Beyond cf push

Christof Marti (@christofmarti)
Senior Lecturer / Researcher @ ZHAW ICCLab
A journey to manage Complex Microservice Applications like a BOSH
The fellowship

- **ZHAW InIT Cloud Computing Lab (ICCLab)**
  Research Lab at School of Engineering at Zurich University of Applied Sciences (ZHAW)
  Working on the forefront of cloud technologies

- **dorma+kaba*)**
  One of the top three companies in the global market for physical security and access solutions with pro forma sales of more than CHF 2 billion (USD 2.1 billion) and around 16'000 employees in more than 50 countries.

- **Swisscom**
  Switzerland's leading telecom provider and one of its leading IT companies. Cloud Foundry certified provider.

*) merger of Kaba Group from Rümlang near Zurich (Switzerland) and Dorma Group based in Ennepetal near Düsseldorf (Germany) in September 2015
The dorma+kaba exivo story
exivo – Access Control as a Service

- dorma+kaba: products, solutions and services in the area of security and access to buildings and rooms.
- Kaba exivo: new easy, convenient and secure solution to plan, monitor and control access to your premises.
exivo – Access Control as a Service

www.kaba.com/exivo/en
exivo – modern system design

- Microservices based cloud native design
- CQRS/ES based architecture
- Distributed Domain Driven Design approach
- 12 factor app compliant
- Very high requirements on security & reliability

Typical structure of an exivo app
exivo – system overview

**App**
- Frontend Applications
  - Customer
  - Partner
  - Management
- Business Domain
- 50 Applications
- 48 Services
- 150 Instances

**IoT**
- Backend connection to periphery
  - Devices
  - Locks
  - Access Manager
- 30 Applications
- 18 Services
- 90 Instances
Main Challenges

Overview of apps and services

Increased effort for maintenance

Slow and unreliable deployments
ICCLab joining the journey

Conception and Engineering of a continuous deployment toolkit for exivo.
Learning from BOSH
Learn from BOSH and adapt

- Reuse successful BOSH concepts
- Adapt for the Application-Layer
- Add flexible and reusable workflows
Introducing
www.push2.cloud
@push2_cloud
push2cloud

Application management and deployment toolkit

- Sophisticated application configuration
- Release- & Deployment management
- Target platform agnostic
- Flexible, customizable workflow framework
- Extensible
- Open Source
Configuration Overview

Binding to a specific target environment

- Deployment Manifest (DM)
- Deployment

Independent of target environment

- Release Manifest (RM)
- Release
- Composition of applications
- Service binding
- Service
- App dependency

- Application Manifest (AM)
- Application

Normalized configuration (state) of a deployment

- Deployment Configuration (DC)
Application Manifest example

```json
{
    "name": "push2cloud-example-api",
    "version": "1.0.0",
    "deployment": {
        "memory": 512,
        "disk": 512,
        "instances": 1,
        "startTimeout": 100,
        "stageTimeout": 200,
        "gracePeriod": 20,
        "buildpack": "https://github.com/cloudfoundry/nodejs-buildpack.git",
        "env": {
            "FOO": "BAR"
        },
        "appConnections": {
            "push2cloud-example-host": {
                "injectedCredentials": false,
                "urls": {
                    "START_URL": "/start"
                }
            }
        },
        "serviceBinding": [
            "todo-db"
        ]
    }
}
```
Release Manifest example

```
{
  "name": "push2cloud-example",
  "description": "release manifest for push2cloud example",
  "version": "1.0.0",
  "source": {
    "type": "git",
    "url": "https://github.com/push2cloud/push2cloud.git",
    "referenceValue": "master"
  },

  "apps": {
    "push2cloud-example-host": {
      "path": "example/apps/host"
    },
    "push2cloud-example-api": {
      "path": "example/apps/api"
    }
  }
}
```
Deployment Manifest example

```json
{
    "name": "push2cloud-example",
    "version": "1.0.0",
    "description": "deployment manifest for push2cloud example",
    "target": {
        "type": "cloudfoundry",
        "api": "https://api.lyra-836.appcloud.swisscom.com",
        "space": "${SPACE}"
    },
    "release": {
        "name": "push2cloud-example",
        "source": {
            "type": "git",
            "url": "https://github.com/push2cloud/push2cloud.git",
            "referenceValue": "master"
        },
        "manifest": "example/manifests/release.json"
    },
    "applicationDefaults": {
        "memory": "512M",
        "env": {
            "SYSTEM_VERSION": "${SYSTEM_VERSION}"
        }
    },
    ...
}
```
Deployment Manifest example

```
"serviceMapping": {
  "*-db": {
    "type": "redis",
    "plan": "small"
  }
},
"domains": {
  "public": "scapp.io",
  "internal": "scapp.io"
},
"apps": {
  "push2cloud-example-host": {
    "routes": {
      "public": [
        "push2cloud-example-host-$\{space}\"
      ]
    }
  },
  "push2cloud-example-api": {
    "routes": {
      "internal": [
        "${appname}-$\{space}\"
      ]
    }
  }
}
```
Deployment Configuration

- Data structure (JSON) containing the fully normalized configuration (state) of a deployment.
- Two ways to build the DC:

  - **AM** → **RM** → **DM** → **DC**
    - **compile**
    - **desired state**

  - **retrieve**
    - **DC**
    - **actual state**

Target environment (CF space)
Get into the flow
Workflows

- Collection of actions to migrate from actual to desired state
- Using the **Desired Deployment Configuration** as input.
- Imperative by design
  - Workflows are code (JavaScript)
  - Easy to adapt and build custom workflows
- Asynchronous
  - Functional design
  - Controlled parallel operations
- Reliable
  - Timeouts and retries
  - Grace periods
Workflow internals

Desired Deployment Configuration → getActual

Actual Deployment Configuration

push app → switch routes

new apps activates

old apps deletes

desired state

retreives

actual state
Workflow example

```javascript
const blueGreen = (deploymentConfig, api, log) =>
    waterfall(
        [ init(deploymentConfig, api, log)
            , map(packageApp, missing.apps)
            , mapSeries(api.createServiceInstance, missing.services)
            , map(api.createRoute, missing.routes)
            , map(api.createApp, missing.apps)
            , mapLimit(api.uploadApp, missing.apps)
            , map(api.createRoute, missing.routes)
            , map(api.createApp, missing.apps)
            , map(api.stageApp, missing.apps)
            , map(api.bindService, missing.serviceBindings)
            , map(api.startAppAndWaitForInstances, missing.apps)
            , map(switchRoutes(api), combine('desired.routes', old.associatedRoutes,
                (r) => (r.unversionedName + r.hostname + r.domain)))
            , map(api.stopApp, old.apps)
            , map(api.unbindService, old.serviceBindings)
            , map(api.deleteApp, old.apps)
        ]
    );
```
Show me how to push 2 the cloud
3 steps to deploy

1. **Configure** the *Deployment Manifest*
   
   vi `../manifests/deployment.json`

2. **Compile** the *Desired Deployment Configuration*
   
   `p2c compile -e ../manifests/deployment.json`

3. **Execute** a *workflow* to migrate the system from actual to desired state
   
   `p2c exec ./push2cloud-cf-workflows/blue-green.js`
Demo

```bash
push2cloud-cf-adapter called getAppInstances... push2cloud-example-host-2.0.0 +4s
push2cloud-cf-adapter called graceRequest... +0ms
push2cloud-cf-adapter:verifyInstancesNotCrashing 5. retry will try 5 for push2cloud-example-host-2.0.0 +262ms
push2cloud-cf-adapter called getAppInstances... push2cloud-example-host-2.0.0 +4s
push2cloud-cf-adapter called graceRequest... +1ms
push2cloud-cf-adapter called associateRoute... push2cloud-example-host-2.0.0 +284ms
push2cloud-cf-adapter called graceRequest... +2ms
push2cloud-cf-adapter called associateRoute... push2cloud-example-api-1.0.0 +1ms
push2cloud-cf-adapter called graceRequest... +1ms
push2cloud-cf-adapter called startApp... push2cloud-example-host-1.0.0 +315ms
push2cloud-cf-adapter called updateApp... push2cloud-example-host-1.0.0 +0ms
push2cloud-cf-adapter called graceRequest... +3ms
push2cloud-cf-adapter called deleteApp... push2cloud-example-host-1.0.0 +1s
push2cloud-cf-adapter called graceRequest... +0ms
Workflow: /node_modules/push2cloud-cf-workflows/blueGreen.js done ✔ 3.25 minutes
</d/p/e/deployer >>> cf a
Getting apps in org dormakaba / space iot-cf-test as michael.erne@gmail.com...
OK

<table>
<thead>
<tr>
<th>name</th>
<th>requested state</th>
<th>instances</th>
<th>memory</th>
<th>disk</th>
<th>urls</th>
</tr>
</thead>
<tbody>
<tr>
<td>push2cloud-example-api-1.0.0</td>
<td>started</td>
<td>1/1</td>
<td>512M</td>
<td>512M</td>
<td>push2cloud-example-api-iot</td>
</tr>
<tr>
<td>push2cloud-example-host-2.0.0</td>
<td>started</td>
<td>1/1</td>
<td>512M</td>
<td>1G</td>
<td>push2cloud-example-host-1o</td>
</tr>
</tbody>
</table>
```

CloudBook
Modular extensible design

- Create customer tailored workflows
- Add your own plugins
- Run it as a REST based service
- Add runtime monitoring
- New platform adapters
- …
Main outcomes

- Faster deployments (>2x)
  - Minimize required steps (desired vs. actual state)
  - Parallelized operations

- Reliable deployments
  - Timeouts and retries
  - Grace period

- Flexible & extensible
  - Easy to adapt and build your custom workflows
  - Easy to integrate in any tooling (CI/CD, build tools, ...)
  - Write your own extension thanks to modular design
Open Source

www.push2.cloud

github.com/push2cloud

@Push2_cloud