Benefits of Eclipse Che When Developing Multi-Container Apps

How to use existing production k8s environment in enterprise development

Eugene Ivantsov
Senior Software Maintenance Engineer
Red Hat
AGENDA:
This presentation covers:

- Introduction to Eclipse Che
- Cloud native development challenges
- Solution Eclipse Che offers
- Demo: deploying a microservice app to a k8s cluster, developing in Che using the same yaml definitions, updating deployments in k8s cluster
What Is Eclipse Che?

**Open Source**
Eclipse Foundation project since 2016. ~4.5k stars, ~100 releases, ~90 contributors, ~800 forks

**Workspace server**
Runs on and supports OpenShift, Kubernetes and Docker as runtime engines. New infras can be plugged in

**Cloud IDE**
Code, compile, run debug and collaborate on projects in your browser. GWT and TypeScript IDEs are available
The Problem

Containerized production

PRODUCTION DEPLOYMENTS ARE WELL DEFINED AND PREDICTABLE
Containers behave in the same way anywhere they are run

VS

localhost development

LOCALHOST ENVIRONMENT NEVER REPLICATES PROD ENVIRONMENTS
Something that works on localhost may fail in containerized production env

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  labels:
    app: microservice-app-example
  name: users-api
spec:
  replicas: 2
template:
  metadata:
The Solution
Develop in your production!

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  labels:
    app: microservice-app-example
  name: users-api
spec: 
  replicas: 2
template:
  metadata:

CHE RUNS PROD IN DEV MODE

Brings IDE tooling, source code and data management on top of prod yaml definition
How It Works
Che workspace engine magic:

Prod containers defined in env recipe

- IDE
- Tooling
- Build
- Shared volumes
- Services
- Separate build and run
- Remote debug
Workspace Environment

- Consist of containers, single/multi-container pods
- Defined by a recipe: k8s yaml, docker image, composefile
- Shared volumes to exchange source code changes and artifacts
- Servers to expose services (publishing ports, OpenShift routes, k8s ingress etc)
IDE

- Fast JavaScript IDE
- Client-Server architecture
- Pluggability
- Plugin registry
- Custom IDE per workspace
- Use of various IDEs (e.g. Theia [https://github.com/theia-ide/theia](https://github.com/theia-ide/theia))
Tooling

- LSP support
- Local LS mode
- Language server as side-cars (TCP mode, shared volumes)
- Registering new Language Servers
- Terminal and exec agent sidecars
Build Containers

- Use custom build images (e.g. the ones used in CI)
- Shared volumes - build artifacts available in run containers
- Resource management - RAM allocation for a particular project
- One service per container
Cloud Native Development with Che

Production → Che Workspace → CI/CD
Demo Time

What will we see?

- Deploy microservice app to MiniKube
- Push to GitHub
- Develop in Che
- CI job triggered
Get Started With Eclipse Che

Sources
https://github.com/eclipse/che

Dev List
che-dev@eclipse.org

Docs
https://eclipse.org/che/docs

Mattermost
https://mattermost.eclipse.org/