**Study Material: Rolling Updates and Self-Healing in Kubernetes**

This study material will guide you through the concepts of **Rolling Updates** and **Self-Healing** in Kubernetes. We'll cover how Kubernetes handles updates to your application without downtime and how it ensures the desired state of your deployment is maintained.

**1. Introduction to Rolling Updates**

**Objective**

* Understand how Kubernetes performs rolling updates to deploy new versions of an application.
* Learn how to update a deployment with a new image version.
* Observe how Kubernetes replaces old pods with new ones without downtime.

**2. Key Concepts**

**Rolling Update Strategy**

* **Definition**: A deployment strategy where new pods are created with the updated version of the application while old pods are gradually terminated.
* **Use Case**: Ensures zero downtime during application updates by maintaining a balance between old and new pods.

**Self-Healing**

* **Definition**: Kubernetes automatically replaces failed or deleted pods to maintain the desired state of the deployment.
* **Use Case**: Ensures high availability and reliability of applications by automatically recovering from failures.

**3. Step-by-Step Guide**

**Step 1: Modify the Application**

**1.1 Update the Application Code**

* **File**: index.mjs
* **Code**:
* import express from 'express';
* import os from 'os';
* const app = express();
* const port = 3000;
* app.get('/', (req, res) => {
* const hostname = os.hostname();
* res.send(`Hello from version 2.0.0 on ${hostname}`);
* console.log(`Received request from ${req.ip}`);
* });
* app.listen(port, () => {
* console.log(`Web server is listening at port ${port}`);
* });
* **Explanation**: We modified the response message to include version 2.0.0 to indicate the new version of the application.

**Step 2: Build and Push the New Docker Image**

**2.1 Build the Docker Image with a New Tag**

* **Command**: docker build -t <your-dockerhub-username>/k8s-web-hello:2.0.0 .
* **Explanation**: This command builds a new Docker image with the tag 2.0.0.

**2.2 Push the New Image to Docker Hub**

* **Command**: docker push <your-dockerhub-username>/k8s-web-hello:2.0.0
* **Explanation**: This command pushes the new image to Docker Hub, making it available for deployment.

**Step 3: Update the Kubernetes Deployment**

**3.1 Set the New Image for the Deployment**

* **Command**: kubectl set image deployment/k8s-web-hello k8s-web-hello=<your-dockerhub-username>/k8s-web-hello:2.0.0
* **Explanation**: This command updates the deployment to use the new image with the tag 2.0.0.

**3.2 Monitor the Rolling Update**

* **Command**: kubectl rollout status deployment/k8s-web-hello
* **Explanation**: This command monitors the status of the rolling update, showing the progress of replacing old pods with new ones.

**3.3 Verify the Update**

* **Command**: kubectl get pods
* **Explanation**: This command lists the pods, showing the new pods with the updated image.

**Step 4: Access the Updated Application**

**4.1 Access the Application via Service**

* **Command**: minikube service k8s-web-hello
* **Explanation**: This command opens the application in a web browser, allowing you to verify that the new version is running.

**4. Self-Healing in Kubernetes**

**Step 1: Delete a Pod Manually**

**1.1 Delete a Pod**

* **Command**: kubectl delete pod <pod-name>
* **Explanation**: This command deletes a specific pod. Kubernetes will automatically create a new pod to replace the deleted one.

**1.2 Verify Self-Healing**

* **Command**: kubectl get pods
* **Explanation**: This command lists the pods, showing that a new pod has been created to replace the deleted one.

**5. Example Commands and Outputs**

**Example 1: Building and Pushing a New Docker Image**

docker build -t <your-dockerhub-username>/k8s-web-hello:2.0.0 .

docker push <your-dockerhub-username>/k8s-web-hello:2.0.0

**Output**:

Successfully built <image-id>

Successfully tagged <your-dockerhub-username>/k8s-web-hello:2.0.0

The push refers to repository [docker.io/<your-dockerhub-username>/k8s-web-hello]

**Example 2: Updating the Deployment**

kubectl set image deployment/k8s-web-hello k8s-web-hello=<your-dockerhub-username>/k8s-web-hello:2.0.0

kubectl rollout status deployment/k8s-web-hello

**Output**:

deployment.apps/k8s-web-hello image updated

Waiting for deployment "k8s-web-hello" rollout to finish: 2 out of 4 new replicas have been updated...

deployment "k8s-web-hello" successfully rolled out

**Example 3: Deleting a Pod**

kubectl delete pod k8s-web-hello-12345

kubectl get pods

**Output**:

pod "k8s-web-hello-12345" deleted

NAME READY STATUS RESTARTS AGE

k8s-web-hello-67890 1/1 Running 0 10s

**6. Conclusion**

By following this guide, you have learned how to:

* Perform rolling updates to deploy new versions of your application without downtime.
* Use Kubernetes' self-healing capabilities to maintain the desired state of your deployment.

These features are essential for maintaining high availability and reliability in production environments.

**7. Additional Resources**

* [Kubernetes Documentation: Rolling Updates](https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/)
* [Kubernetes Documentation: Self-Healing](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#self-healing)

### Study Material: Rolling Updates and Self-Healing in Kubernetes

This study material will guide you through the concepts of \*\*Rolling Updates\*\* and \*\*Self-Healing\*\* in Kubernetes. We'll cover how Kubernetes handles updates to your application without downtime and how it ensures the desired state of your deployment is maintained.

---

## \*\*1. Introduction to Rolling Updates\*\*

### \*\*Objective\*\*

- Understand how Kubernetes performs rolling updates to deploy new versions of an application.

- Learn how to update a deployment with a new image version.

- Observe how Kubernetes replaces old pods with new ones without downtime.

---

## \*\*2. Key Concepts\*\*

### \*\*Rolling Update Strategy\*\*

- \*\*Definition\*\*: A deployment strategy where new pods are created with the updated version of the application while old pods are gradually terminated.

- \*\*Use Case\*\*: Ensures zero downtime during application updates by maintaining a balance between old and new pods.

### \*\*Self-Healing\*\*

- \*\*Definition\*\*: Kubernetes automatically replaces failed or deleted pods to maintain the desired state of the deployment.

- \*\*Use Case\*\*: Ensures high availability and reliability of applications by automatically recovering from failures.

---

## \*\*3. Step-by-Step Guide\*\*

### \*\*Step 1: Modify the Application\*\*

#### \*\*1.1 Update the Application Code\*\*

- \*\*File\*\*: `index.mjs`

- \*\*Code\*\*:

```javascript

import express from 'express';

import os from 'os';

const app = express();

const port = 3000;

app.get('/', (req, res) => {

const hostname = os.hostname();

res.send(`Hello from version 2.0.0 on ${hostname}`);

console.log(`Received request from ${req.ip}`);

});

app.listen(port, () => {

console.log(`Web server is listening at port ${port}`);

});

```

- \*\*Explanation\*\*: We modified the response message to include `version 2.0.0` to indicate the new version of the application.

### \*\*Step 2: Build and Push the New Docker Image\*\*

#### \*\*2.1 Build the Docker Image with a New Tag\*\*

- \*\*Command\*\*: `docker build -t <your-dockerhub-username>/k8s-web-hello:2.0.0 .`

- \*\*Explanation\*\*: This command builds a new Docker image with the tag `2.0.0`.

#### \*\*2.2 Push the New Image to Docker Hub\*\*

- \*\*Command\*\*: `docker push <your-dockerhub-username>/k8s-web-hello:2.0.0`

- \*\*Explanation\*\*: This command pushes the new image to Docker Hub, making it available for deployment.

### \*\*Step 3: Update the Kubernetes Deployment\*\*

#### \*\*3.1 Set the New Image for the Deployment\*\*

- \*\*Command\*\*: `kubectl set image deployment/k8s-web-hello k8s-web-hello=<your-dockerhub-username>/k8s-web-hello:2.0.0`

- \*\*Explanation\*\*: This command updates the deployment to use the new image with the tag `2.0.0`.

#### \*\*3.2 Monitor the Rolling Update\*\*

- \*\*Command\*\*: `kubectl rollout status deployment/k8s-web-hello`

- \*\*Explanation\*\*: This command monitors the status of the rolling update, showing the progress of replacing old pods with new ones.

#### \*\*3.3 Verify the Update\*\*

- \*\*Command\*\*: `kubectl get pods`

- \*\*Explanation\*\*: This command lists the pods, showing the new pods with the updated image.

### \*\*Step 4: Access the Updated Application\*\*

#### \*\*4.1 Access the Application via Service\*\*

- \*\*Command\*\*: `minikube service k8s-web-hello`

- \*\*Explanation\*\*: This command opens the application in a web browser, allowing you to verify that the new version is running.

---

## \*\*4. Self-Healing in Kubernetes\*\*

### \*\*Step 1: Delete a Pod Manually\*\*

#### \*\*1.1 Delete a Pod\*\*

- \*\*Command\*\*: `kubectl delete pod <pod-name>`

- \*\*Explanation\*\*: This command deletes a specific pod. Kubernetes will automatically create a new pod to replace the deleted one.

#### \*\*1.2 Verify Self-Healing\*\*

- \*\*Command\*\*: `kubectl get pods`

- \*\*Explanation\*\*: This command lists the pods, showing that a new pod has been created to replace the deleted one.

---

## \*\*5. Example Commands and Outputs\*\*

### \*\*Example 1: Building and Pushing a New Docker Image\*\*

```bash

docker build -t <your-dockerhub-username>/k8s-web-hello:2.0.0 .

docker push <your-dockerhub-username>/k8s-web-hello:2.0.0

```

\*\*Output\*\*:

```

Successfully built <image-id>

Successfully tagged <your-dockerhub-username>/k8s-web-hello:2.0.0

The push refers to repository [docker.io/<your-dockerhub-username>/k8s-web-hello]

```

### \*\*Example 2: Updating the Deployment\*\*

```bash

kubectl set image deployment/k8s-web-hello k8s-web-hello=<your-dockerhub-username>/k8s-web-hello:2.0.0

kubectl rollout status deployment/k8s-web-hello

```

\*\*Output\*\*:

```

deployment.apps/k8s-web-hello image updated

Waiting for deployment "k8s-web-hello" rollout to finish: 2 out of 4 new replicas have been updated...

deployment "k8s-web-hello" successfully rolled out

```

### \*\*Example 3: Deleting a Pod\*\*

```bash

kubectl delete pod k8s-web-hello-12345

kubectl get pods

```

\*\*Output\*\*:

```

pod "k8s-web-hello-12345" deleted

NAME READY STATUS RESTARTS AGE

k8s-web-hello-67890 1/1 Running 0 10s

```

---

## \*\*6. Visual Aids\*\*

### \*\*Rolling Update Process\*\*

```plaintext

Old Pods: [Pod1, Pod2, Pod3, Pod4]

New Pods: [Pod5, Pod6, Pod7, Pod8]

Step 1: Create Pod5 (New)

Step 2: Terminate Pod1 (Old)

Step 3: Create Pod6 (New)

Step 4: Terminate Pod2 (Old)

...

Final State: [Pod5, Pod6, Pod7, Pod8]

```

### \*\*Self-Healing Process\*\*

```plaintext

Before Deletion: [Pod1, Pod2, Pod3, Pod4]

After Deletion: [Pod2, Pod3, Pod4, Pod5]

```

---

## \*\*7. Conclusion\*\*

By following this guide, you have learned how to:

- Perform rolling updates to deploy new versions of your application without downtime.

- Use Kubernetes' self-healing capabilities to maintain the desired state of your deployment.

These features are essential for maintaining high availability and reliability in production environments.

---

## \*\*8. Additional Resources\*\*

- [Kubernetes Documentation: Rolling Updates](https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/)

- [Kubernetes Documentation: Self-Healing](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/#self-healing)

---

This study material provides a comprehensive understanding of rolling updates and self-healing in Kubernetes, with clear explanations, examples, and visual aids to help you grasp the concepts effectively.