ORACLE
Cloud Native Labs

Container Native Development Tools Compared: Draft, Skaffold, and Tilt

Mickey Boxell – Oracle Cloud Native Labs

#OracleCloudNative
cloudnative.oracle.com

ORACLE Cloud Native Labs

Who am I?

Mickey Boxell

Cloud Advocate, etc.

Oracle Cloud Native Labs

Share best practices and build original solutions and content for Oracle Cloud developers with a key focus on cloud native/container native, open source, and DevOps

http://cloudnative.oracle.com/

ORACLE Cloud Native Labs

Microservice Environments

- Distributed
- Container-based
- Polyglot
- Highly-scalable
- Ephemeral

ORACLE Cloud Native Labs

Development Workflow

- Step 1: Write code
- Step 2: Build code
- Step 3: Run code
- Step 4: Identify issues and return to Step 1

Container Native Development Workflow

- Step 1: Write code
- Step 2: Build code

Step 2.1: Build a container image

Step 2.2: Push the image to a registry

- Step 3: ~~Run code~~ Deploy to Kubernetes cluster
- Step 4: Identify issues and return to Step 1

ORACLE Cloud Native Labs

Traditional Deployment: Helidon/Java

```
$ mvn archetype:generate -DinteractiveMode=false \
        -DarchetypeGroupId=io.helidon.archetypes \
        -DarchetypeArtifactId=helidon-quickstart-se \
        -DarchetypeVersion=1.1.1 \
        -DgroupId=io.helidon.examples \
        -DartifactId=helidon-quickstart-se \
```

ORACLE Cloud Native Labs

Traditional Deployment: Helidon/Java

```
$ cd helidon-quickstart-se

$ mvn package

$ java -jar target/helidon-quickstart-se.jar
```

Container Native Deployment: Helidon/Java

```
$ docker build -t helidon-quickstart-se .

$ docker run --rm -p 8080:8080 helidon-quickstart-se:latest
```

ORACLE Cloud Native Labs

Local Kubernetes Cluster Deployment: Helidon/Java

```
$ kubectl apply -f app.yaml
```

Remote Kubernetes Cluster Deployment: Helidon/Java

```
$ docker tag \ helidon-quickstart-se:latest \ <region-code>.ocir.io/<tenancy-name>/<repo-name>/<image-name>:<tag>
$ docker push \ <region-code>.ocir.io/<tenancy-name>/<repo-name>/<image-name>:<tag>
$ kubectl apply -f app.yaml*
```

* modified with a container image matching the registry

ORACLE Cloud Native Labs

The Whole Flow

Step 1: Write code

Step 2: Build code AND build the image AND push the image to a registry

```
$ mvn package
$ docker build -t helidon-quickstart-se .
$ docker tag \ helidon-quickstart-se:latest \ <region-code>.ocir.io/<tenancy-name>/<repo-name>/<image-name>:<tag>
$ docker push \ <region-code>.ocir.io/<tenancy-name>/<repo-name>/<image-name>:<tag>
```

Step 3: Deploy to Kubernetes Cluster

```
$ kubectl apply -f app.yaml
```

ORACLE Cloud Native Labs

That seems like a lot of typing

ORACLE Cloud Native Labs

Of the same set of commands

ORACLE Cloud Native Labs

Over and over

ORACLE Cloud Native Labs

Why Did I Care?

- Simple code changes took too much time & too many keystrokes
- e.g. Was my endpoint zipkin.monitoring:9411 or 10.0.32.4:9411/zipkin or something else?
- Each change required me to: build code, build image, tag image, push image, apply manifest

When Does This Take Place?

- The inner loop of the container native development workflow: the period of time during which you are writing code, but have not yet pushed it to a version control system
- More simply: “when you’re iterating on code pre-commit”

ORACLE Cloud Native Labs

Why Deploy To A Cluster?

- Run integration and dependency tests
- Run diagnostic tools – logging, tracing, etc.

Why Deploy To A Remote Cluster?

- Match test environment to production environment
- Compliance – not everyone has access to a local cluster

ORACLE Cloud Native Labs

There’s even more going on under the covers

ORACLE Cloud Native Labs

Dockerfile

```
# 1st stage, build the app
FROM maven:3.5.4-jdk-9 as build
WORKDIR /helidon
# Create a first layer to cache the "Maven World" in the local
repository. Incremental docker builds will always resume after
that, unless you update the pom
ADD pom.xml .
RUN mvn package –DskipTests
# Do the Maven build! Incremental docker builds will resume here
when you change sources
ADD src src
RUN mvn package –DskipTests
RUN echo "done!”
```

```
# 2nd stage, build the runtime image
FROM openjdk:8-jre-slimWORKDIR /helidon
# Copy the binary built in the 1st stage
COPY --from=build /helidon/target/helidon-quickstart-se.jar ./
COPY --from=build /helidon/target/libs ./libsCMD ["java", "-jar",
"helidon-quickstart-se.jar"]
```

ORACLE Cloud Native Labs

Build, Push, Deploy Tools

What Are These Tools?

- Draft by Microsoft Azure
- Skaffold by Google
- Tilt by Windmill Engineering

ORACLE Cloud Native Labs

What Do These Tools Do?

- Build code
- Build an image of your project
- Push the image to a registry service of your choice
- Deploy the image onto a Kubernetes cluster
- And they are all open source

Pre-Requisites

- Docker
- A Kubernetes cluster
  - Docker For Desktop/Minikube
  - Oracle Container Engine for Kubernetes (OKE)
- Kubectl
- An image registry service
  - Oracle Cloud Infrastructure Registry (OCIR)

ORACLE Cloud Native Labs

Sample Application

helidon.io

- Helidon Framework
  - Java libraries for writing microservices
- Quickstart-SE sample application

ORACLE Cloud Native Labs

Draft

ORACLE Cloud Native Labs

Draft

- Low barrier to entry: Draft packs
    - draft create: boilerplate artifacts to run existing apps in K8s
- Uses Helm

Using Draft

Pre-Reqs: Docker, Kubectl, Helm

- draft init – install packs/plugins and configure $DRAFT_HOME
- draft create – create boilerplate based on application language
- draft config set registry phx.ocir.io/oracle-cloudnative/draft - creates .draft file
- docker login
- draft up + draft delete - make registry public or use imagepullsecrets

ORACLE Cloud Native Labs

Using Draft

- Port forward: draft connect
- Logs: draft logs

Draft

- Boilerplate is helpful to get started
- No watch/continuous deployment feature
- Helm can be overly-complicated
    - Didn’t use the ports set in app.yaml because of the Helm chart/values.yaml

Skaffold

ORACLE Cloud Native Labs

Skaffold

Flexible

- Many build options (Dockerfile locally, Dockerfile in-cluster with Kaniko, Dockerfile on the cloud, Jib Maven/Gradle locally, etc.)
- Many deploy options (kubectl, Helm, Kustomize)
- Many image tag policies

ORACLE Cloud Native Labs

Using Skaffold

Pre-Reqs: Docker, Kubectl

- vi skaffold.yaml – specifies workflow steps
- skaffold config set default-repo phx.ocir.io/oracle-cloudnative/skaffold – creates .skaffold file
- docker login
- skaffold run + skaffold delete or skaffold dev - make registry public or use imagepullsecrets + change image spec in app.yaml

Using Skaffold

Logs: skaffold run –tail

Port-forward: automatic based on pod spec configuration

Skaffold

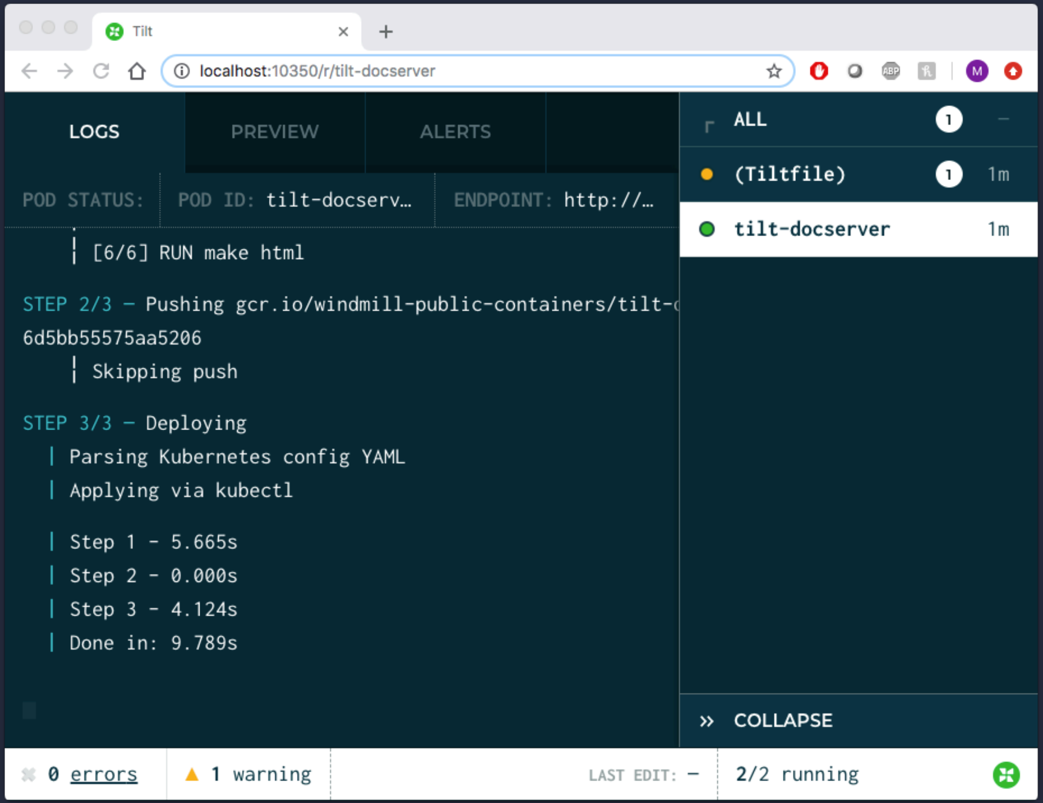
- Profiles feature
  - A set of settings stored in skaffold.yaml that overrides the build, test, and deploy sections of your current configuration
  - skaffold run -p [PROFILE]
- Deploy multiple microservices at once – referenced in skaffold.yaml
- Deploy once with skaffold run or continuously with skaffold dev

Tilt

Cloud Native Labs

Tilt

- Heads up display and browser UI



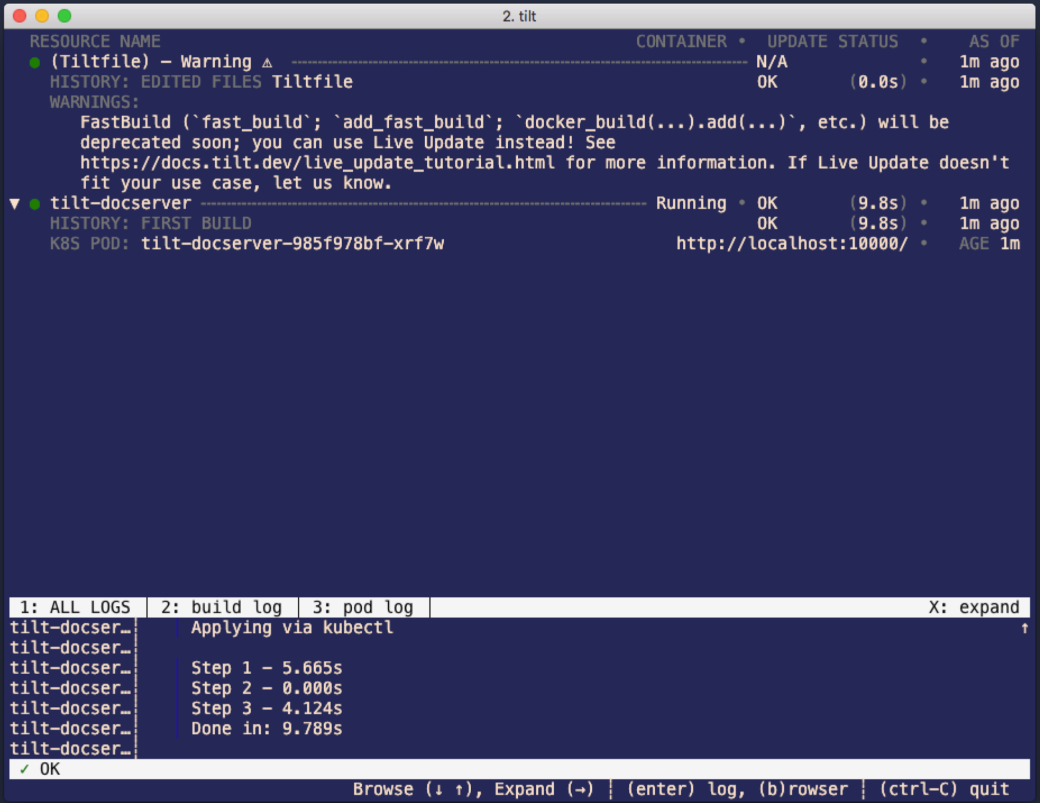
```
POD STATUS:   POD ID: tilt-docserv…   ENDPOINT: http://…
    ┆ [6/6] RUN make html

STEP 2/3 — Pushing gcr.io/windmill-public-containers/tilt-c
6d5bb55575aa5206
    ┆ Skipping push

STEP 3/3 — Deploying
  | Parsing Kubernetes config YAML
  | Applying via kubectl

  | Step 1 - 5.665s
  | Step 2 - 0.000s
  | Step 3 - 4.124s
  | Done in: 9.789s

0 errors   1 warning   LAST EDIT: —   2/2 running
```



```
2. tilt
 RESOURCE NAME                                                CONTAINER •  UPDATE STATUS  •    AS OF
 ● (Tiltfile) — Warning ⚠ ------------------------------------------- N/A                •   1m ago
   HISTORY: EDITED FILES Tiltfile                                       OK        (0.0s) •   1m ago
   WARNINGS:
      FastBuild (`fast_build`; `add_fast_build`; `docker_build(...).add(...)`, etc.) will be
      deprecated soon; you can use Live Update instead! See
      https://docs.tilt.dev/live_update_tutorial.html for more information. If Live Update doesn't
      fit your use case, let us know.
▼ ● tilt-docserver ------------------------------------------ Running • OK        (9.8s) •   1m ago
   HISTORY: FIRST BUILD                                                 OK        (9.8s) •   1m ago
   K8S POD: tilt-docserver-985f978bf-xrf7w                   http://localhost:10000/ •   AGE 1m

 1: ALL LOGS │ 2: build log │ 3: pod log │                                                X: expand
tilt-docser…┆    │ Applying via kubectl                                                          ↑
tilt-docser…┆
tilt-docser…┆    │ Step 1 — 5.665s
tilt-docser…┆    │ Step 2 — 0.000s
tilt-docser…┆    │ Step 3 — 4.124s
tilt-docser…┆    │ Done in: 9.789s
tilt-docser…┆
 ✓ OK
                           Browse (↓ ↑), Expand (→) ┆ (enter) log, (b)rowser ┆ (ctrl-C) quit
```

ORACLE Cloud Native Labs

Using Tilt

Pre-Reqs: Docker, Kubectl

- vi Tiltfile – specifies workflow steps
- Set registry in the Tiltfile or tilt_option.json
- docker login
- tilt up + tilt down - make registry public or use imagepullsecrets + change image spec in app.yaml

ORACLE Cloud Native Labs

Using Tilt

- B opens a port forward based on Tiltfile resource URL
  - Browser UI includes resource preview page
- Logs available on the UI – X to expand logs

ORACLE Cloud Native Labs

Tilt

- Heads up display and browser UI
- Support for Helm
- LiveUpdate: update a running container in place
  - Instead of building a new image and redeploying from scratch
- Deploys multiple microservices - sample application “servantes”
- No single deploy option

ORACLE Cloud Native Labs

Differentiators

DRAFT

SKAFFOLD

TILT

- Getting started boilerplate
- Flexibility
- Heads up display

Key Takeaways

- These tools automate away countless manual steps
- These tools can deploy to both local and remote clusters
  - The registry step can be bypassed for local clusters
- Useful as a step before pushing to source control and/or CI

ORACLE Cloud Native Labs

Stay Connected

Medium: https://medium.com/@m.r.boxell

Twitter: @mickeyboxell

Linkedin: https://www.linkedin.com/in/mickeyboxell/

Try Oracle Cloud: https://cloud.oracle.com/tryit

#OracleCloudNative
cloudnative.oracle.com

ORACLE Cloud Native Labs