Running Node.js on customer machines
API design for building a cloud-native application

Atishay Jain
Senior Computer Scientist, Adobe
@atishay811
Atishay Jain
Senior Computer Scientist, Adobe

Part of many iconic software products from Photoshop and Illustrator to Adobe Capture. Creator of award winning augmented reality mobile apps from the IxDA Interaction Design award to the FWA App of the day. 10 years of industry experience working on all platforms from desktop, frontend, backend and mobile.

https://atishay.me       https://linkedin.com/in/atishay
Agenda

• Two definitions of Cloud Native
• Making the “Cloud” Native
• History of computing
• Where does JavaScript fit in
• Running Node.js as a background process
• It's all about use cases

Image from Littlefun.org
Two definitions of Cloud Native
Two definitions of Cloud Native

Cloud Native for developers

- Containers
- DevOps
- Continuous Delivery
- Microservices
Two definitions of Cloud Native

Cloud Native for developers
- Microservices
- Continuous Delivery
- Containers
- DevOps

Cloud Native for users

Image from gyfcat
Two definitions of Cloud Native

Cloud Native for developers

- Continuous Delivery
- Microservices
- Containers
- DevOps

Cloud Native for users

Image from giphy
Two definitions of Cloud Native

Cloud Native for developers

- Continuous Delivery
- Microservices
- Containers
- DevOps

Cloud Native for users
What is cloud & native for the users?

User’s Cloud
- Collaboration
- Synchronization
- Automatic updates

User’s Native
- Realtime performance.
- Handle Flaky network.
- Offline support.
Making the “Cloud” Native
Making the “Cloud” Native

Cloud Native for developers

- Microservices
- Continuous Delivery
- Containers
- DevOps

User’s Cloud
- Collaboration
- Synchronization
- Automatic updates

User’s Native
- Realtime performance.
- Handle Flaky network.
- Offline support.
Making the “Cloud” Native

Fighting with the laws of Physics.
Making the “Cloud” Native

- Multi-regions
- CDN
- Edge computing
- ??
Making the “Cloud” Native

Does it feel native?
Making the “Cloud” Native

Cloud Native for developers

- Microservices
- Continuous Delivery
- Containers
- DevOps

User’s Cloud
- Collaboration
- Synchronization
- Automatic updates

User’s Native
- Realtime performance.
- Handle Flaky network.
- Offline support.
Making the “Cloud” Native

Cloud Native

Continuous Delivery

Containers

DevOps

Microservices

Developer’s Cloud Native

User

CDN/Edge

Replication Regions

Core Region
Making the “Cloud” Native

Cloud Native

Continuous Delivery

Containers

DevOps

Microservices

Microservice as Daemon

User

CDN/Edge

Replication Regions

Core Region
Making the “Cloud” Native

Microservice as Daemon

Bad Idea
• Security?
• How about updates?
• Offline support?
• Where is the truth?
Making the “Cloud” Native

Cloud Native for developers

- Continuous Delivery
- Microservices
- Containers
- DevOps

User’s Cloud
- Collaboration
- Synchronization
- Automatic updates

User’s Native
- Realtime performance.
- Handle Flaky network.
- Offline support.
History of Computing
History of Computing

Mainframes
When there was a single machine.

Personal Computer
When everyone had a machine

Cloud
When one machine was not enough.

Future
????
History of Computing

Power

Flexibility
Where does JavaScript fit in?
Where does JavaScript fit in?

- Built for network based async communication.
- Heavily invested language.
- The language designers have been challenged with making the language look native.

Image from Pixabay, Wikipedia
Where does JavaScript fit in?

Service Workers

Image from Pixabay, Wikipedia
Where does JavaScript fit in?

Cloud Native for developers

Service Workers

Image from Pixabay, Wikipedia
Where does JavaScript fit in?
Where does JavaScript fit in?

- Built to be run 24x7.
- Asynchronous.
- I/O Optimized.
- Mostly self-contained.
Running Node.js as a background process
Running Node.js as a background process

Cloud Native for developers

- Continuous Delivery
- Microservices
- Containers
- DevOps

Logo from Wikipedia
Running Node.js as a background process

Caveats with Node.js

• Built for the server.
• No proxy support built in.
• Packaging is messy.
Its all about use cases
Conclusion

- Cloud & native for user
- Service as a daemon
- Node.js on desktop