-dir.yml

kubectl rollout status deploy jenkins
```

We applied the new definition and waited until the rollout finished.

Now we can open the *New Job* Jenkins screen and repeat the same process we followed before.

```
open "http://$(minikube ip)/jenkins/newJob"
```

Please type *test* in the *item name* field, select *Pipeline* as the type, click the *OK* button, and finish by clicking the *Save* button.

Now we'll kill the container and see what happens.

```
POD_NAME=$(kubectl get pods \
    -l service=jenkins,type=master \
    -o jsonpath="{.items[*].metadata.name}")

kubectl exec -it $POD_NAME kill 1

kubectl get pods
```

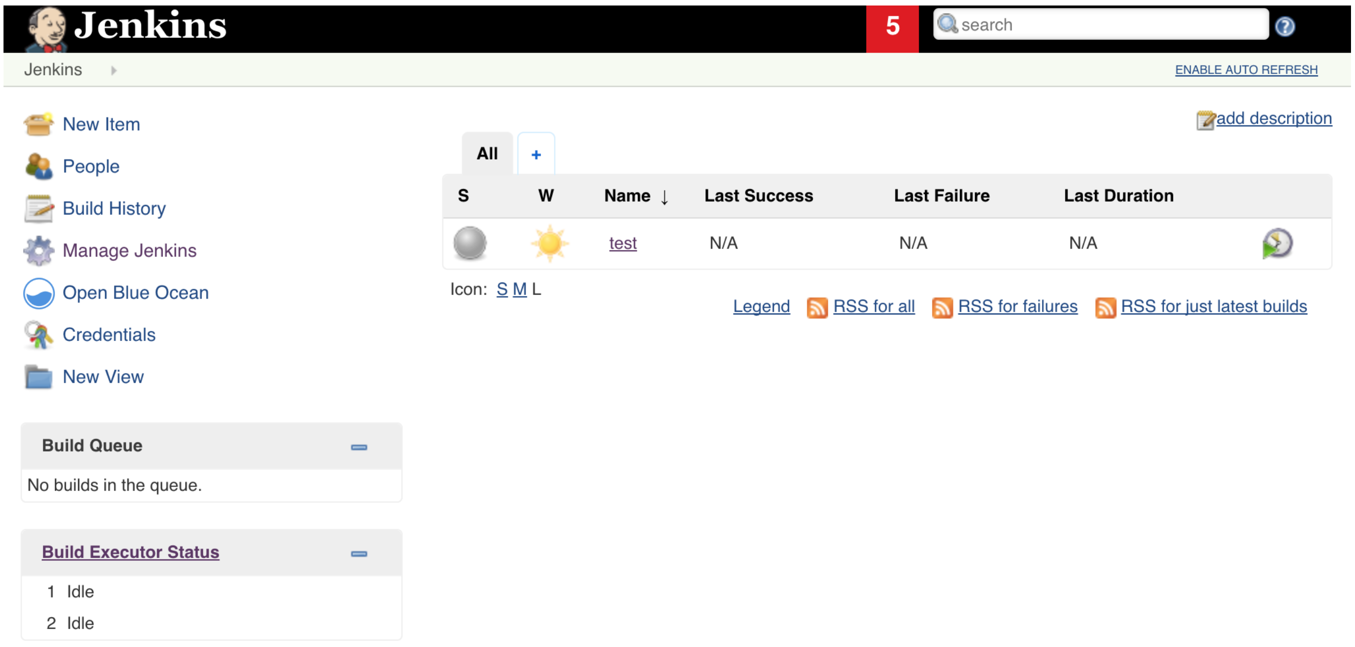
The **output** should show that there is a container running or, in other words, that Kubernetes detected the failure and created a new container.

Persisting State

Finally, let's open Jenkins' Home screen one more time.

```
open "http://$(minikube ip)/jenkins"
```

This time, the `test` job is there. The state of the application was preserved even when the container failed, and Kubernetes created a new one.



S	W	Name ↓	Last Success	Last Failure	Last Duration
		test	N/A	N/A	N/A

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Jenkins with preserved state

The emptyDir Volume

Now let's talk about the `emptyDir` Volume. It is considerably different from those we explored thus far.

> An `emptyDir` Volume is created when a Pod is assigned to a node. It will exist for as long as the Pod continues running on that server.

What that means is that `emptyDir` can survive container failures. When a container crashes, a Pod is not removed from the node. Instead, Kubernetes will recreate the failed container inside the same Pod and, thus, preserve the `emptyDir` Volume. All in all, this Volume type is only partially fault-tolerant.

If `emptyDir` is not entirely fault-tolerant, you might be wondering why we are discussing it in the first place.

The `emptyDir` Volume type is closest we can get to fault-tolerant volumes without using a network drive. Since we do not have any, we had to resort to `emptyDir` as the-closest-we-can-get-to-fault-tolerant-persistence type of Volume.

As you start deploying third-party applications, you'll discover that many of them come with the recommended YAML definition. If you pay closer attention, you'll notice that many are using `emptyDir` Volume type. It's not that

`emptyDir` is the best choice, but that it all depends on your needs, your hosting provider, your infrastructure, and quite a few other things.

There is no one-size-fits-all type of persistent and fault-tolerant Volume type. On the other hand, `emptyDir` always works. Since it has no external dependencies, it is safe to put it as an example, with the assumption that people will change to whichever type fits them better.

> There is an unwritten assumption that `emptyDir` is used for testing purposes, and will be changed to something else before it reaches production.

As long as we're using Minikube to create a Kubernetes cluster, we'll use `emptyDir` as a solution for persistent volumes. Do not despair. Later on, once we move into a "more serious" cluster setup, we'll explore better options for persisting state.

In the next lesson, we will test your understanding of Volumes with the help of a quick quiz.