T5

DevOps & The Cloud

2019-05-02 09:45

Leveraging Kubernetes as a Tester

Presented by:

Glenn Buckholz

Coveros

Brought to you by:

888-268-8770 · 904-278-0524 - info@techwell.com - http://www.stareast.techwell.com/

Glenn Buckholz

Glenn Buckholz leads CI and deployment automation efforts at Coveros. Using his more than fifteen years of industry experience, Glenn brings success to his customers. Beginning his career as a consultant implementing automated test frameworks, he introduced the concept of change management to many, many projects. After moving on from consulting, Glenn settled down at the Public Company Accounting Oversight Board as their full-time enterprise change manager. He eventually ventured back into the real world at Coveros, where he specializes in implementing agile practices, implementing CI, and engineering configuration management instead of simply documenting it.

coveros

Agility. Security. Delivered.

Leveraging Kubernetes as a Tester

Glenn Buckholz
E-mail: glenn.buckholz@coveros.com
Coveros, Inc.

Agenda

coveros

- Intro
- What is Kubernetes?
- Docker Recap
- Kubernetes Overview
- That Sounds Complicated How Can I Haz Kubernetes
- Ok Soooo What Does This Have To Do With Testing?
- Manual Testing
- Automated Testing
  - Forcing Some of Your Tests to Run at Deployment
- Kubernetes Specific Testing
- Questions

Introduction

AS A: Tester
I WANT: An accurate representation of the application in as it will be in production.
SO THAT: I can accurately gauge the quality of the next release.

AS A: Tester
I WANT: My most critical tests to gate any deployment
SO THAT: I can prevent builds with critical errors from ever reaching production or being promoted to testing environments.

What is Kubernetes?

Kubernetes - Scalable, production grade container orchestration with automated deployment, scaling, and management capabilities.

- This sounds like an OPS thing, how is it relevant to testing?
  - Setting it up and maintaining it is an OPS thing. It is a tool that you can very effectively add to your testing tool box.
- What do I need to know about containers?
  - As a tester, you do not needs to know the gory details just how they can help run your tests and what they mean for bug reports (What version am I testing?).

Docker Recap

- Docker ?!
  - If kubernetes orchestrates and coordinates Docker, first you need to know what is Docker.
  - Where virtualization multiplexes (shares) hardware docker gives processes the ability to share a single operating system in isolation, which is less overhead.
  - Image - This is the static part of docker, it is the definition of a container that has the filesystem and a specified entrypoint (the process to be run)
  - Container - This is an active image with a running process, and unless otherwise defined everything in the container is ephemeral.
    - Specified by <FQDN of REPO>/name:label
    - dockerhub.com/nginx:stable

Docker Recap (Cont)

It all goes away when the process crashes?

- Persistence is achieved in one of two ways:
  - Database backend - the process commits everything meaningful to a database.
  - Volume mounts - the host system shares its persistent filesystem as a mount point onto the ephemeral Copy On Write (COW) filesystem of the container. This allows data to persist on the host between invocations of the same image.

Docker Recap (Cont)

Why do I as a tester care about persistence?

- You need logs to correlate your bug reports with system errors.
- If you run tests in a container you need a place to store your results.
- Need to easily get the system back into an error state for troubleshooting.

Kubernetes Overview

- The application is defined in YAML
  - The relationship between all the containers
  - The network layer
    - Internal connections between container
    - Exposed external endpoints
  - How and what to scale
  - Resource monitoring
  - Self healing- Containers restart when they die
  - Hooks for initialization and timed jobs
  - Namespaces separate different applications
  - RBAC for both users and applications

Kubernetes Overview (Cont.)

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  ports:
    - port: 80
  selector:
    app: wordpress
    tier: frontend
  type: LoadBalancer
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

```
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
      - image: wordpress:4.8-apache
        name: wordpress
        env:
        - name: WORDPRESS_DB_HOST
          value: wordpress-mysql
        - name:
```

```
WORDPRESS_DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        ports:
        - containerPort: 80
          name: wordpress
        volumeMounts:
        - name: wordpress-persistent-
storage
          mountPath: /var/www/html
      volumes:
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: wp-pv-claim
```


That Sounds Complicated How Can I Haz Kubernetes?

- **MicorK8s,** Minikube, Kubeadmin-dind, Minishift, Docker Desktop - All local solutions
- Separate namespace on the company cluster with the proper RBAC permissions (Lamborghini Solution)
  - Can be cost effective depending on the application
- Namespace in a Turnkey cloud solution like EKS from Amazon

That Sounds Complicated How Can I Haz Kubernetes?

Demo - Sample App

Ok Soooo What Does This Have To Do With Testing?

- I was just showing you the hammer, now I will teach you how to drive in the nail.
- Testers can have personal environments to troubleshoot specific issues or run destructive test cases
- If we make the developers create docker containers with sane labels, you can test any version a developer has created.
- You can help troubleshoot issues in real time. Developers can create releases for you and only you to verify the fix before submitting to the CI/CD pipeline and making a formal release.

Ok Soooo What Does This Have To Do With Testing?

- Very fast response cycle between developers and testers.
- Environment should be production like in all aspects except space and performance.
- You can use docker and kubernetes nomenclature to be explicit about what version of the code is broken.

Manual Testing

- With a reasonably sized equipment I can now fit an entire application on my personal work machine.
- This will give me access to all aspects of the system:
  - Database / Logs / Configurations
- Developers must keep to a standard docker naming convention that is accepted by testers, developers, and operations.
- I can freeze system state and execute test cases that may be hard to prepare over and over.
- Simulate component failure scenarios locally without operations.
- Have an exact copy of production to troubleshoot with production data.

Automated Testing

- Tests can be packaged into a docker container and versioned with the code.
  - TDD can be used to gate container promotion
  - Test containers can be blocking or non-blocking to the release.
- Many different versions can be tested in parallel since environments are throw away and cheap.
  - Development speed is only limited by development and testing resources NOT environments.
- Critical tests can be burned into the container startup as probes or Jobs gating a deployment or release.

AUTOMATED TESTING CAN BE A MANDATORY PART OF THE DEPLOYMENT!

Automated Testing

Demo - Testing From Containers

Manual Testing

Demo - Fast Feedback Loops

Automated Testing

Demo - Building Tests into The Deployment

Docker/Kubernetes Specific Testing

- Testing the Containers
  - Twistlock/Nexus - Scanning the containers or LINT for docker
- Testing the Kubernetes Configuration
  - Can all the pieces talk to each other?
  - Write the liveness and readiness tests in the YAML definitions (Who better than a tester to write a test to see if a component is active and ready?)
  - Testing the scaling (When more containers spin up)
  - What happens when a node dies?
  - Testing the self healing (What happens when a container restarts)
  - Is persistent storage behaving as expected?

What are My Takeaways as a Tester?

- Kubernetes is a tool, not a DevOps tool and it can easily be used to benefit the testing effort of any software project.
- Environments are not a limitation, everyone can have one.
- CM is key, now that everyone has an environment I need to know exactly what I am testing.
- Destructive test cases can be developed without fear of impacting other testers.
- As a tester I need to know how to access the logs and diagnostic pieces of the application, since now I am allowed to have access to it.

Conclusions

- Shared environments will not go away the release needs to converge somewhere.
- Testing can become an impartial automated part of the deployment process.
- Testers and developers should work hand and hand now that they can easily exchange precise information and new builds.
- Testing production issues faithfully should only be as difficult as getting production data. The environment should no longer be a variable.
- Kubernetes itself is a testable thing and introduces new manual and automated test cases that need to be considered.

Questions?