Building a Global-Scale Multi-Tenant Cloud Platform on AWS and Docker: Lessons Learned

Felix Gessert, Florian Bücklers
{fg,fb}@baqend.com
@baqendcom
Baqend & Our Infrastructure

Docker Concepts

Clustering: AWS ECS vs. Docker Swarm
Presentation is loading
The Latency Problem

Average: 9.3s

Loading…
The Latency Problem

100 ms

Loading…

Average: 9.3s

-1%Revenue
The Latency Problem

400 ms

Loading…

Average: 9,3s

-9% Visitors

-1% Revenue

Yahoo!

Amazon.com
The Latency Problem

500 ms

-20% Traffic

-9% Visitors

-1% Revenue

Average: 9.3s

Loading…
If perceived speed is such an import factor

...what causes slow page load times?
State of the Art
Two bottlenecks: latency und processing
State of the Art

Two bottlenecks: latency und processing
State of the Art

Two bottlenecks: latency und processing
Problem: Network Latency

Page Load Time as bandwidth increases

- 1 Mbps: 3500 ms
- 2 Mbps: 2000 ms
- 3 Mbps: 1500 ms
- 4 Mbps: 1200 ms
- 5 Mbps: 1000 ms
- 6 Mbps: 800 ms
- 7 Mbps: 600 ms
- 8 Mbps: 400 ms
- 9 Mbps: 200 ms
- 10 Mbps: 100 ms

Page Load Time as latency decreases

- 200 ms: 3500 ms
- 180 ms: 3000 ms
- 160 ms: 2500 ms
- 140 ms: 2000 ms
- 120 ms: 1500 ms
- 100 ms: 1000 ms
- 80 ms: 500 ms
- 60 ms: 200 ms
- 40 ms: 100 ms
- 20 ms: 50 ms

Problem: Netzwerklatenz

$2 \times \text{Bandwidth} = \text{Same Load Time}$

$\frac{1}{2} \text{ Latency} \approx \frac{1}{2} \text{ Load Time}$
Low-Latency
Data is served by ubiquitous web-caches
Low-Latency
Data is served by ubiquitous web-caches
Low-Latency
Data is served by ubiquitous web-caches

Low Latency

Less Processing
Scaling
Scalable and highly available
Innovation

5 Years Research & Development

New Algorithms Solve Consistency Problem

Innovation Problem: Changes cause stale data
Innovation
Problem: changes cause stale data

5 Years
Research & Development

New Algorithms
Solve Consistency Problem
Innovation

Problem: changes cause stale data

5 Years
Research & Development

α
New Algorithms
Solve Consistency Problem
Innovation

Problem: changes cause stale data

5 Years
Research & Development

New Algorithms
Solve Consistency Problem
Innovation

Solution: Baqend proactively revalidates data

5 Years
Research & Development

α
New Algorithms
Solve Consistency Problem

Bloom filter

0 1 1 1 1 1 0 1

update
Innovation
Solution: Baqend proactively revalidates data

5 Years
Research & Development

New Algorithms
Solve Consistency Problem

Bloom filter
Is 📝 still fresh?
update

0 1 1 1 1 1 1 0 1
Innovation

Solution: Baqend proactively revalidates data


Page-Load Times
What impact does caching have in practice?
Page-Load Times

What impact does caching have in practice?
Page-Load Times
What impact does caching have in practice?
How is this used from a developer’s perspective?

Backend-as-a-Service
DB.Tankstellen.find()
  .near("location", myLoc, 5000)
  .lessThen("closing", time)
  .greaterThen("opening", time)
  .descending("price")
  .resultList();
Baqend Architecture
Our Infrastructure

Polyglot Storage
Baqend Architecture

Our Infrastructure

Database-as-a-Service Middleware: Caching, Transactions, Schemas, Invalidation Detection, ...
Baqend Architecture
Our Infrastructure

Standard HTTP Caching

- InvalidDB
- Expiring Bloom
- Reverse-Proxy Caches
- Orestes Servers
- Node.js
- TTL Estimator
- User-defined Business Logic

- Dynamic Web App
- Internet
- Desktop
- Mobile
- Tablet

- mongoDB
- redis
- elasticsearch

Content-Delivery-Network
Baqend Architecture
Our Infrastructure

Unified REST API
Baqend Architecture
Our Infrastructure

IaaS-Cloud on

Desktop
Mobile
Tablet

Content-Delivery-Network

InvalidDB Streaming Queries
Cache Lifetime Prediction
Stale Data
User-dep Business Logic

Reverse-Proxy Caches
mongodb
Redis

web services

amazon

elasticsearch
Baqend Architecture
Our Infrastructure

CDN on

IaaS-Cloud on

Desktop
Mobile
Tablet
Baqend Architecture
Our Infrastructure

CDN on fastly
IaaS-Cloud on Amazon Web Services
AWS Services

Services we use

- Route 53, EC2, ASGs, IAM etc.
- **Elastic Load Balancer**: TCP Balancing for Logging
  - Not suited for multi-tenant SSL termination: ELB cannot dynamically route to an IP:port pair
- **Redis ElastiCache**: Metadata Storage
  - Easy to use but very limited: no Redis cluster support, no append-only files, bad snapshotting
- **What we don’t use:**
  - **Beanstalk**: supports Docker but needs a dedicated EC2 instance
  - **Cloudfront**: useless invalidations, expensive
  - **DynamoDB**: difficult to scale, very limited queries
Containerization
Why we need containers & cluster management

- Every tenant needs a private JVM and Node.JS process

Diagram:
- Baqend Server
- Customer’s Business Logic
Containerization
Why we need containers & cluster management

- Every tenant needs a private JVM and Node.JS process
- Provisioning new instances needs to be fast & easy:

![Diagram](image-url)

- Launch App → BBQ Manager → Configure databases, CDN, etc. → Start
Problem: Many Technology Choices
Emerging Frameworks and Tools

- Cluster Managers & Orchestration Tools:
  - Google Kubernetes
  - Apache Mesos
  - Docker Swarm
Problem: Many Technology Choices

Emerging Frameworks and Tools

- Cluster Managers & Orchestration Tools:
  - Google Kubernetes
  - Apache Mesos
  - Docker Swarm

- Container Cloud Platforms:
  - Amazon Elastic Container Service
  - Tutum
  - Google Container Engine
  - Rancher
Problem: Many Technology Choices
Emerging Frameworks and Tools

- Cluster Managers & Orchestration Tools:
  - kubernetes
  - MESOS
  - Google Kubernetes Engine
  - Amazon Elastic Container Service
  - Azure Container Service (Microsoft)
  - Nomad (HashiCorp)
  - Diego (Cloud Foundry)
  - Fleet (CoreOS)
  - ContainerShip
  - YARN (Hadoop)
  - and many more: Tutum, Rancher
Live Demo: Launching a container
Docker Concepts

What is Docker?

- Docker typically isolates a **single application**
- An application is built into a **Docker image** (including the OS)
- The docker image can be hosted and transferred to different hosts (**Docker Registry**)
- The docker image can be executed as a new container on any machine that runs a **Docker daemon**
- **Updates** are handled by just stopping and starting a new container

Source: https://docs.docker.com/engine/introduction/understanding-docker/
Docker Architecture

How to set up a Docker host

- Docker runs on all common **Linux** distributions
- Docker can be installed from Docker’s own package repository
- The Docker daemon can be configured by editing `/etc/default/docker`
- The Docker daemon allows many useful configurations:
  - Inter-container communication
  - Docker remote REST API
  - Labeling
  - DNS configuration
  - IP forwarding (disables internet for containers)
  - SSL encryption for the Docker daemon
The Dockerfile
How to build a Docker image

FROM ubuntu:latest

ENV DEBIAN_FRONTEND noninteractive

# java
RUN apt-get install -y software-properties-common && \
    add-apt-repository -y ppa:webupd8team/java && \
    apt-get update && \
    echo debconf shared/accepted-oracle-license-v1-1 select true \ 
        | debconf-set-selections && \
    apt-get install -y oracle-java8-installer

# extract and install packages
ADD baqend-package*.tgz /opt
ADD config.json /opt/baqend/

EXPOSE 8080

WORKDIR /opt/baqend/

ENTRYPOINT ["java", "-classpath", "/opt/baqend/lib/*", "info.orestes.Launcher"]
CMD ["--config", "config.json"]
How a Docker container works
Isolation, performance, light-weight

- **Filesystem**: by using multiple read-only file systems and mounting a read-write file system on top
- **Data volumes**: mount additional physical volumes into the container
- **CPU**: by CPU shares and core limitation
- **Memory**: by defining memory constraints
- **Network**: by using virtual networks
- **System privileges**: such as port binding, execution rights, inter process communication, etc.
- **Logging**: by using docker logging capabilities or external loggers (json, syslog, aws, etc...)
Docker Options
Imposing constraints on containers

- Most constraints are set when the container is started

--add-host=[] Add a custom host-to-IP mapping (host:ip)
--cpu-shares=0 CPU shares (relative weight)
--cpu-quota=0 Limit CPU CFS (Completely Fair Scheduler) quota
-e, --env=[] Set environment variables
-l, --label=[] Set metadata on the container (e.g., --label=key=value)
--link=[] Add link to another container
--memory=TOTAL MEMORY Limit (memory + swap), '-' to disable swap
--mem-swappiness=0 Swap swappiness
default: 50
--name=container name Assign a name to the container
--net=network Connects a container to a network
'bridge': creates a new network stack on the docker bridge
'host': use the host network stack inside the container
'NETWORK': connects the container to user-created network
--oom-kill-disable=false Whether to disable OOM Killer for the container or not
-p, --publish=[] Publish a container's port(s) to the host
--read-only=false Mount the container's root filesystem as read only
--restart="no" Restart policy (no, on-failure[:max-retry], always)
-v, --volume=[] Bind mount a volume
Docker Networking
Making containers talk to each other

- Docker containers can talk to each other by default
- Communication between containers can be restricted by the daemon option: `--icc=false`
- Docker containers can discover other linked containers by their names

```
EXPOSE 8080
```

Port 8080 not published, (can’t be accessed from host or other containers)

```
docker run --name="orestes" orestes
docker run --link="orestes" node
```

Can access orestes:8080
Docker Networking
Making containers talk to each other

- Docker containers can talk to each other by default
- Communication between containers can be restricted by the daemon option: `--icc=false`
- Docker containers can discover other linked containers by their names

```bash
docker run --name="orestes" -p 0.0.0.0:80:8080 orestes
docker run --link="orestes" node
```
AWS provides ECS-optimized AMIs for simple deployment.

ECS manages EC2 instances by running an ECS Agent on each instance.

ECS can automatically deploy and scale new Docker containers specified by a Task definition across the ECS Cluster.
ECS: Tasks and Services

Defining groups of containers

- ECS groups containers into Tasks and deploys them together

- A Task definition describes:
  - The Docker images
  - Resource requirements
  - Environment variables
  - Network links
  - Data Volumes

- ECS Services can be used to keep a specified number of Tasks running

- ECS can autoscale a Service when it is used with an ELB
Limitations that AWS fixed

Old Docker, Parameterization

- ECS has used an outdated version of docker, now it’s 1.9, yeah!
- Tasks can now be parametrized using commandline args

- Previously only environment variables could be passed while starting a Task
- Environment variables are exposed to linked containers, this can be a security issue!

Secured Process

Can access env ORESTES_SECRET

Untrusted Process

Can access env ORESTES_SECRET

docker run --name="orestes" --env SECRET=7kekfjd9e
docker run --link="orestes" node

https://docs.docker.com/engine/userguide/networking/default_network/dockerlinks/#environment-variables
ECS uses hard memory constraints (\texttt{run -m}) for Tasks to schedule container placement.

- This allocates a \textbf{fixed amount of memory} on the EC2 instance and can’t be exceeded by the process.

- This is very ugly for shared, multi-tenant applications:
  - Setting the constraint too low causes Docker to kill the process on memory peaks.
  - Setting the value too high limits the number of containers that can be launched per EC2 instance.

- Neither Docker’s memory swapping nor unlimited memory usage is allowed by ECS.
Current Limitation: Networking
Docker’s new network API not supported

- Docker has introduced a new network API, which allows to create custom virtual networks
- **Bridge Networks** connect groups of containers together and isolate them from other groups on the same host
- **Overlay Networks** use a key-value store to connect containers across different host machines

Source: https://docs.docker.com/engine/userguide/networking/dockernetworks/
Wrap-up: ECS
Pros and Cons

- **Very simple setup**, thanks to the optimized ECS AMI
- **Task** abstraction makes it really comfortable to start multiple containers together
- **Services** ensures that the desired count of tasks are always up and running
- **Automatically starts new EC2 instances** if no capacity is left for new containers
- Can be combined with an ELB for a **high availability** setup

- **Many Docker options** aren’t available
- Service Tasks can’t be **parametrized**
- **Running the same Services** for different tenants on the same EC2 instance is not possible
- Only the **legacy networking** is supported
- **New features** will always be delayed since they must first be implemented in ECS
Docker Swarm
A replacement for ECS

- Docker Swarm is Docker’s **native solution** for cluster management
- Docker Swarm uses a **discovery service** to manage the shared state of the cluster
- The following backends for discovery are supported:
  - Docker Hub (for development only)
  - Static file
  - etcd
  - consul
  - zookeeper
  - IP list or a range pattern of IPs
Swarm Architecture
Cluster management with Docker Swarm

Docker Swarm Cluster
Docker Daemon
Swarm Agent
Expose 2375
Docker Daemon
Swarm Agent
Expose 2375
Docker Daemon
Swarm Agent
Expose 2375
ZooKeeper
Swarm Manager
Docker Client

Docker Swarm Cluster
Swarm is Docker
Fixing the shortcomings of ECS

- The Swarm manager acts as a **proxy** of the Docker Remote API
  - All Docker run options are available in Swarm, too
- Docker Swarm can be combined with **overlay networks**
  - Containers can connect to others by just using the containers name (**service discovery**)  
  - Works across Docker hosts, availability zones and external hosts
- Containers can use any other service without defining them in a group (such as a Task)
Autoscaling in Swarm

Scale-out and scale-in

- Docker hosts can be added and removed to the Swarm Cluster silently
- Swarm provides an API to gather CPU usage and memory consumption of hosts or containers
- Swarm provides no concept to scale services within containers
High Availability in Swarm
Handling failures and outages

- **Labeled** Docker daemons can be used by the manager to run specific containers only on specific hosts
- Containers can be launched:
  - On the same host where other containers are running
  - In a specific availability zone
  - On hosts with special capabilities (RAM, CPU or SSD)
- The Docker daemon can **restart** failed containers using a restart policy --restart="yes"
- Containers will also be restarted if the docker host restarts
- Failed machines must be handled manually
Securing Swarm Hosts

Security pitfalls

- Swarm requires that the Docker daemon is exposed via TCP
- In most setups this will be a security issue since you can easily get root permission on the Docker host
- Also containers can access the exposed API by default
- Therefore it is recommended to always secure the Docker daemons on each host with SSL
- Docker supports SSL client, server and both authentication mechanisms
- SSL server authentication is not very practical since it requires a signed certificate for each host
Securing a Swarm cluster requires signed SSL certificates on all docker hosts, on the swarm manager and the docker client.
Wrap-up: Docker Swarm

Pros and Cons

- **Swarm is Docker**, all Docker options are available
- **Labeling** Docker hosts, allows to deploy containers on specific hosts
- **Overlay Networks** allow containers to communicate across hosts
- **Service Discovery** across containers is made really simple

- **Complex setup** and many components are required for a complete setup
- No built-in way for **autoscaling** services
- **Still many bugs**
- The Docker Swarm **API integration** into Docker is not yet completed
Conclusions

ECS vs Swarm

- **Simple Setup**
  - Task and Service definition makes it easy to deploy and update containers
  - Detect failures and restart failed tasks within services
  - Integrated into other AWS Services such as Elastic Load Balancers and Auto Scaling Groups

- **Complex Setup**
  - Many configuration options for deploying containers
  - Is compatible to the Docker API, allows to use all Docker clients
  - Supports Docker’s network API
  - No Vendor Lock-In
Want to try Baqend?

- Download **Community Edition**
- Invited-Beta **Cloud Instance**
  - support@baqend.com
- **Baqend Cloud** launching this February