AARHUS UNIVERSITET

Microservices and DevOps

DevOps and Container Technology

Seminar 3 Exercises

Henrik Bærbak Christensen

AARHUS UNIVERSITET

Scaling

AARHUS UNIVERSITET

Scaling Exercise 1

- Solve Iteration 5's 'horizontal-scaling-failing' exercise

Exercise 'horizontal-scaling-failing'

Learning Goal: To demonstrate the *inability* of SkyCave to do scale horizontally.

Exercise work in progress

Note: This exercise may strain your laptop quite a bit. If you can spare it, increase Memory (6GB) in the VM.

We aim to make a stack running 2-3 skycave daemons as replicas, but configured to accept any login credentials; and next generate some load on it.

Experiment Part I: Setting up

- There is a null-object implementation of the SubscriptionServiceConnector, 'cloud.cave.doubles.NullSubscriptionService'. Review the code to see how it implements the 'authorize()' method.
- Create a new CPF 'swarm-db-load.cpf' which configures the skycave for running in a stack with a Redis and the above mentioned NullSubscriptionService.
- Copy your 'swarm-db.yml' so your 'daemon' uses that CPF.
- Create a new image, upload to Hub, and start the stack.
- *Small Steps*: Run your 'cmd' against the locally deployed stack and verify that any credentials are valid:

```
gradle cmd -Pid=anything -Ppwd=anything
```

- *Load testing*: Now try the load generator

```
gradle load
```

AARHUS UNIVERSITET

Scaling Exercise 2

- Have a go at the ‘session-database’ exercise.

AARHUS UNIVERSITET

CI and Pipelines

AARHUS UNIVERSITET

CI Exercise 1

- Start on

Exercise 'bitbucket-pipeline-ci' (*) [M 80]

We have a lot of trickery going to move from a code change, until it provides value for our customers, and while gradle and docker IaC helps a lot, we still do a lot by hand. In this exercise, we start automation in a *build pipeline*, but focuse on the production of a 'release candidate deployment unit'

In this exercise, you should write a BitBucket IaC pipeline, which does Continuous Integration, to produce a Docker Hub image release candidate.

Requirements: Your pipeline must (at least) consist of four stages:

- Advice:
  - *Take small steps (I had 45 failed commits ☹)*
  - Or maybe have a look at their ‘try locally’ guide (?)

AARHUS UNIVERSITET

Exam

AARHUS UNIVERSITET

Exam Exercises

- My dryrun of the exam exercise showed that I spend quite some time in the ‘use command pattern’ part. So…

Exercise 'exam-command'

A feature of SkyCave that *will be used at the exam*, is the ability to dynamically add new commands to the 'cmd' by updating the server only. To do that, the SkyCave daemon utilizes the **Command Pattern**.

Requirements:

- Review the Command pattern on wikipedia or some other design pattern resource.
- Review the 'JumpCommand', 'HomeCommand', etc. in the `cloud.cave.extension` package. Note that Command classes *must* be in this package.
- Create a **PlusCommand** so a running 'cmd' can make simple two integer additions, and provide output ala:

```
== Welcome to SkyCave, player Mikkel ==
Entering command loop, type "q" to quit, "h" for help.
> exec PlusCommand 64 32
The result is: 96.

> exec PlusCommand 999 777
The result is: 1776.
```

Exercise 'exam-command-http'

The Command you develop at the exam *will be contacting a REST service, provided by me*. Be sure you have a working knowledge of the UniRest HTTP client (or whatever you choose) framework, and can find code blocks to reuse from your quote service and subscription service implementations.

Alternative

- Look into CD instead…

Exercise 'operations-pipeline-cd' [M 40]

You have to design some kind of actual deployment step, which ensures that your production cave is updated. Either as a direct consequence of the pipeline, like a ssh command on the production server (proactive approach), OR, as part of a scheduled/regular update, like a cron job on the production server (reactive approach).

Requirements: A successfull run of the CI pipeline that ends in a new release of an update skycave image, should result in your production servers being updated.