Manifests, Environment Variables, and Scaling

A closer look at practical Cloud Foundry usage

Manifests, Environment Variables, and Scaling

Pivotal

Overview

- After completing this lesson, you should be able to:
  - Define a Manifest
  - Set Environment Variables
  - Understand how Scaling works

Pivotal™

Roadmap

- **Manifest Files**
- Environment Variables
- Scaling

Pivotal™

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 prompted)

YAML Format

- 3 dashes
  - Indicate 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

Pivotal

Manifest Inheritance

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

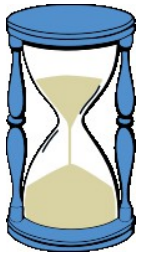
```
---
applications:
  name: app
  …
```

base-manifest.yml

```
---
inherit: base-manifest.yml
env:
  FOO: foo
…
```

prod-manifest.yml

Specifying Timeouts

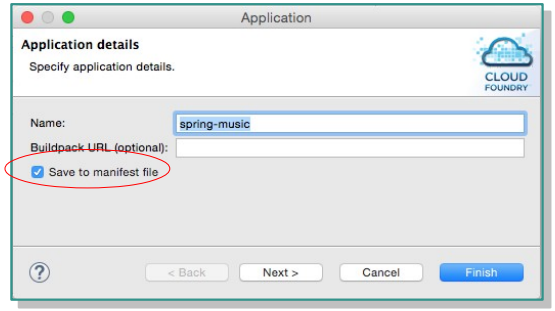


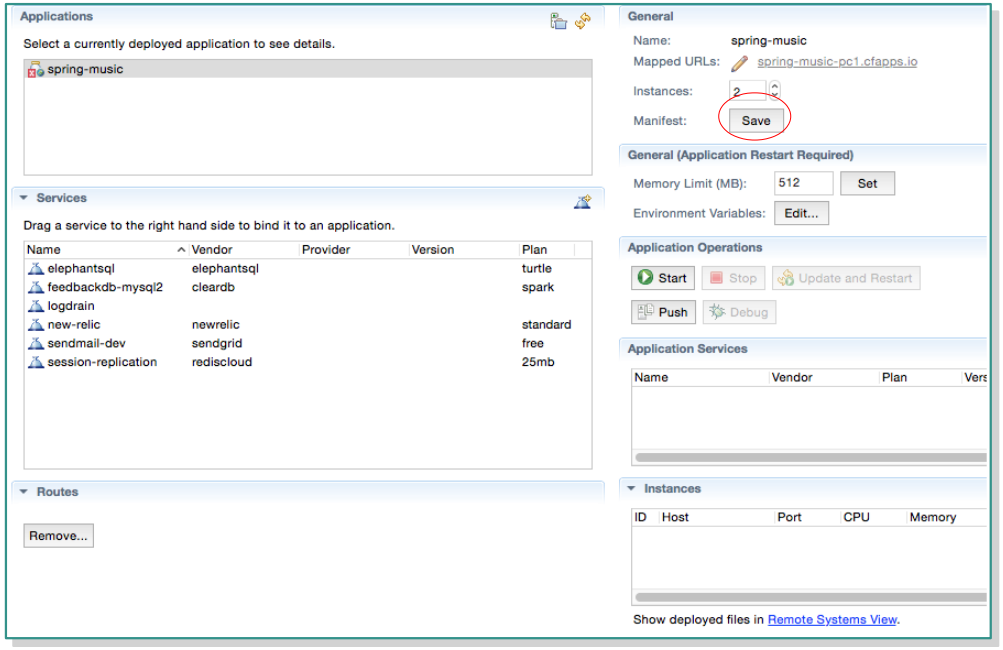
- Can use the manifest to specify timeouts for push, staging or startup
  - When you know your application is too large for the defaults

```
---
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
```

Manifests and STS – I

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





Manifests and STS – II

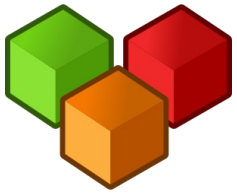
- Manifest created using same info as original push
  - Note { } in syntax (valid but optional)

- 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: {
    }
```

Pivotal

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 make 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 1024 M limit each, regardless of manifest settings

Pivotal™

Roadmap

- Manifest Files
- **Environment Variables**
- Scaling

Pivotal™

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

Pivotal

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

Pivotal™

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`

Pivotal™

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

Pivotal™

Roadmap

- Manifest Files
- Environment Variables
- **Scaling**

Pivotal™

Scaling Applications

- 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

Pivotal™

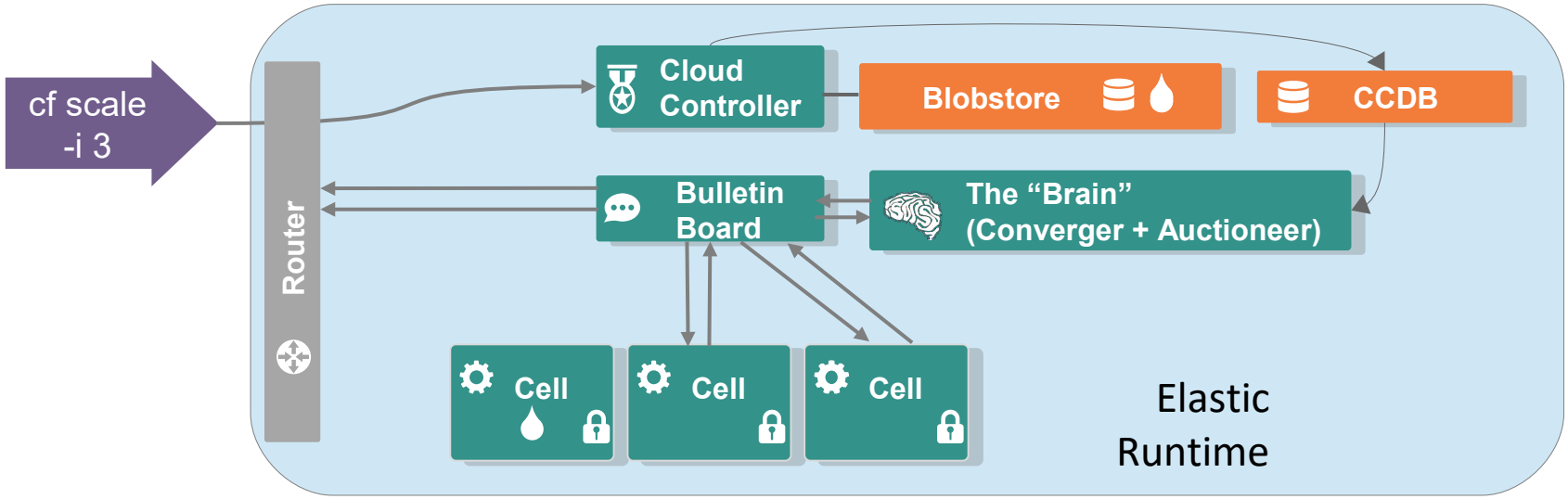
How is CPU Allocated?

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

Pivotal™

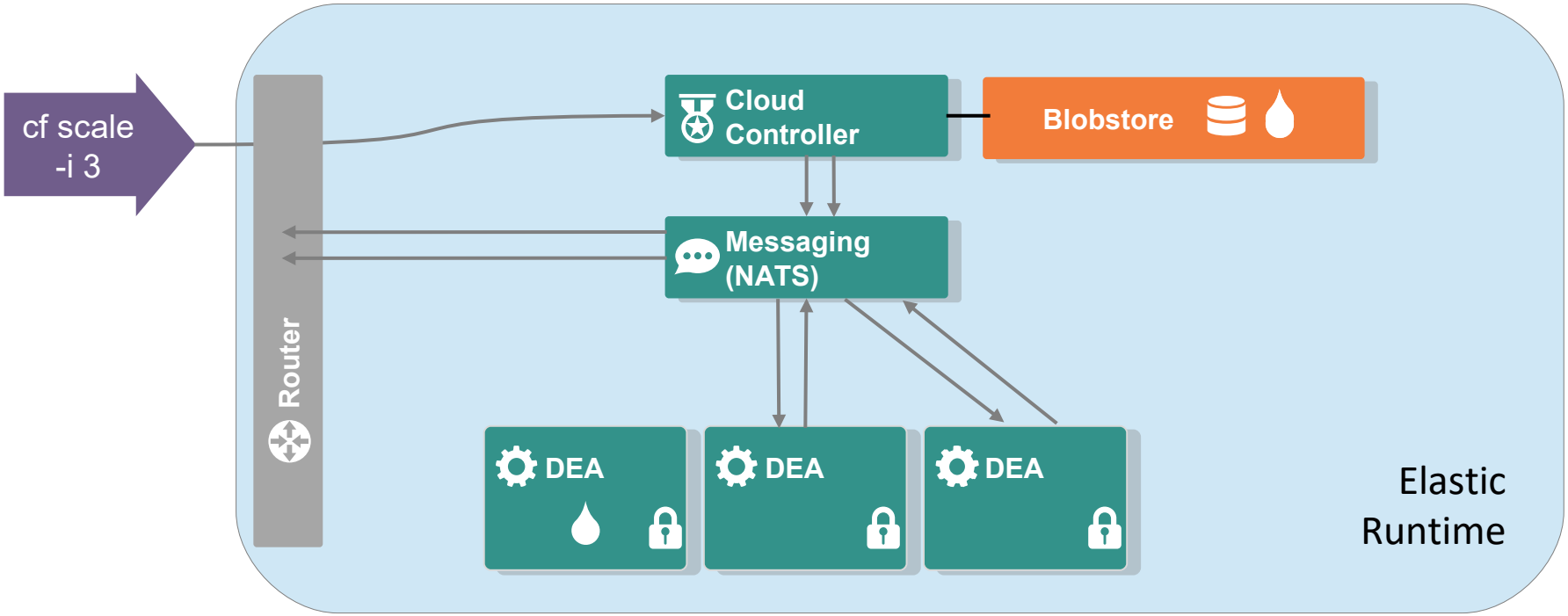
Scaling an Application

- Cloud Controller indicates instances should be started or stopped
  - Converger adds tasks to start/stop instances to central Bulletin Board
  - Auctioneer determines the least busy Cells to use



Scaling an Application (DEA)

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



What if 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 settings from the manifest.

Pivotal™

File System Implications

- Application instances (Droplets) run in isolated Warden 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*

Pivotal™

Summary

- After completing this lesson, you should have learned:
  - Use of environment variables
  - All about application manifests
  - How scaling happens

Pivotal™

Lab

Using Environment Variables and Scaling Applications

Pivotal™