ABOUT THIS TALK

This is our **technical adventure**

- multi-tenancy with advanced per-tenant customization
- build a Software factory using VMs, Docker and now CaaS

This is **our story** but, hopefully, this should also make sense for

- applications that are **modular and configurable**
- applications **moving to the cloud / SaaS model**
A LONG ROAD

2012/2013  initial idea and first tentative

2014     first viable version

now       current development
          and future architecture
2012/13 - IDEA & FIRST TRY

Multi-Tenants, Cloud & Application Factory
we provide a **Platform** that **developers** can use to build **highly customized** Content Applications

we provide **components**, and the **tools** to assemble everything we do is **open source**

our customers are people building software in-house - **software vendors** - **SaaS**
Repository

Services

*Workflows, Conversions, Diff, Notifications, Activity* ...
BUILDING ON TOP OF A PLATFORM
BUILDING ON TOP OF A PLATFORM

- Addon components
- Platform Infrastructure
- elastic
- MongoDB
- PostgreSQL
BUILDING ON TOP OF A PLATFORM
BUILDING ON TOP OF A PLATFORM

- Application Configuration
- Custom components
- Addon components
- Platform Infrastructure

- elastic
- MongoDB
- PostgreSQL
BUILDING ON TOP OF A PLATFORM

Project Management

Application Configuration
Custom components
Addon components
Platform Infrastructure
BUILDING ON TOP OF A PLATFORM

Digital Assets Management
BUILDING ON TOP OF A PLATFORM

Case Management
BUILDING ON TOP OF A PLATFORM

Collaboration
BUILDING ON TOP OF A PLATFORM

Application Configuration

Custom components

Addon components

Platform Infrastructure

elastic
mongoDB
PostgreSQL
1 CUSTOMER

Custom deployment
MORE CUSTOMERS

Industrialization
PLM SOLUTION

Customer/Tenant 1

Customer/Tenant 2

Styles
Screen Layouts
Meta-Data
Workflows
Business Logics
Connectors
SOLVING THE PROBLEM
APPLICATION LEVEL MULTI-TENANTS

all clients share the same application

application manages data and configuration partitionning

Document Store
Security
Life Cycle
Indexing
Versioning

Shared Repository (security based isolation)
APPLICATION LEVEL MULTI-TENANTS

Shallow isolation

- quota management is not efficient
- customization options are limited

Monolithic

- same version, same component set
- same upgrade and maintenance policy

Not even simple

- scale out is not that easy (i.e. move a tenant)
- per-tenant Backup/Restore is not easy
- Heterogeneous deployment units
  VM level / JVM level / App level
CONTAINER LEVEL MULTI-TENANTS

rely on infrastructure to provide tenants isolation
application does not need to be impacted

Flexible
Unlimited Customization
Full Isolation & Quotas
Create "on demand" application for each customer

- use Container level isolation
- provision infrastructure from the Cloud
- custom assembly for each customer
Create "on demand" application for each customer

- use Container level isolation
- provision infrastructure from the Cloud
- custom assembly for each customer
APPLICATION FACTORY

Create "on demand" application for each customer

- use Container level isolation
- provision infrastructure from the Cloud
- custom assembly for each customer
APPLICATION FACTORY

Create "on demand" application for each customer

- use Container level isolation
- provision infrastructure from the Cloud
- custom assembly for each customer
Create "on demand" application for each customer

- use Container level isolation
- provision infrastructure from the Cloud
- custom assembly for each customer
Let's do this ...

... for our own use cases!
NUXEO TRIALS USE CASE

We build a platform and customization tools

We want people to be able to taste the full experience

- choose their components
- configure business rules
- run the app they build

Online demo site with full per-tenant customization
PAAS VS AUTOMATION

Early testing with **CloudFoundry / OpenShift**
- Fast moving ground
- Very opinionated
- Our app does not fit in Java PaaS

Go with **Deployment Automation on IaaS**
- Seems easier
- Better match for sys-admin / devops
SOME CHALLENGES

free customers with high expectations!

Prospects do not pay
Prospects want to access all features
We want them to have a great experience

Cost is a major concern

Optimize hosting infrastructure
CUTTING COSTS

EC2 Spot instances

Start/Stop instances when needed

Leverage AWS Services
EARLY RESULTS

works but ...

A lot of moving parts

- no clear deployment unit
- lot of scripting

Slow

- provisioning is slow
- start/stop too slow to be usable
DEAD END?

We need lighter footprint
We need faster startup
Some developers have started playing with lightweight containers because VMs are too fat & slow!

[dev] Cool new stuff on cloud related techs 03/2013

http://www.docker.io/: a command line tool for launching and managing arbitrary processes using LXC. Open Source project contributed by the dotcloud guys.

and Docker!
2014 - NUXEO.IO V1

Build the Application Factory on Docker
First target is to power Nuxeo Trials
   need a smooth UX

but also build a **generic infrastructure**
   publish work as opensource

Assign a team of java **developers**
   **not** system-administrators

Expect results in **3 months**!
   **rebuild / refactor every 3 months**
Build Your Own Application
Build Your Own Application

Select target Platform
Build Your Own Application

Pick additional components
Build Your Own Application
Build Your Own Application

Build new components
Build Your Own Application

Select custom Components
Build Your Own Application

Define Application
Model
Build Your Own Application
Build Your Own Application

Choose deployment environment
Build Your Own Application

**Deploy & Run!**
Build Your Own Application

Make your own container recipe

- choose your ingredients
- assemble
- run

Available to the web user
NUXEO.IO FACTORY

nuXeo.io offers services for managing and running NuXeo Platform clusters on the Cloud. It provides a quick, easy way to access lightweight NuXeo Platform instances on demand.

This management console enables you to create, start, stop and wake up environments. An environment gives you access to a NuXeo Platform instance. You can create a new environment just by entering a name in the input and clicking on the Create button. You will be given the URL to access your new NuXeo Platform instance.

If you registered for a NuXeo Connect trial, we have already created an environment for you. It is bound to your trial NuXeo Studio project, making it very easy to configure your NuXeo Platform Instance. Click on the Environment name on the right, and then click the Start button. Your trial NuXeo Platform Instance will be up in about a minute. Enjoy!
PRINCIPLES

Docker containers!

Leverage AWS infrastructure
PRINCIPLES

Passivation

- Less than 5% of trials are active at a given time
- Fast start/stop (no data)
- High density hosting on AWS
PRINCIPLES

Dual state orchestration

• Expected vs Actual
• Decoupling & Monitoring

Use a distributed registry
CASTING

Containers infrastructure:
Docker + CoreOS

Scheduler:
Fleet

Distributed registry:
etcd

Monitoring:
DataDog
CASTING
Casting
CASTING

The Containers:

Nuxeo Servers  (1 per tenant)

Manager.io (Nuxeo +AngularJS)

Passivator (Go Service)

ArkenCtl (Go Cmd)

Dynamic reverse proxy Gogeta
Deploy new tenant
BIG PICTURE

Register new tenant
BIG PICTURE

Deploy more tenants
BIG PICTURE

Route request to Customer X
BIG PICTURE

Route request to Customer X
PASSIVATE

/elb

Gogeta

Gogeta

Gogeta

Docker

Passivator

Docker

NXIO-001

Docker

NXIO-002

Docker

NXIO-003

etcd

/services/NXIO-001/lastAccessTime: 1 hour and 1 minute ago
/services/NXIO-002/lastAccessTime: 1 minute ago
/services/NXIO-003/lastAccessTime: 3 minutes ago
/services/NXIO-001/status/current: passivated
/services/NXIO-002/status/current: started
/services/NXIO-003/status/current: started
/services/NXIO-001/status/expected: passivated
/services/NXIO-002/status/expected: started
/services/NXIO-003/status/expected: started

/services/NXIO-001/status/current: passivated
/services/NXIO-002/status/current: started
/services/NXIO-003/status/current: started
ACTIVATE

-services/NXIO-01/status/expected: started
-services/NXIO-02/status/expected: started
-services/NXIO-03/status/expected: started

-services/NXIO-01/status/current: started
-services/NXIO-02/status/current: started
-services/NXIO-03/status/current: started
MONITORING
SOME GOOD RESULTS

1000+ instances/month managed on 4 EC2 VMs (m3.2xlarge)
Production hosting for some Nuxeo based applications
Eventually stable

Docker and Go are really great
The vision is good!
All Good?
Almost!

- Still lots of shell scripts
- Shared storage doesn't scale well
- Setup complexity
- Only one deployment model
MISERIES

Lots of **moving parts** + Lots of **young solutions**

*Experience a new type of failure every day!*

- CoreOS updates
- btrfs fragmentation and IO issues
- etcd stability
- fleet "shortest path" scheduling
- ...

[Caution Wet paint]
Lot of boring **glue code**

- Networking, Port Mapping, provisionning ...
- developers are sick of shell scripts

**Storage** management is an issue

- asymmetrical model & scalability
- **Security** concerns on shared infrastructure
Troubleshooting is tricky
Command line tools
Still too scary for a customer
the system is complex to setup
NOW - NUXEO.IO V2

Leverage our experience and Docker evolutions
DRIVERS

Docker ecosystem evolved
Kubernetes, Swarm, Compose, Rancher ...

We believe in the initial vision is good
Customers starts to like the idea of Containers

We have learned a lot from Nuxeo.io v1
Time for a reboot
ADDITIONAL CONSTRAINTS

Cluster configurations
1 node, 2 nodes, 7 nodes ...

Production hosting
take over Nuxeo-Cloud

Customer compliant
avoid or hide and package the glue code
Remove as much code as we can
LEVERAGE DOCKER EVOLUTION

Networking in Docker / Rancher / Kubernetes
- No more "manual" port & DNS mapping
- Use **Software Defined Network**

Clustering in Docker-Swarm / Rancher / Kubernetes
- No more scripting automation

Stack templating in Docker-Compose / Rancher / Kubernetes
- Manage an application as a set of containers

Free from the Shell!
less glue code to write / debug
focus on application level
Avoid building a CaaS infrastructure. Focus on Application Customization & Templating. Keep some flexibility on Cluster Orchestration.
CAAS CHOICE

Nice high level REST API abstraction on CaaS provider Swarm/Kubernetes/Cattle

Administration UI
- Infrastructure vs Application view
- Monitoring and SSH access

Close to Docker
- Built on top of Docker and Compose
- Contribute to Docker
CAAS CHOICE

Application Management
• Stack definition & templating

Provide additional features
• Health Monitoring
• Load Balancing and DNS
• Rolling upgrades

Volume plugins:
convoy: NFS / GlusterFS
NUXEO.IO WITH RANCHER

Arken

Gogeta
Passivator
ArkenD

Rest API

Nuxeo.IO Manager
ArkenCtl

goarken

CaaS Adapter

etcd

nuxeo-io V1 CoreOS / Fleet

Swarm

Rancher

Swarm

Cattle
Kubernetes
VM1: 192.168.99.100

Rancher Server


rancher-agent
NUXE0.IO WITH RANCHER - DEMO
DIRECT GAINS

Rancher & Docker do the heavy lifting

Deploying Arken is easy

One unique API to deploy new tenants
ARKEN CONTENTS

Application Templating
Package Selection: Wizard + Config + Docker File
Deployment template: Compose + Rancher

Passivation Management
Passivation aware Routing
State management
API & Adapter

Independent
Go Based
OpenSource
MULTI-TENANTS?

Application Level Multi-Tenancy

- *data only isolation*

Container Level Multi-Tenancy - Nuxeo.io v1

- *processing isolation - shared data sources*

Container Level Multi-Tenancy - Nuxeo.io v2

- *complete data and process isolation*
MULTI-TENANTS?

Tenants sharing an Application ➞ Tenants sharing Infrastructure + dedicated applications

**processing** isolation: *docker containers*

**reusable execution units**

**data** isolation: *docker and volumes*

**stateful containers**

application **configuration**: Rancher + Arken + Nuxeo

**software defined tenant**

Application Factory: Rancher + Arken
What's Next?

- External resource provisioning (S3, MongoDB Atlas)
- End to end security
- Shared services stack (converters)
Jenkins: Can push images, therefore can replace keyset but not see them
Rancher: Can pull images so have indirect access to private key
Arken: Have only view on the public key stored in etcd
Pulling Customer: Can only have access to the Nuves, but not to the cloud credentials and key
ANY QUESTIONS ?

Thank You !

We are hiring !
New York, Paris, Lisboa

http://www.nuxeo.com/careers/