Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Pivotal Cloud Foundry – Manifests , Environment Variables and AutoScaling

People matter, results count.

Objectives of PCF

- Purpose:
  - To learn pivotal cloud foundry Manifest yml , environment variables and auto scaling.
- Product:
  - Manifests
  - Environment Variables
  - Auto Scaling
- Process:
  - Understand manifest, environment variables and auto scaling.

Table of Contents

MANIFEST

Cloud Foundry Manifest File

- Describes the application deployment options
  - Automates subsequent deployments
  - Same options as **cf push** command
    - Plus options only available in the manifest
- Default name : **manifest.yml**
  - **YAML*** format
  - Human friendly data serialization standard
  - Supported by many programming languages
  - Less verbose than XML, similar to JSON
  - http://www.yaml.org

Create using your favorite text editor

* *YAML Ain't Markup Language*

Using a Manifest with Push

- cf push automatically detects manifest
  - In current directory or parent directories
  - Expects file named manifest.yml
  - Override with -f option
  - cf push -f dev-manifest.yml
  - Or ignore with --no-manifest option.
- No manifest found?
  - cf push will default all deployment options
  - Not the best choices
  - Different to previous version of cf (which prompter)

YAML Format

- 3 dashes
  - Indicates start of document
- Indent with spaces, not tabs!
  - Determines hierarchy
  - Each indent 2 spaces
  - "-" defines "group"
- Syntax: property : value pairs
- #starts a one line comment

```
---
applications:
- name: nodetestdh01
  memory: 64M
  instances: 2
  host: crn    # comment
  domain: cfapps.io
  path: .
  # comment

- name: nextapp    # group 2
  memory: 256M
  ...
```

- See http://www.yaml.org/spec/1.2/spec.html

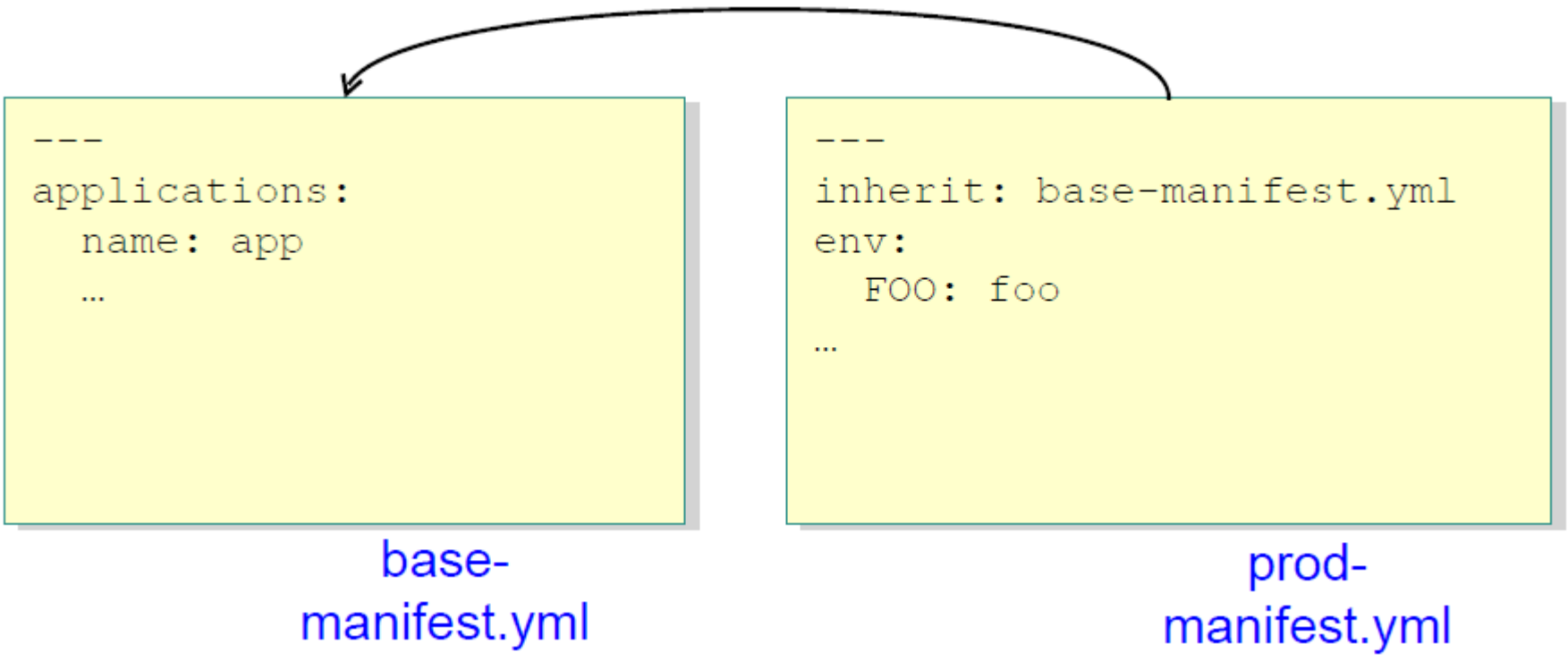
manifest.yml Example

```
---
applications:
- name: cf-node-demo
  command: node app.js
  memory: 128M
  instances: 1
  host: demo-${random-word}
  domain: cfapps.io
  path: .
```

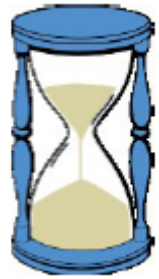
- **Applications:** can describe one or more applications
- **Name:** of the app - used in commands
- **Command:** command to run (optional)
- **Memory:** ceiling / **instances** to run
- **Host:** your choice, must be unique (within domain)
  - Tip: ${random-word}
- **Path:** to executable

Manifest inheritance

- You may wish to have multiple manifest for an App
  - Different manifests for each space
- One manifest can "inherit" from another



Specifying Timeouts

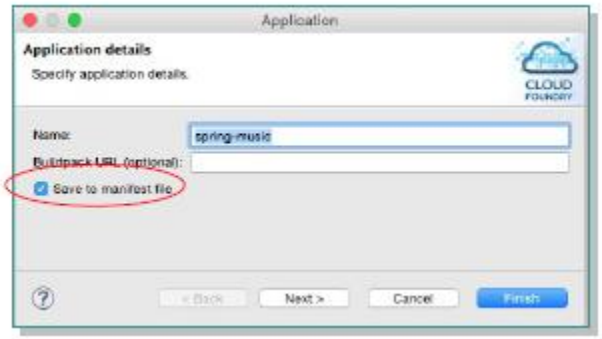


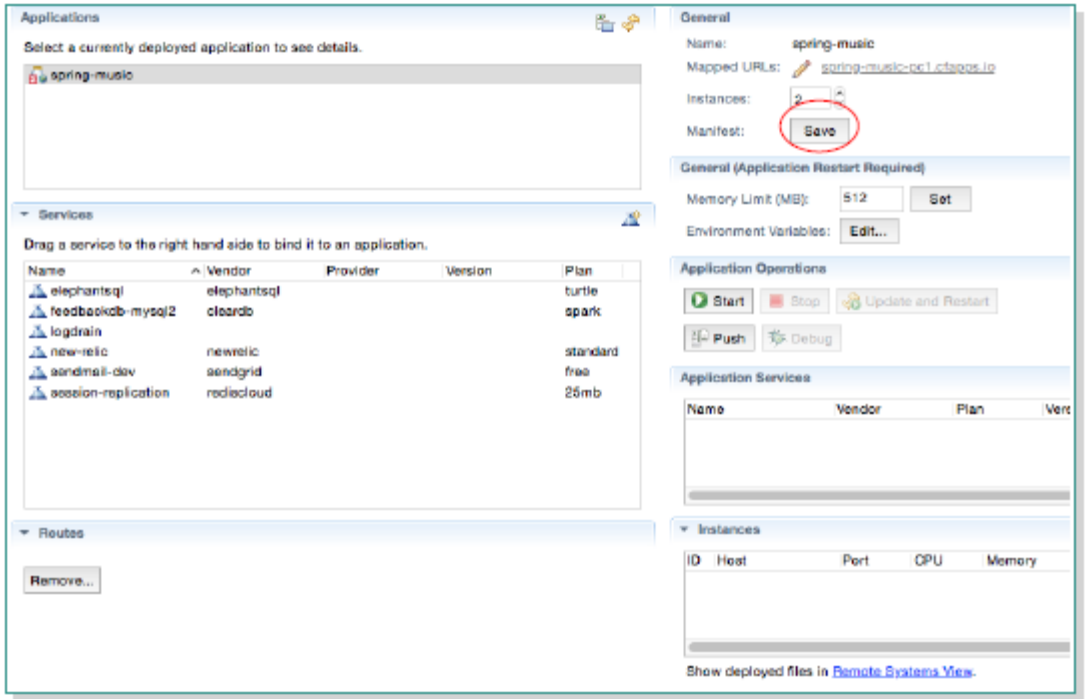
- Can use the manifest to specify timeouts for push, staging or startup
  - When you know your application is too large for the defaults

```
---
applications:
- name: nodetestdh01
  memory: 64M
  instances: 2
  host: crn
  domain: cfapps.io
  path: .
  timeout: 120  # 120 secs
  env:
    # Set to 20 and 10 mins
    CF_STAGING_TIMEOUT: 20
    CF_STARTUP_TIMEOUT: 10
```

Manifest and STS - I

- STS users can create a manifest
  - Checkbox during push
  - Or push an application has been pushed
    - Applications tab of Server properties





Manifest and STS - II

- Manifest created using same info as original push
  - Note{} in syntax (valid but optional)

- WARNING:
  - STS does not use manifest when pushing
  - Even if it has generated one

```
---
applications:
- name: dlbsmvc
  memory: 512M
  host: dlb-spring-mvc-demo
  domain: cfapps.io
  env: {
    }
  services: {
    }
```

Note on Spaces

- A space cannot be specified in a manifest file
  - Set first: cf target -s development
- To determine the current space
  - Just run: cf target

Manifest vs CLI

- A manifest reduces the amount of typing when deploying via CLI
  - Purpose is to makes deployment easily repeatable

- Options specified via CLI override options specified via manifest
  - **Example:** cf push my-app -i 8 -m 1024M
  - Deploys 8 instances with 1024M limit each, regardless of manifest settings

ENVIRONMENT VARIABLES

Environment Variables

- Key / value pairs
  - Used for anything you like
  - Specify via manifest

```
---
env:   # global, all apps
  spring_profiles_active: dev
  another_variable: foo
applications:
 …
```

```
---
applications:
- name: myapp
  memory: 256M
  instances: 1
  host: crn
  domain: cfapps.io
  env:   # this app only
    spring_profiles_active: dev
    another_variable: foo
```

- Or via command line
- cf set-env <app-name> <env-var-name> [<value>]
- Requires re-staging (i.e. cf restage or cf push) to take effect
- Or use App Manager or Eclipse plug-in

Environment Variables - Precedent

- Environment variables via manifest take precedent over CLI
  - Opposite from the push options defined earlier!
- Example
  - cf set-env app Foo fromCLI
  - cf push
  - Result? 'fromManifest'!

```
---
env:
  FOO: fromManifest
applications:
  name: app
```

- Use cf push app --no-manifest to bypass manifest values

Environment Variables - Persistence

- Environment variables retain their values
  - Whether application is running or not
- To view use cf env <app>
- Use cf unset-env <app> <var> to remove
- If changed while app is running
  - Use cf restage <app> to make change take effect

Environment Variables - Accessing

- CF environment variables are available to applications
  - Appear like any other environment variable

- Access via...
  - Java: System.getenv("some_variable");
  - Ruby: ENV['some_variable']
  - Node.js: process.env.some_variable

Environment Variables – VCAP_APPLICATION

- Information on memory, instances
  - JSON formatted object (described later):

```
{"instance_id":"451f045fd16427bb99c895a2649b7b2a",
 "instance_index":0,
 "host":"0.0.0.0",
 "port":61857,
 "started_at":"2013-08-12 00:05:29 +0000",
 "started_at_timestamp":1376265929,
 "start":"2013-08-12 00:05:29 +0000",
 "state_timestamp":1376265929,
 "limits":{"mem":512,"disk":1024,"fds":16384},
 ...
}`
```

Environment Variables – VCAP_SERVICES

- Information on all bound services
  - JSON formatted object (described later):

```
{ "elephantsql": [
  { "name": "elephantsql-c6c60",
    "label": "elephantsql-n/a",
    "tags": [ "postgres", "postgresql", "relational" ],
    "plan": "turtle",
    "credentials": {
      "uri": "postgres://PHxTPn@babar.elephantsql.com:5432/selmbd"
    }
  }  ],
  "sendgrid": [
  { "name": "mysendgrid",
    "credentials": {
      "username": "QvsXMbJ3rK",
      "password": "HCHMOYluTv"
    }
  } ] }
```

Integration with Spring

- spring_profiles_active
  - "Activates" a profile in a Java/Spring application.

```
---
env:
  spring_profiles_active: dev
  another_variable: foo
applications:
 …
```

Environment Variables Management

- Not all environment variables can be changed
  - Those set by CF runtime cannot
    - Such as **VCAP_SERVICES** (set by service binding)
    - See URL below for list of variables set by runtime

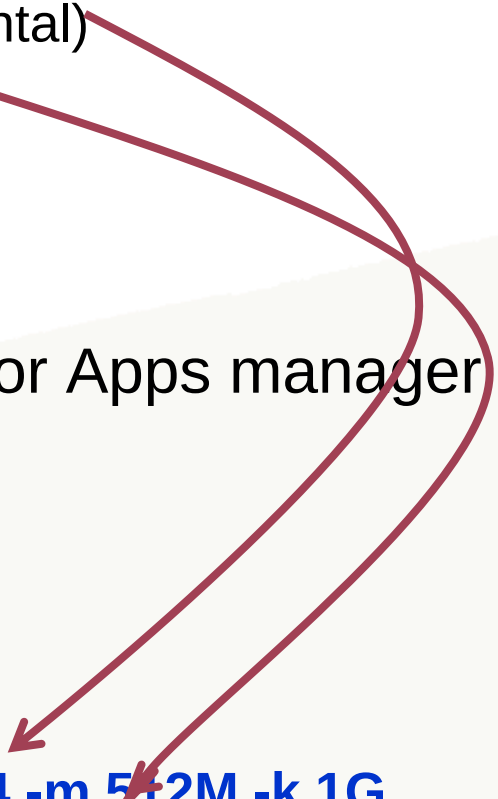
- To view environment variables:
  - **cf env [app-name]**
  - Displays user defined and system defined variables
  - Some variables only available to running instances

http://docs.pivotal.io/pivotalcf/devguide/deploy-apps/environment-variable.html

SCALING

Scaling

- Allows updating application to adjust to changes in load
  - Update instances (horizontal)
  - Update memory (vertical)



- Done from CLI, Eclipse or Apps manager
- From CLI
  - **cf scale <app-name>**
  - Reports current scaling
  - **cf scale <app-name> -i 4 -m 512M -k 1G**
  - make 4 instances, each with 512 Meg memory, 1 Gig disk space.

Applications and Resource Limits

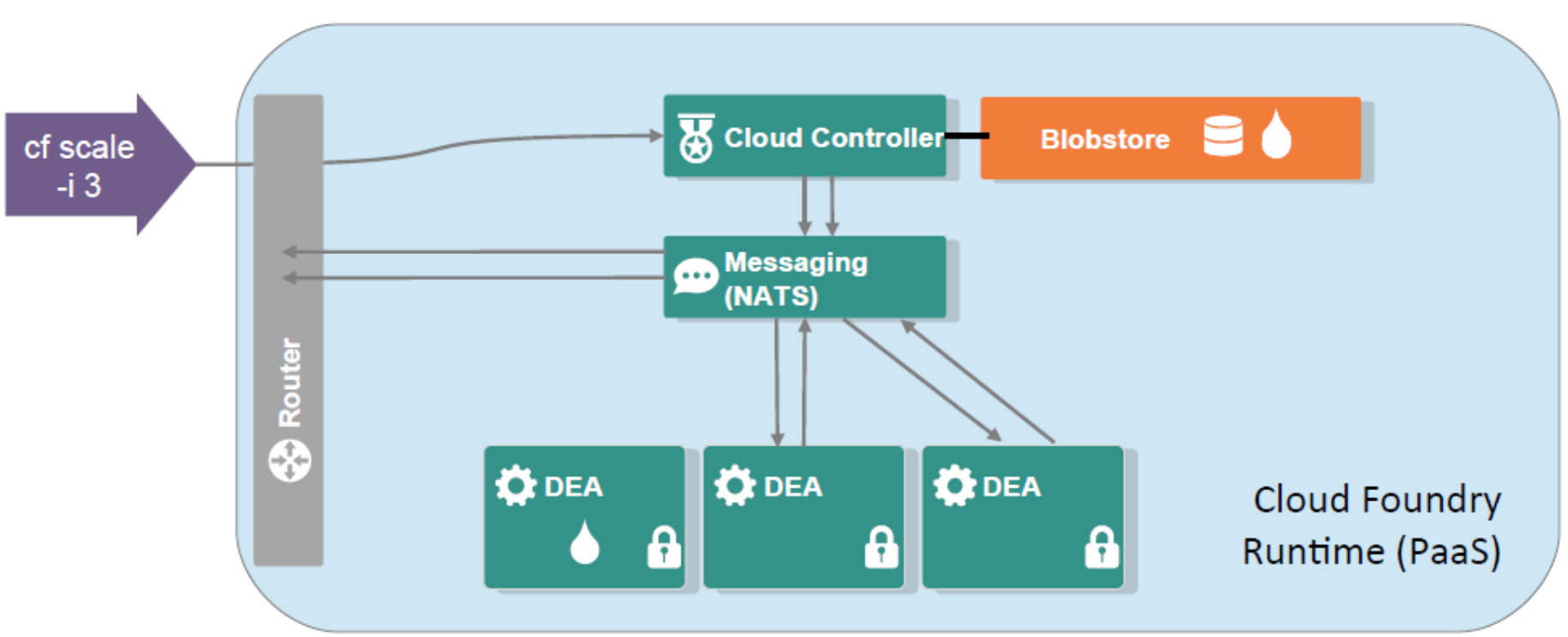
- When an application is pushed/scaled
  - Disk and memory limits are specified
- Application may not exceed its limits
  - Will fail instead
  - If limits exceeded at start-up: **flapping**
    - Repeating sequence of fail, restart, until timed out
- Quotas limit disk and memory usage by all applications in a space and/or an organization
  - Attempt to push/scale-up refused

How is CPU Allocated?

- Memory and local-disk limits can be set, what about CPU?
- "Fair-share" CPU allocation based on memory allocation
  - Each DEA has 256 "shares" of CPU
  - Regardless of memory or cores
  - Allocation = **256 * container-memory-limit / total-DEA-memory**
- Example
  - 1G application: **cf push myapp -m 1G**
  - Container size = instance size = 1G
  - DEA size = 32G (set by OPs)
  - CPS shares = 8 (**256 * 1 /32 =8**)

Scaling an Application

- Cloud Controller starts new instances or stops surplus instances
  - Load-balancing router shares load across all instances



What is I re-push?

- Subsequent **cf push** commands override the number of instances
  - Based on 1) CLI and 2)manifest
  - IF CLI/manifest do not specify instances, previous values stand
  - Application restarted

- Recommendation:
  - Use manifest to store 'default' scaling settings
  - OR omit scaling setting from the manifest.

File System Implementation

- Application instances (droplets) run in isolated Wa containers
  - Own resources, non-shared- such as files
  - For *safety and security*
- When an application is scaled up
  - New container, new isolated file-system
- When application ends (Stopped, failed or scaled down)
  - Local file-storage is destroyed with container

- Implication
  - *Do not rely on files: transient and not shareable*

Summary

- Use of Environment variables
- All about application manifests
- How scaling happens

Recap

Scaling

Manifest

YAML

environment

cf set-env

cf env

cf unset-env

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

People matter, results count.

About Capgemini

With more than 130,000 people in 44 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2012 global revenues of EUR 10.3 billion.

Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

www.capgemini.com