Effortlessly Scale Your Applications with OpenShift Serverless

Willian Rampazzo

What we'll discuss today

- What is OpenShift Serverless?
- Which services are available in OpenShift Serverless?
- OpenShift Serverless demo.

Red Hat

What is OpenShift Serverless?

OpenShift Serverless provides Kubernetes native **building blocks** that enable developers to **create and deploy serverless**, **event-driven applications** on OpenShift Container Platform.

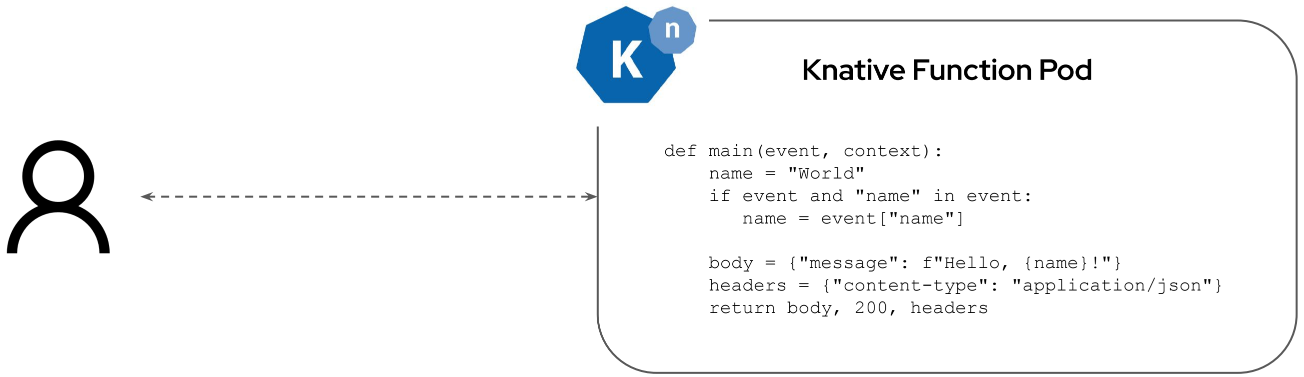
Serverless applications **can scale up and down (to zero) on demand** and be triggered by a number of event sources.

OpenShift Serverless is based on the open source **Knative project**, which provides portability and consistency for hybrid and multi-cloud environments by enabling an enterprise-grade serverless platform.

Source - June, 2025

Which services are available in OpenShift Serverless?

Functions: enables you to create functions that are deployed as Knative Services, leveraging the capabilities of Knative Serving and Eventing.

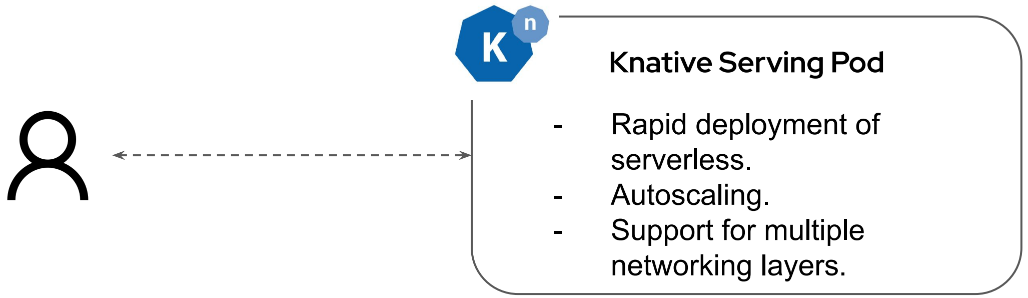


```
def main(event, context):
    name = "World"
    if event and "name" in event:
       name = event["name"]

    body = {"message": f"Hello, {name}!"}
    headers = {"content-type": "application/json"}
    return body, 200, headers
```

Source - June, 2025

Which services are available in OpenShift Serverless?

Serving: enables the deployment and serving of applications and functions through serverless containers. It streamlines application deployment, automatically scales in response to incoming traffic, and provides custom rollout strategies that involve traffic splitting.

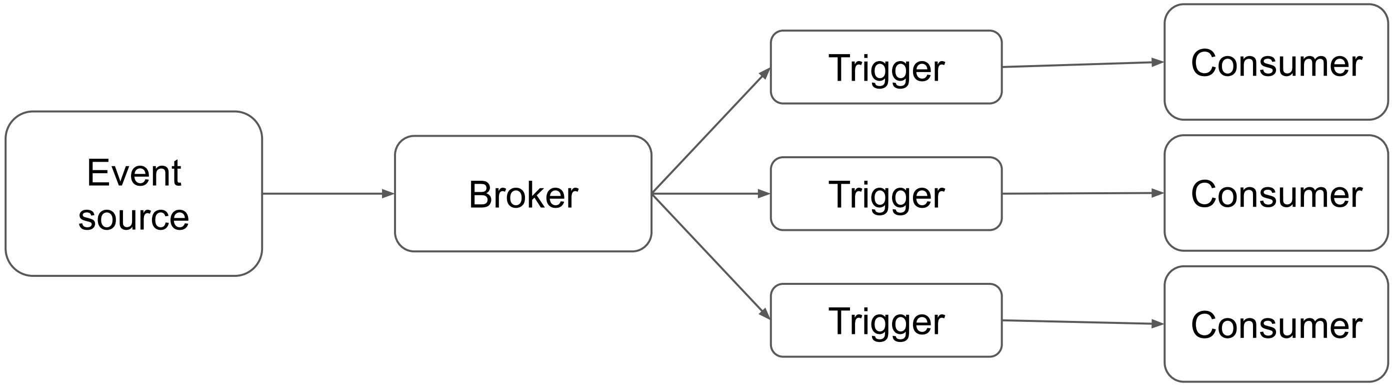


- Rapid deployment of serverless.
- Autoscaling.
- Support for multiple networking layers.

Source - June, 2025

Which services are available in OpenShift Serverless?

Eventing: is a platform that offers flexible components for connecting event sources and consumers on-demand.

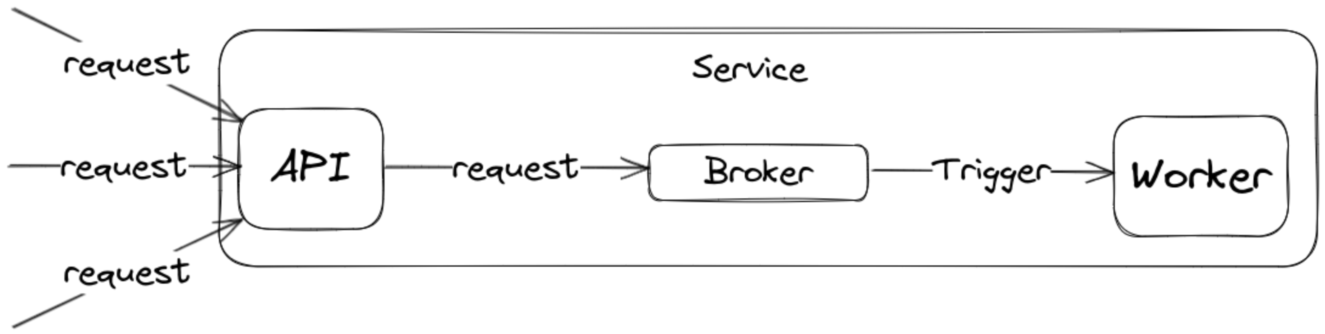


Source - June, 2025

OpenShift Serverless demo: The infrastructure.

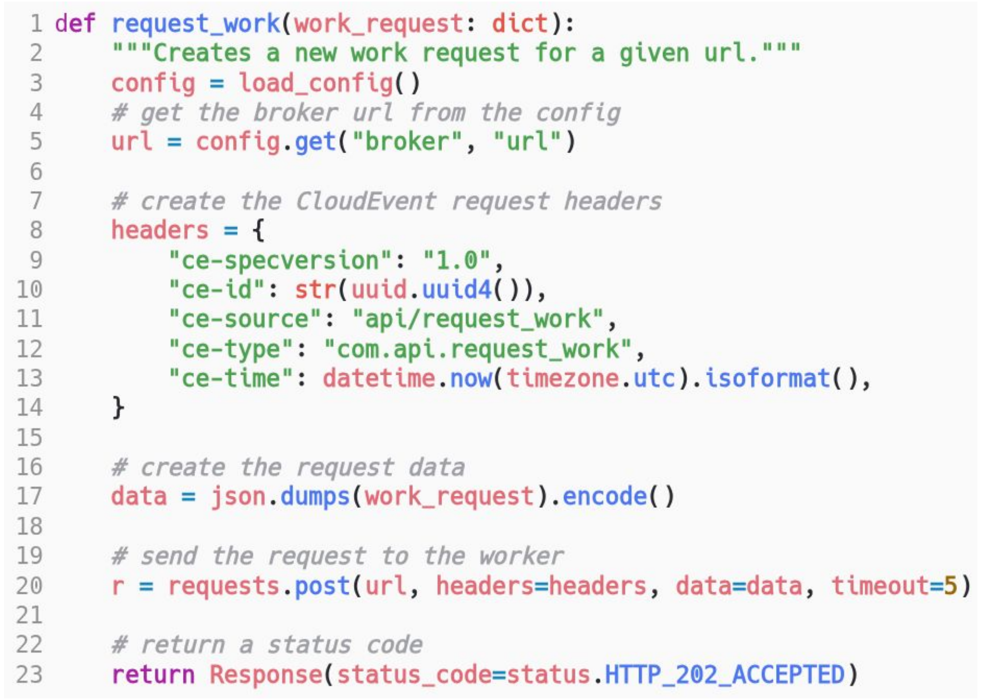
- Red Hat OpenShift Local (formerly Red Hat CodeReady Containers)
- Red Hat OpenShift Serverless

OpenShift Serverless demo: The application.



OpenShift Serverless demo: The application.

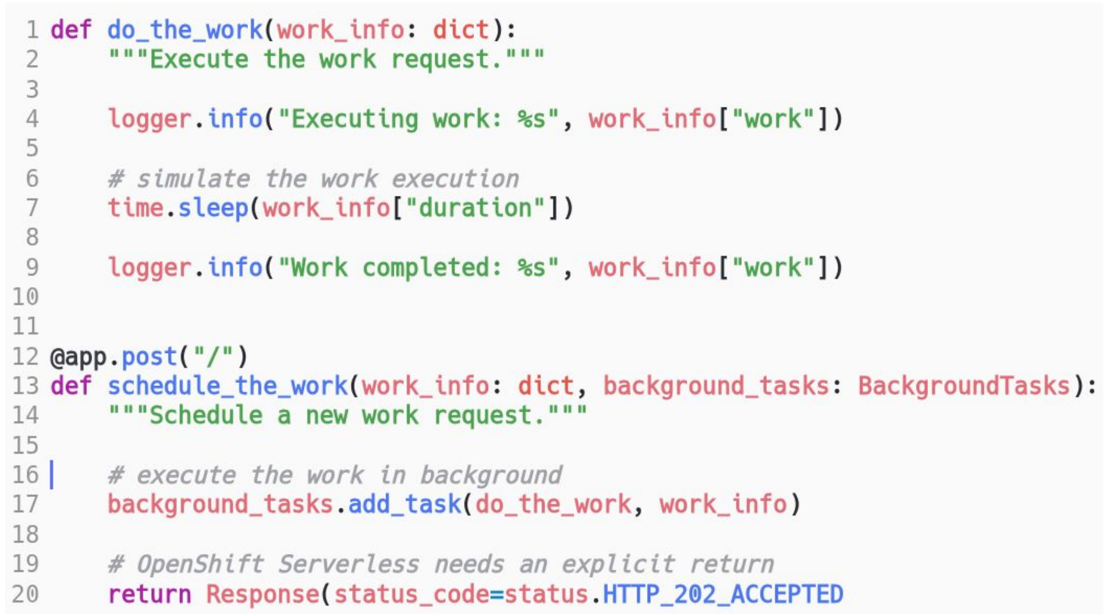
The **API** receives the work request and forwards it to the broker.



```python
def request_work(work_request: dict):
    """Creates a new work request for a given url."""
    config = load_config()
    # get the broker url from the config
    url = config.get("broker", "url")

    # create the CloudEvent request headers
    headers = {
        "ce-specversion": "1.0",
        "ce-id": str(uuid.uuid4()),
        "ce-source": "api/request_work",
        "ce-type": "com.api.request_work",
        "ce-time": datetime.now(timezone.utc).isoformat(),
    }

    # create the request data
    data = json.dumps(work_request).encode()

    # send the request to the worker
    r = requests.post(url, headers=headers, data=data, timeout=5)

    # return a status code
    return Response(status_code=status.HTTP_202_ACCEPTED)
```

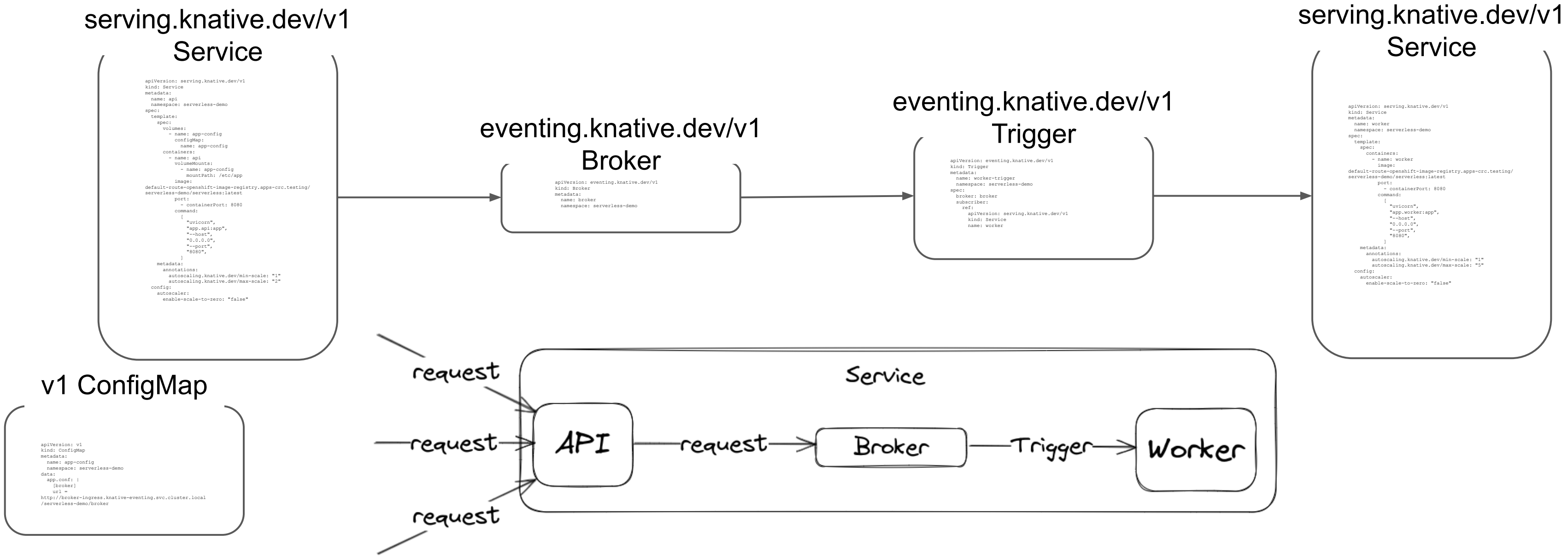
OpenShift Serverless demo: The application.

The **Worker** receives the work request from the broker, schedules it, and executes it.



```python
def do_the_work(work_info: dict):
    """Execute the work request."""

    logger.info("Executing work: %s", work_info["work"])

    # simulate the work execution
    time.sleep(work_info["duration"])

    logger.info("Work completed: %s", work_info["work"])


@app.post("/")
def schedule_the_work(work_info: dict, background_tasks: BackgroundTasks):
    """Schedule a new work request."""

    # execute the work in background
    background_tasks.add_task(do_the_work, work_info)

    # OpenShift Serverless needs an explicit return
    return Response(status_code=status.HTTP_202_ACCEPTED)
```

OpenShift Serverless demo: The deployment.



```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: api
  namespace: serverless-demo
spec:
  template:
    spec:
      volumes:
        - name: app-config
          configMap:
            name: app-config
      containers:
        - name: api
          volumeMounts:
            - name: app-config
              mountPath: /etc/app
          image:
default-route-openshift-image-registry.apps-crc.testing/
serverless-demo/serverless:latest
          port:
            - containerPort: 8080
          command:
            [
              "uvicorn",
              "app.api:app",
              "--host",
              "0.0.0.0",
              "--port",
              "8080",
            ]
    metadata:
      annotations:
        autoscaling.knative.dev/min-scale: "1"
        autoscaling.knative.dev/max-scale: "2"
  config:
    autoscaler:
      enable-scale-to-zero: "false"
```

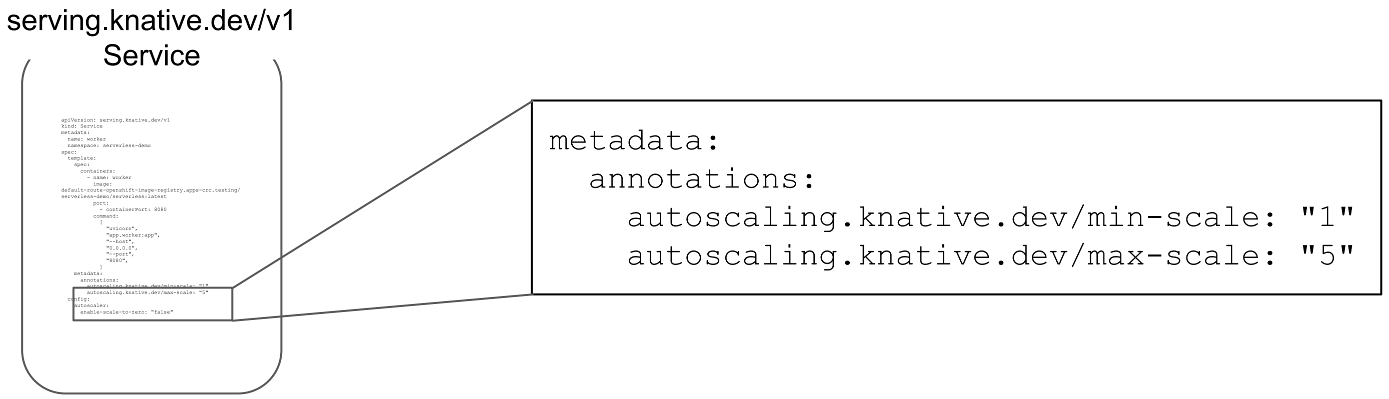
```yaml
apiVersion: eventing.knative.dev/v1
kind: Broker
metadata:
  name: broker
  namespace: serverless-demo
```

```yaml
apiVersion: eventing.knative.dev/v1
kind: Trigger
metadata:
  name: worker-trigger
  namespace: serverless-demo
spec:
  broker: broker
  subscriber:
    ref:
      apiVersion: serving.knative.dev/v1
      kind: Service
      name: worker
```

```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: worker
  namespace: serverless-demo
spec:
  template:
    spec:
      containers:
        - name: worker
          image:
default-route-openshift-image-registry.apps-crc.testing/
serverless-demo/serverless:latest
          port:
            - containerPort: 8080
          command:
            [
              "uvicorn",
              "app.worker:app",
              "--host",
              "0.0.0.0",
              "--port",
              "8080",
            ]
    metadata:
      annotations:
        autoscaling.knative.dev/min-scale: "1"
        autoscaling.knative.dev/max-scale: "5"
  config:
    autoscaler:
      enable-scale-to-zero: "false"
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
  namespace: serverless-demo
data:
  app.conf: |
    [broker]
    url =
http://broker-ingress.knative-eventing.svc.cluster.local
/serverless-demo/broker
```

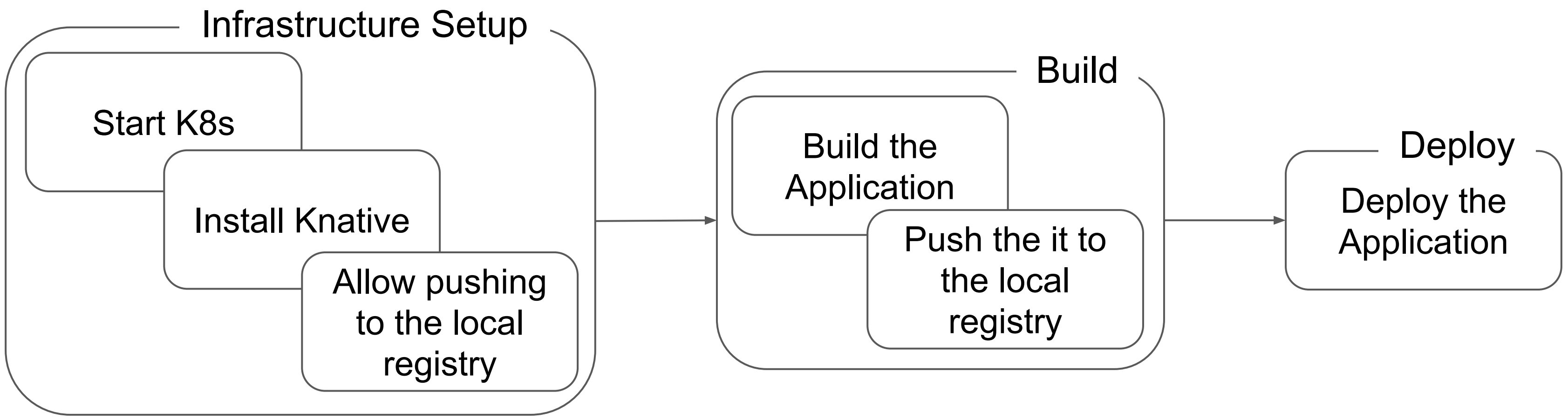
OpenShift Serverless demo: The deployment.



```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: worker
  namespace: serverless-demo
spec:
  template:
    spec:
      containers:
        - name: worker
          image:
default-route-openshift-image-registry.apps-crc.testing/
serverless-demo/serverless:latest
          port:
            - containerPort: 8080
          command:
            [
              "uvicorn",
              "app.worker:app",
              "--host",
              "0.0.0.0",
              "--port",
              "8080",
            ]
    metadata:
      annotations:
        autoscaling.knative.dev/min-scale: "1"
        autoscaling.knative.dev/max-scale: "5"
  config:
    autoscaler:
      enable-scale-to-zero: "false"
```

```
metadata:
  annotations:
    autoscaling.knative.dev/min-scale: "1"
    autoscaling.knative.dev/max-scale: "5"
```

OpenShift Serverless demo: Steps.



OpenShift Serverless demo: Build and Push.

Build and push video.

OpenShift Serverless demo: Deploy.

Deploy video.

OpenShift Serverless demo: Test.

Test the application video.

Red Hat

OpenShift Serverless demo: Flood.

Flood the application video.

Thank you!

Demo source: https://github.com/willianrampazzo/serverless-demo