Contemporary Web Development
Lesson 11

https://www.youtube.com/user/Kosmicd12/

Web Hosting

Static VS Dynamic serving and the newer "Serverless" option

Static Websites

- A standard static website has no backend. Just a bunch of HTML files with javascript. It can still be very rich but has no "persistence", user authentication, backend logic, data etc. Useful for:
  - Portfolio
  - Blog
- That makes it super fast because in serving pages because there is no backend logic.
- Can use a limited amount of storage on the user's device via Cookies or the *LocalStorage* API.
- Can become more dynamic by using a *Static website generator* such as Jekyll or Hugo – The author updates metadata on their own PC and then publishes the newly generated HTML files.

A static website has no server logic – no authentication, no backend, no data is saved remotely. all information is available on the client. It can still be very rich.

Can also access remote APIs, but only with the client's credentials.

Can even save a local state in the browser "Localstorage" but there's a limit to how much you can save.

Hosting for Static websites

- Github pages
- You own computer / Raspberry Pi clone (I use this).
- Peer to Peer hosting! Such as Hashbase or Cloudflare's Distributed Web Gateway on IPFS

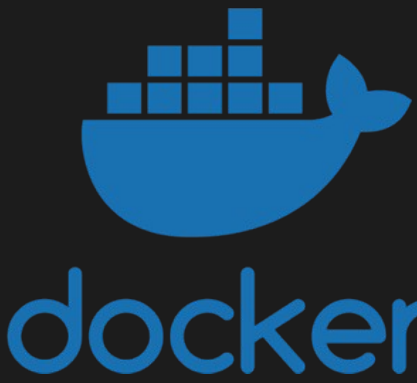
Dynamic/Server websites

- Can use any technology on the backend.
- Usually connects to other services, a database etc.

Hosting for Dynamic websites (Case study: Node JS)

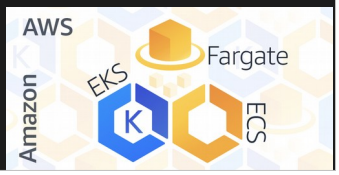
- Virtual servers on the cloud such as Amazon EC2.
- Heroku virtual Node JS server.
- You own computer / Raspberry Pi clone (I use this).
- Rent someone else's server for cheap such as from https://lowendbox.com/

Server/service hosting got easier



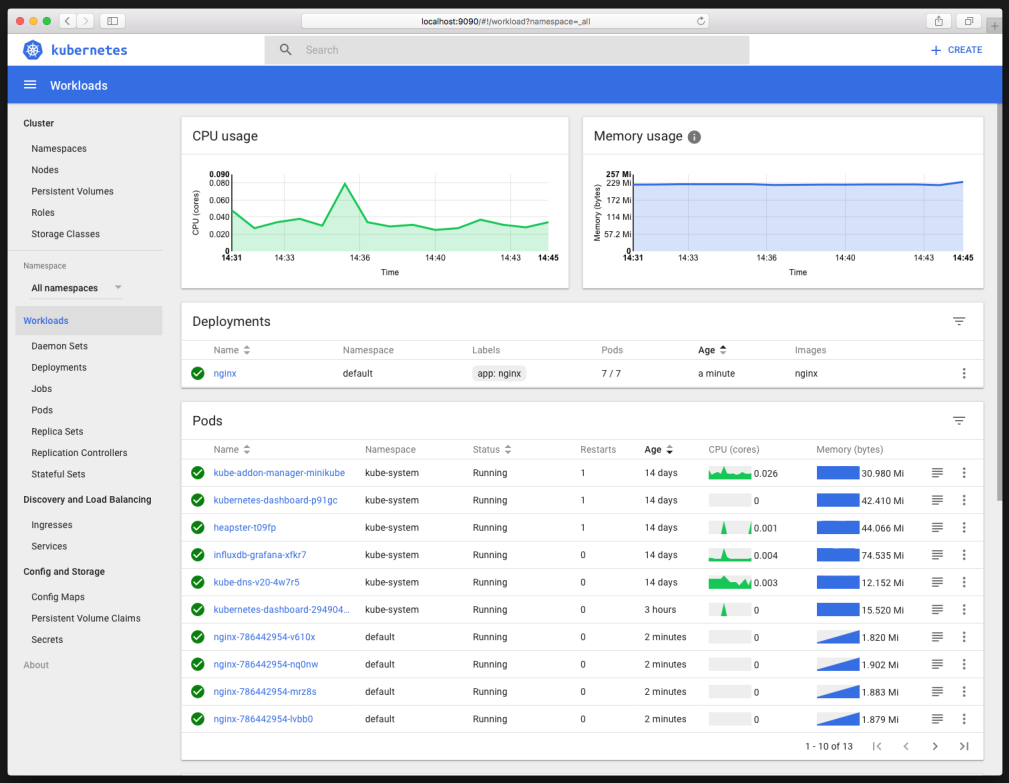
multiplied

kubernetes

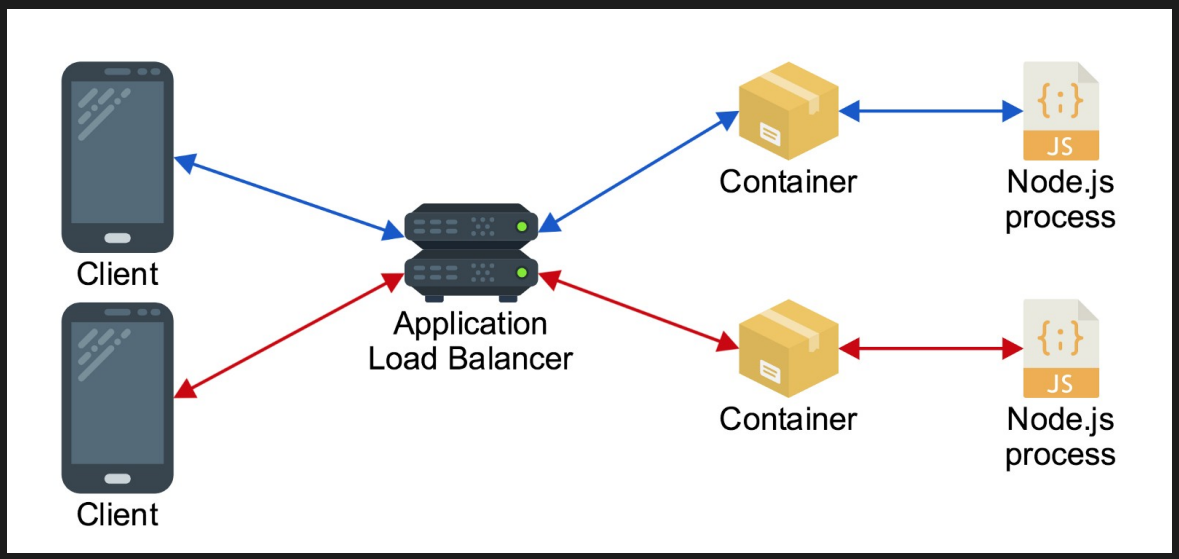


When your server components, are stateless, you can easily share the load between them, do a failover, move them around etc, with no need to do any replication.

Generally application servers are stateless while database/store services are stateful.


localhost:9090/#!/workload?namespace=_all
kubernetes
Search
CREATE
Workloads
Cluster
Namespaces
Nodes
Persistent Volumes
Roles
Storage Classes
Namespace
All namespaces
Workloads
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Discovery and Load Balancing
Ingresses
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
About
CPU usage
CPU (cores)
0.090
0.080
0.060
0.040
0.020
0
14:31
14:33
14:36
14:40
14:43
14:45
Time
Memory usage
Memory (bytes)
257 Mi
229 Mi
172 Mi
114 Mi
57.2 Mi
0
Deployments
Name
Namespace
Labels
Pods
Age
Images
nginx
default
app: nginx
7 / 7
a minute
nginx
Pods
Name
Namespace
Status
Restarts
Age
CPU (cores)
Memory (bytes)
kube-addon-manager-minikube
kube-system
Running
1
14 days
0.026
30.980 Mi
kubernetes-dashboard-p91gc
kube-system
Running
1
14 days
0
42.410 Mi
heapster-t09fp
kube-system
Running
1
14 days
0.001
44.066 Mi
influxdb-grafana-xfkr7
kube-system
Running
0
14 days
0.004
74.535 Mi
kube-dns-v20-4w7r5
kube-system
Running
0
14 days
0.003
12.152 Mi
kubernetes-dashboard-294904...
kube-system
Running
0
3 hours
0
15.520 Mi
nginx-786442954-v610x
default
Running
0
2 minutes
0
1.820 Mi
nginx-786442954-nq0nw
default
Running
0
2 minutes
0
1.902 Mi
nginx-786442954-mrz8s
default
Running
0
2 minutes
0
1.883 Mi
nginx-786442954-lvbb0
default
Running
0
2 minutes
0
1.879 Mi
1 - 10 of 13

Load balancing



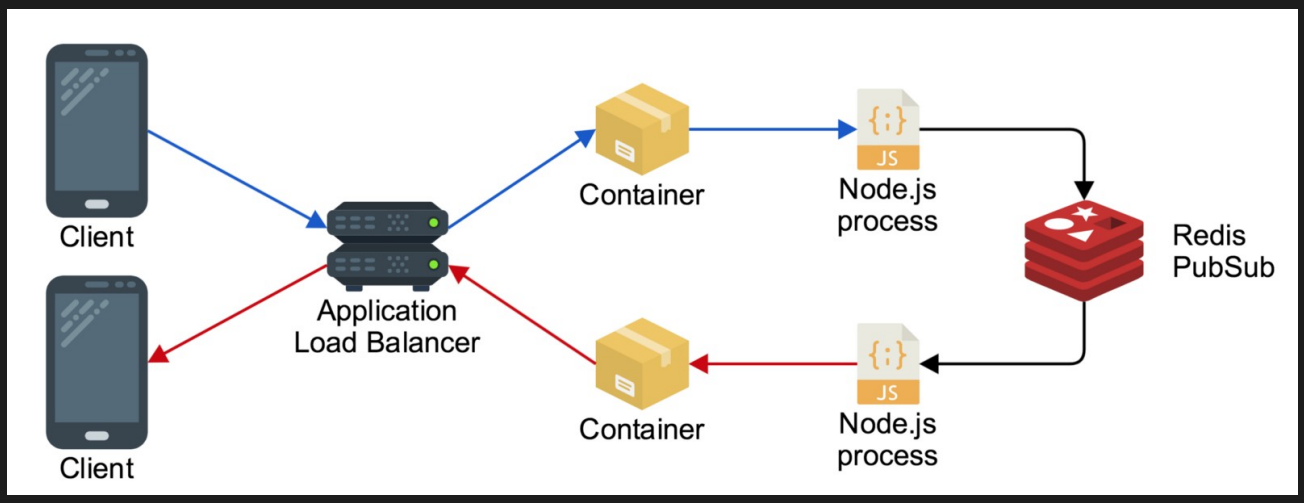
REST vs Websockets

Websockets are statefull because they maintain connected clients.

The session itself also occurs in one server until disconnected.

But what happens when a server (let’s say a chat server) publishes a message to all of the clients?


redis


Client
Client
Application
Load Balancer
Container
Container
Node.js
process
Node.js
process
Redis
PubSub

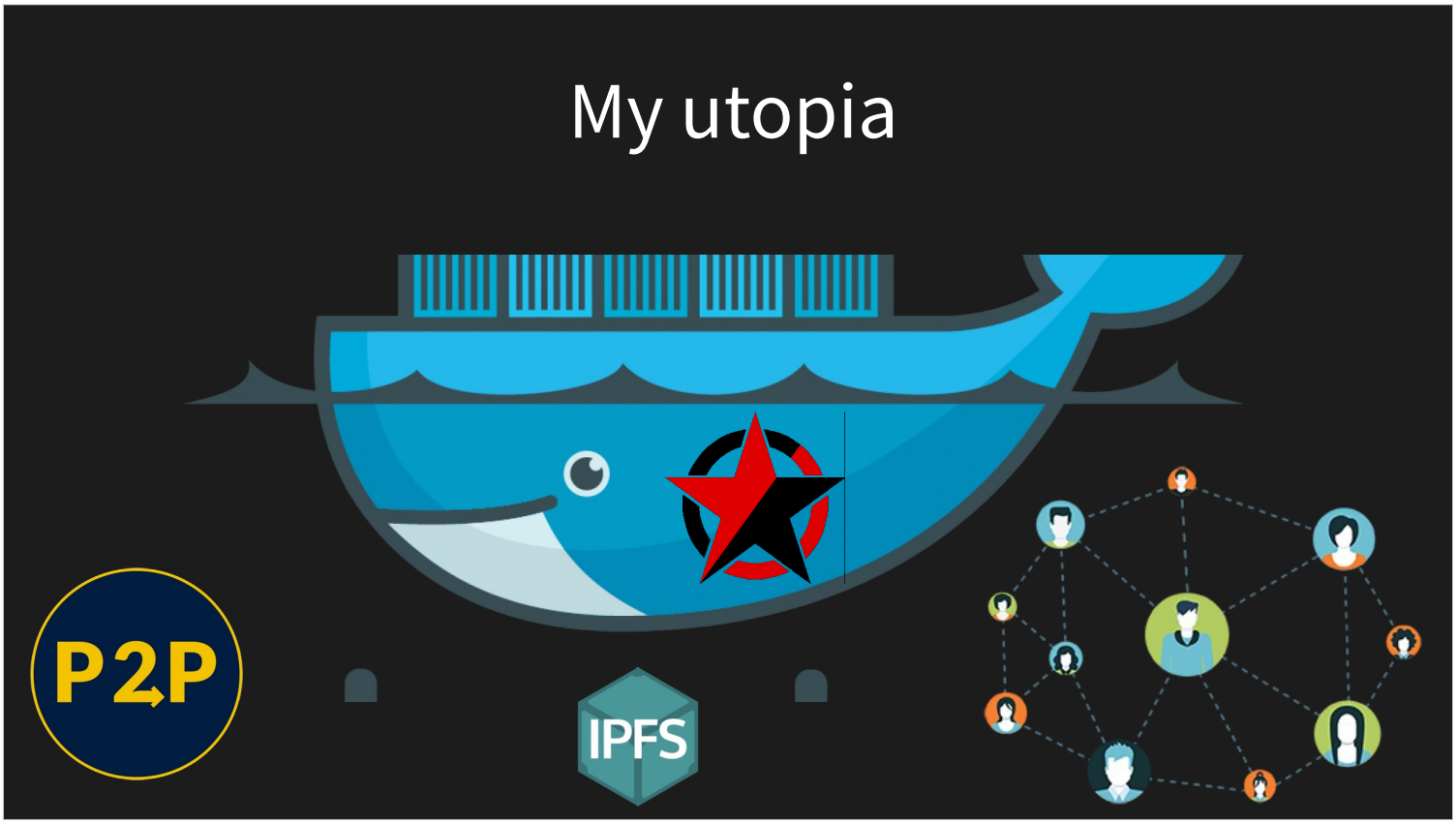
More here

Serverless websites

- All the power of the server, but no need to worry about infrastructure, load balancing, scaling, etc.
- Can pay per usage and not per up-time

Serverless websites providers

- Firebase.
- Amazon's AWS ecosystem.



How about security?
Blockchain?

Continuing YOLO

Organizing the State

state.js

```
export const Render = {
  backgroundColor: "rgba(255,0,0,0)",
  updateBackground: function(color) {
  }
};
observable(Render);
```

Index.js

```
import * as State from './state'

 State.Render.on(
      'background-updated', (color) => {
      }
 )

State.Render.updateBackground("rgba(1,1,1,1)");
```

More tips

- Think about how we can trigger a background change when a bottle appears , disappears and then appears again.
- Let's have another state variable – the state of the bottle. Let's trigger a 'bottle-present-updated' event **only when the state changes –** Only when the bottle switches from present to non-present or vise-versa.
- Also it might be more readable if we send the actual changed value along with the event trigger.