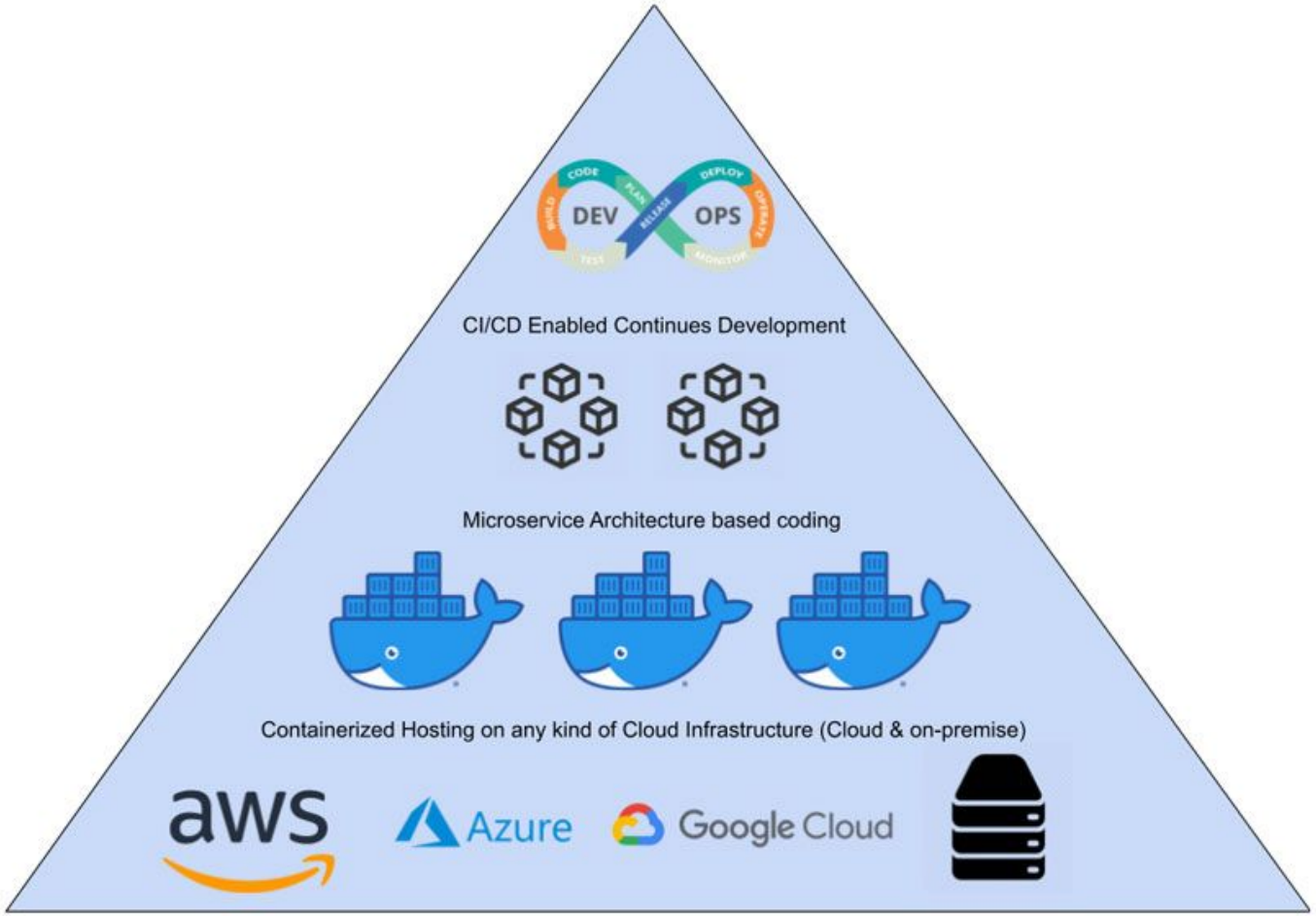
Liferay DXP 7.4 in a cloud-native, headless and serverless world

George Karouzos
CEO, Technopolis S.A.

Cloud Native


DEV
OPS
CI/CD Enabled Continues Development
Microservice Architecture based coding
Containerized Hosting on any kind of Cloud Infrastructure (Cloud & on-premise)
aws
Azure
Google Cloud

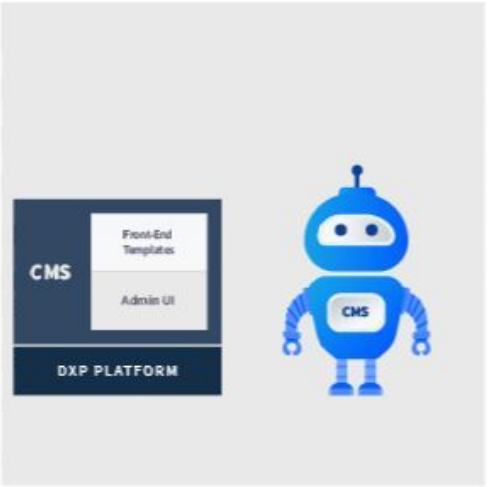
Headless

"headless" (that is, configured without a graphics card and monitor)

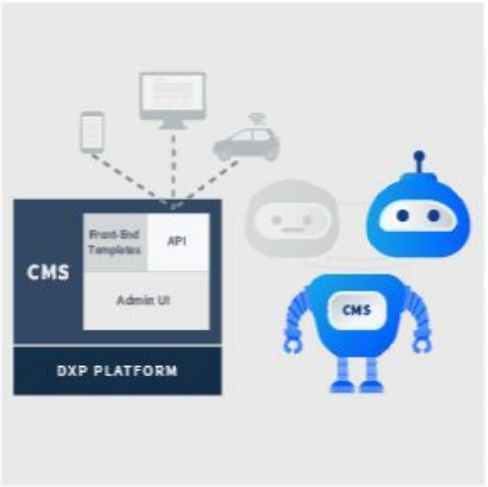
SPARCstation (90ies)

"headless" (that is, configured without front-end rendering)

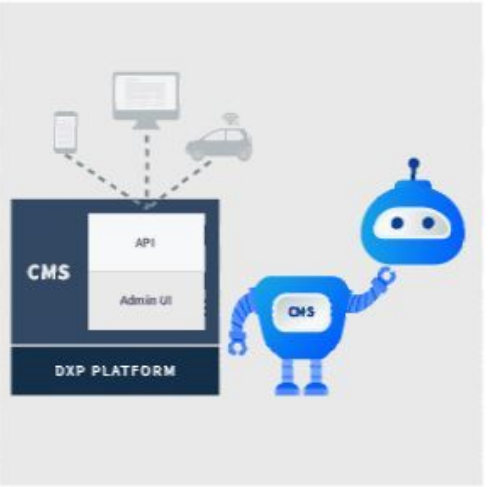
Contemporary use… for an App, an API or a Service



TRADITIONAL

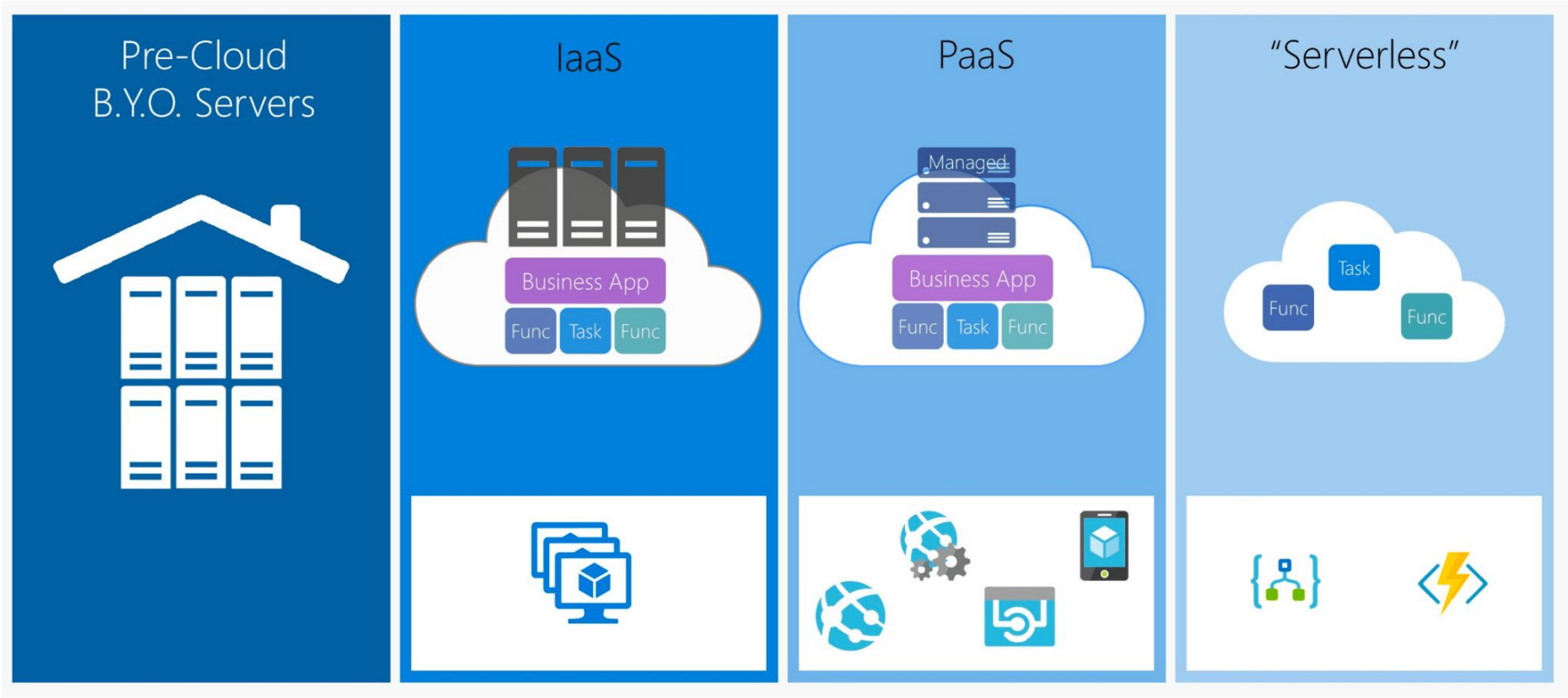


HYBRID



HEADLESS

Serverless


Pre-Cloud
B.Y.O. Servers
IaaS
Business App
Func
Task
Func
PaaS
Managed
Business App
Func
Task
Func
"Serverless"
Task
Func
Func

Serverless is a deployment paradigm

- Deploy and get an endpoint
- Everything else is a black-box (servers, ingress, containers, network,...)
- FaaS is the common case (AWS lambda, GCP/Azure Functions,... Cloudflare Workers, Vercel, Fly,…)
- SaaS is Serverless with no deployment

And where does Liferay fit in?

Liferay Deployment Options


Liferay Digital Experience Platform Self-Hosted
Liferay Experience Cloud Self-Managed
Liferay Experience Cloud
On Premise
PaaS
SaaS

Which of them are Cloud Native?

ALL!

(Potentially)

#1 Self-Hosted

- Not necessary On-Premises
- Can be deployed on any IaaS Cloud
- Can be deployed on any K8S Cloud offering (+ On-Premise e.g. Tanzu?)
- But you have to manage resources, network, ingress, ci/cd, monitoring, upgrades of the full stack
- DB, Elasticsearch and Block storage can be XaaS
- Priced per server instance (or k8s pod) and per environment

#2 Self-Managed (LXC SM)

- Cloud native (K8S, docker, ci/cd, devops console)
- All inclusive deployment blueprint (ingress, DB, Elasticsearch and Block storage)
- But you have to manage the Liferay instance (e.g. upgrades)
- Priced per instance and per environment
- Autoscaling ready (PAYG, /hour/instance)

#3 LXC

- Cloud native - but you shouldn’t care…
- All inclusive deployment blueprint - but you shouldn’t care…
- You only have to manage configuration and data
- Upgrades by Liferay!
- Priced per usage (users and views)
- 2 Environments, PROD and UAT
- Autoscaling included

But what about microservices?

Is Liferay a Monolith?

Yes and No…

- ❏ Internally is modularized using OSGi microservices
- ❏ But ok, this is not “pure” microservice architecture
- ❏ Persistent data .vs. Stateless microservices…
- ❏ Headless does not presuppose microservices…
- ❏ CMSs are inherently monolithic (shared data)
- ❏ Could, at least, customizations follow microservice architecture?

Enter “Client Extensions”

- Introduced along with LXC
- The new way to extend and customize Liferay
- Works nicely with Headless
- The new “deployment” artefacts which live outside Liferay
  - “Custom Element” - a js widget in any framework
  - Spring-boot microservice or app
  - Node.js microservice or app
  - Other customizations (styling, config, assets, cron)

Client Extensions benefits

- ❏ Liferay stays “clean” → effortless upgrades!
- ❏ Extensions still have access to Liferay (assets, session, etc)
- ❏ Developer’s choice of language / framework / tooling
- ❏ Broader pool of experienced developers
- ❏ Easier debugging
- ❏ Extensions can be scaled independently

Headless is the new black

Everyone wants it and looks nice on it!

Headless and Liferay

- It’s here since 7.1 (5 years)
- Evolved to support everything, not just get content
- Supports both REST and GraphQL
- Supports authentication, permissioning, RBAC
- Supports also low-code/no-code features (e.g. Objects, Forms)
- Compared to pure headless solutions, allows hybrid option

Liferay Architecture Guidelines

Do’s and Don'ts

- ❏ Use docker and k8s even for self-hosted deployments
- ❏ Model your content and data within Liferay (WC, D&M, Objects, etc)
- ❏ Use Headless delivery either in hybrid or decoupled, choice is yours
- ❏ Use Client Extensions as much as possible (even if not on LXC) for customizations
- ❏ Use a powerful frontend framework as Custom Elements
- ❏ Content admin workflow is still the way to go. Deploying for content changes is going backwards
- ❏ Open source is still relevant. Closed sourced solutions and SaaS-only solutions lock you in

Thank you!

and Q&A…