# Helsinki Software QA and Testing Meetup

January 26th 2017 @ Unity Helsinki

# Agenda - approx. times

18.00 - Welcome. Thoughts and ideas for this QA Meetup group

18.15 - Bitbar monitoring (Bitbar)

18.45 - Using Docker and Docker Compose for testing (Unity)

19.45 - Quick evaluation and feedback

Afterwards grab a drink and network with other attendees

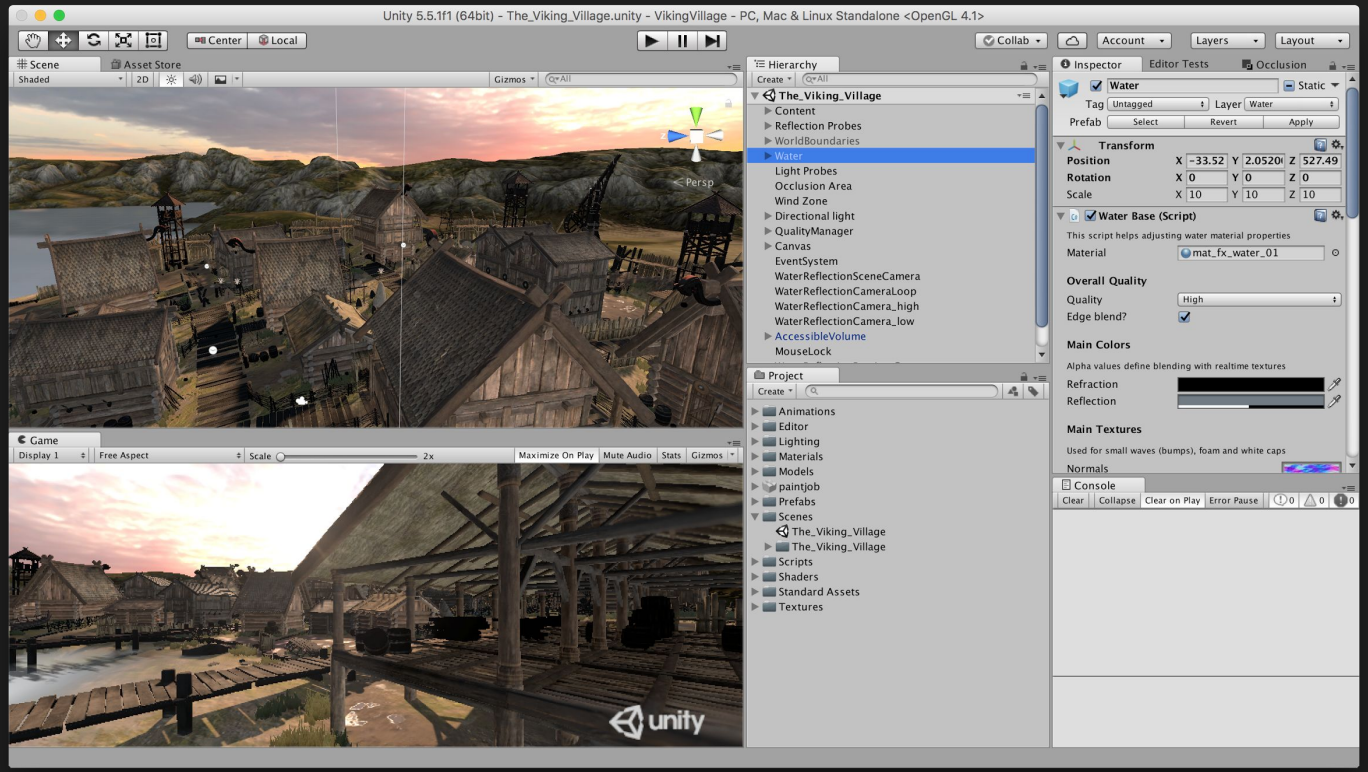
# Helsinki Software QA and Testing Meetup

- Motivations for this meetup
- Focus on technical aspects of software testing and automation

# Bitbar Monitoring

# Docker for Testing

# Unity Engine



# Unity Helsinki

Developing infrastructure and SDK to help game developers acquire users and make a sustainable business on their games.

Ads :)

# The Testing Challenge

**Papa Devops**
@stahnma

Follow

Everybody has a testing environment. Some people are lucky enough enough to have a totally separate environment to run production in.

| RETWEETS | LIKES |
| --- | --- |
| 96 | 103 |

3:07 PM - 21 Aug 2015

96 103

https://twitter.com/stahnma/status/634849376343429120

# Docker - a quick overview (probably skip this slide)

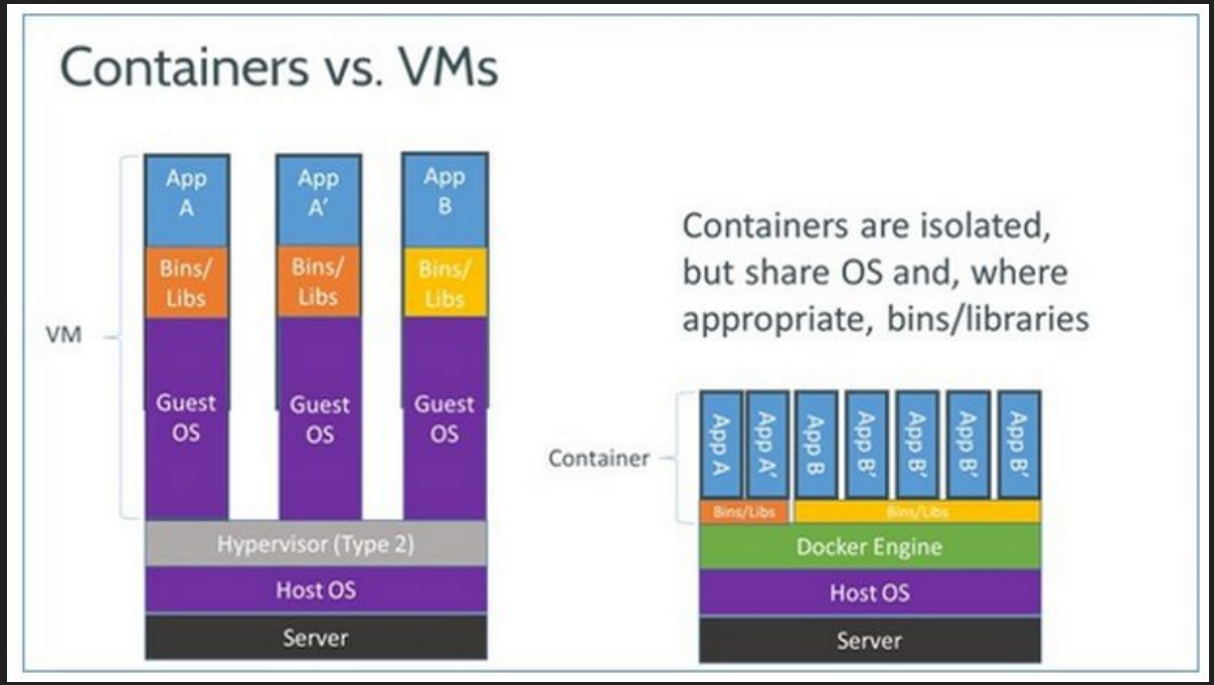
**Conceptually:**

- Similar to virtual machines
- Allows software to run in isolation

**On implementation level:**

- Not the full OS, only the “virtualized” components
- Faster to start. Virtual machines typically takes minutes, whereas docker container takes seconds (or even milliseconds)

# How is Docker different from Virtual Machines?




http://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/

# Docker Concepts

| | |
|---|---|
| Image | Read-only template used for building containers. Either made from scratch or reused from others |
| Container | Running instance of a Docker image |
| Registry | Repository for Docker images. Docker hosts repository at https://hub.docker.com/, but you can also host your own |

# Docker - How to get started

- Download from https://www.docker.com/products/docker
- Go to https://docs.docker.com/engine/getstarted/

Or just google for “Docker”...

# Demo

A simple node.js http server running in Docker
(https://github.com/rasmusselsmark/DockerForTesting)

# Our node.js http server

```
1   var http = require("http");
2   var os = require("os");
3
4   var port = 8888;
5
6   http.createServer(function(request, response) {
7     response.write("Hello from " + os.hostname() + " running node " + process.version);
8     response.end();
9   }).listen(port);
10
11  console.log("Node.js http server running at port " + port);
```

# Running the node server locally

Run

```
$ node server/server.js
```

Open http://localhost:8888/

Should see e.g.:

```
Hello from (hostname) running node v6.9.2
```

# The Dockerfile

- Self-contained file describing dependencies and deployment of system under test

```
# our base image
FROM node:7.4

# environment
ENV WORK_DIR /DockerForTesting
WORKDIR $WORK_DIR

# files included in image/container
# as Docker caches the intermediate built images, order matters
COPY server/ $WORK_DIR/server/
COPY Dockerfile $WORK_DIR/
COPY Makefile $WORK_DIR/

# we're by default using port 8888
EXPOSE 8888

# the command invoked when calling `docker run`
ENTRYPOINT ["make", "run-server"]
```

# Building Docker image and running container

```
$ docker build --tag docker-for-testing-image .

$ docker run --name docker-for-testing --publish 8888:8888 docker-for-testing-image
```

Open http://localhost:8888. Should see something like:

```
Hello from 3191e282b178 running node v7.4.0
```

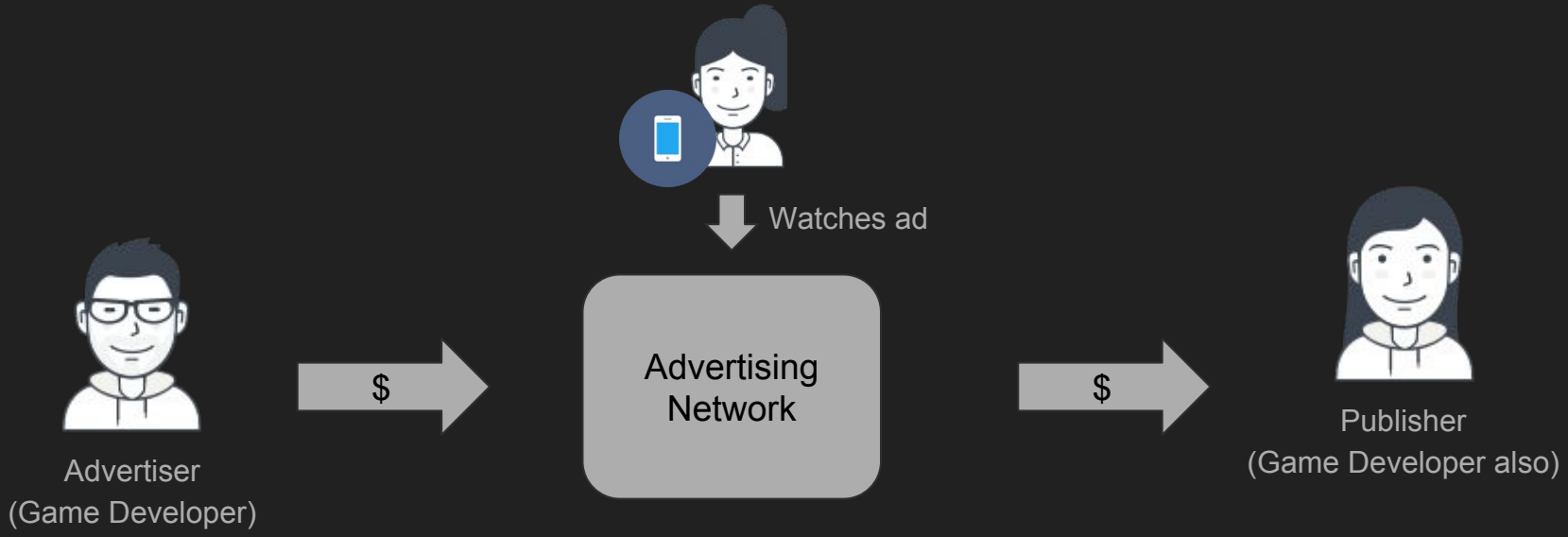
I.e. different node version inside docker container (running on same physical machine)

# Demo #1

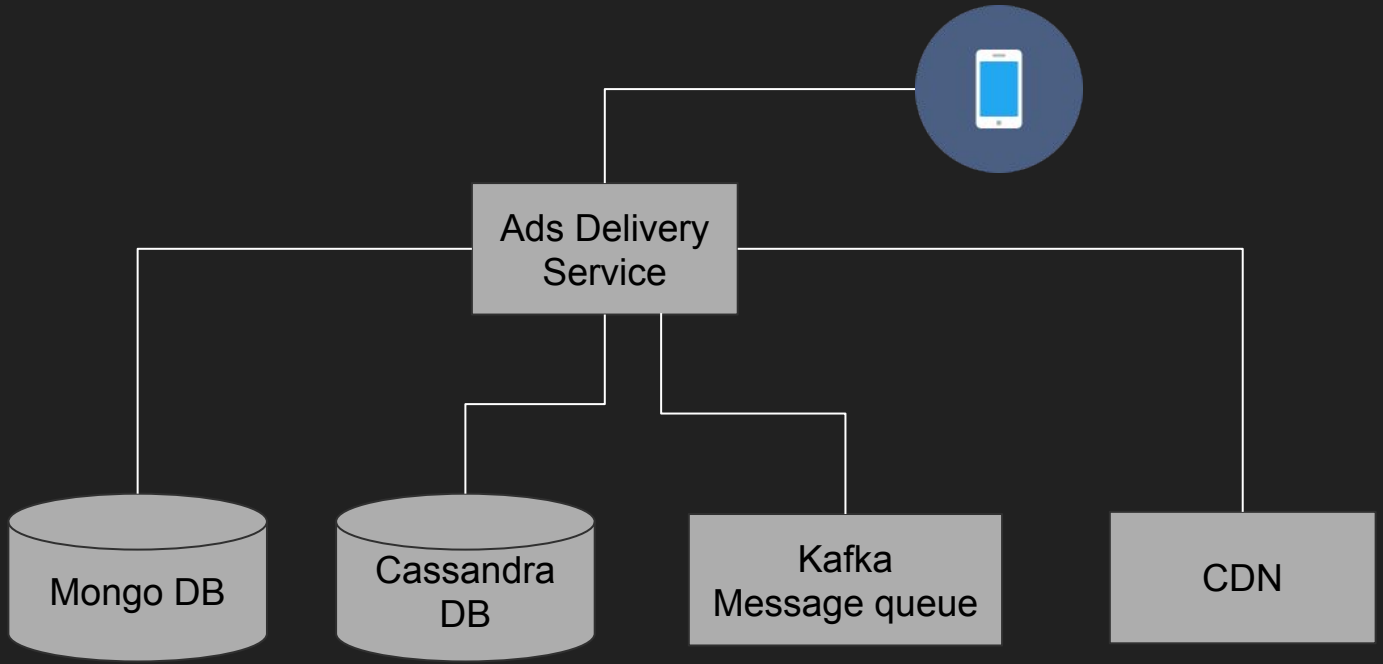
Docker and Docker Compose
for End-to-End testing

# Unity Ads - Advertising Network for games

- User Acquisition for game developers (advertisers)
- Help game developers make a sustainable business (publishers)
- Handling ~30K requests/events per second from games



# (Part of) our setup for testing



# Docker Compose

https://docs.docker.com/compose/overview/:

“Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a Compose file to configure your application’s services. Then, using a single command, you create and start all the services from your configuration.”

# (Part of) Docker-compose.yml file example

```
version: '2'
services:

  delivery:
    image: unity-registry-url/delivery:latest
    restart: always
    environment:
      - NODE_ENV=compose
      - LOG=true
      - KAFKA_HOST=kafka
    ports:
      - "3500:3500"
    depends_on:
      - cassandra-seed
    links:
      - cassandra
      - mongo
      - nats
      - redis
      - (...)
```

```
(...)

  ads-auth-session:
    build: ./ads-auth-session
    ports:
      - "3000:3000"
    environment:
      AUTH_HOST: (hostname)
```

# Demo #2

Running UI tests using Docker on Jenkins

# Protractor UI tests

**Goal:** Have tests running on Jenkins, without need for installing dependencies on the Jenkins node

Process:

- Google for “protractor in docker”
- Eventually find https://github.com/jciolek/docker-protractor-headless (simplest)
- Modify slightly and use :)

Result:

- Self-contained scripts allowing running protractor tests locally and on Jenkins

# End of demo - questions?

# Unity Helsinki Open House event

- February 16th 2017
- More info on the Unity Helsinki meetup group (https://www.meetup.com/Unity-Helsinki-Events/)

# Evaluation

- Feedback?
- Possible presentations for next time:
  - ELK-stack in testing (Elasticsearch, Logstash, Kibana) - by Anoop, Tuxera
  - Others?
- When?