LESSONS LEARNED MIGRATING AN EXISTING PRODUCT TO A MULTI TENANT CLOUD NATIVE ENVIRONMENT

Natalia Angulo / github.com/angulito / @nangulito

Carlos Sanchez / csanchez.org / @csanchez

Natalia / Software Developer Engineer

Maths, coding

Carlos / Principal Scientist

OSS contributor, Jenkins Kubernetes plugin

Adobe Experience Manager Cloud Service

ADOBE EXPERIENCE MANAGER

An existing distributed Java OSGi application

Using OSS components from Apache Software Foundation

A huge market of extension developers

AEM ON KUBERNETES

Running on **Azure**

37+ clusters and growing

Multiple regions: US, Europe, Australia, Singapore, Japan, India, more coming

Adobe has a **dedicated team** managing clusters for multiple products

Customers can **run their own code**

Cluster permissions are **limited** for security

ie. Traffic leaving the clusters must be encrypted

AEM ENVIRONMENTS

- Customers can have multiple AEM environments that they can self-serve
- Each customer: 3+ Kubernetes namespaces
- Each environment is a micro-monolith ™

Using namespaces to provide a scope

- network isolation
- quotas
- permissions

SERVICES

Multiple teams building services

Different requirements, different languages

You build it you run it

Using APIs or Kubernetes operator patterns

ENVIRONMENTS

Using init containers and (many) sidecars to apply division of concerns

SIDECARS

- Service warmup
- Storage initialization
- httpd fronting the Java app
- Exporting metrics
- fluent-bit to send logs
- Java threaddump collection
- Envoy proxying
- Autoupdater

SERVICE WARMUP

Ensure that the service is ready to serve traffic

Probes the most requested paths for lazy caching

Without requiring expensive starts

FLUENT-BIT TO SEND LOGS

Using a shared volume to send logs to a central location

Configured independently from the application

ENVOY PROXYING

Using Envoy for traffic tunneling and routing

Enables dedicated ips per tenant and VPN connectivity

AUTOUPDATER

Runs on startup and updates any configuration needed

Allows patching the whole cluster fleet live

OPERATORS

AEM ENVIRONMENT OPERATOR

Overarching operator that manages lifecycle of environments

AEM ENVIRONMENT OPERATOR

An operator to rule them all

Launches jobs pre/post environment creation

Reconciles with other **internal operators**

FLUXCD HELM OPERATOR

https://fluxcd.io/

FLUXCD HELM OPERATOR

Allows **managing Helm charts** using **declarative** state vs imperative commands

Integrated with our operators to manage the lifecycle of the Helm releases

and to **gather state** from the Helm operations

ARGO ROLLOUTS OPERATOR

https://argoproj.github.io/rollouts/

ARGO ROLLOUTS OPERATOR

Provides **advanced deployment strategies**

Canary, Blue/Green, A/B testing, etc.

Automated rollbacks

SCALING AND RESOURCE OPTIMIZATION

Each customer environment (17k+) is a micro-monolith ™

Multiple teams building services

Need ways to scale that are orthogonal to the dev teams

Kubernetes workloads can define resource requests and limits:

Requests:

how many resources are guaranteed

Limits:

how many resources can be consumed

And are applied to

CPU

Memory

Ephemeral storage

Memory: limit enforced, results in Kernel OOM killed

Ephemeral storage: limit enforced, results in pod eviction

CPU REQUESTS IN KUBERNETES

It is used for scheduling and then a relative weight

It is not the number of CPUs that can be used

1 CPU means it can consume one CPU cycle per CPU period

Two containers with 0.1 cpu requests each can use 50% of the CPU time of the node

CPU LIMITS IN KUBERNETES

This translates to cgroups quota and period.

Period is by default 100ms

The limit is the number of CPU cycles that can be used in that period

After they are used, the container is throttled

ARM ARCHITECTURE

15-25% cost savings for the same performance

Easy switch for containerized Java

JAVA AND KUBERNETES

WHAT IS THE DEFAULT JVM HEAP SIZE?

1. **75% of container memory** (< 256 MB)
2. 75% of host memory
3. **25% of container memory** (> 512 MB)
4. 25% of host memory
5. **127MB** (256 MB to 512 MB)

Typically can use up to 75% of container memory

Unless there is a lot of off-heap memory used (ElasticSearch, Spark,...)

JVM takes all the memory on startup and manages it

JVM memory use is hidden from Kubernetes, which sees all of it as used

Set request and limits to the same value

WHAT IS THE DEFAULT JVM GARBAGE COLLECTOR?

1. **SerialGC** *<2 processors & < 1792MB available*
2. **ParallelGC** *Java 8*
3. **G1GC** *Java >=11*
4. ZGC
5. ShenandoahGC

KUBERNETES AUTOSCALING

KUBERNETES AUTOSCALING

- Cluster Autoscaler
- Horizontal Pod Autoscaler
- Vertical Pod Autoscaler

KUBERNETES CLUSTER AUTOSCALER

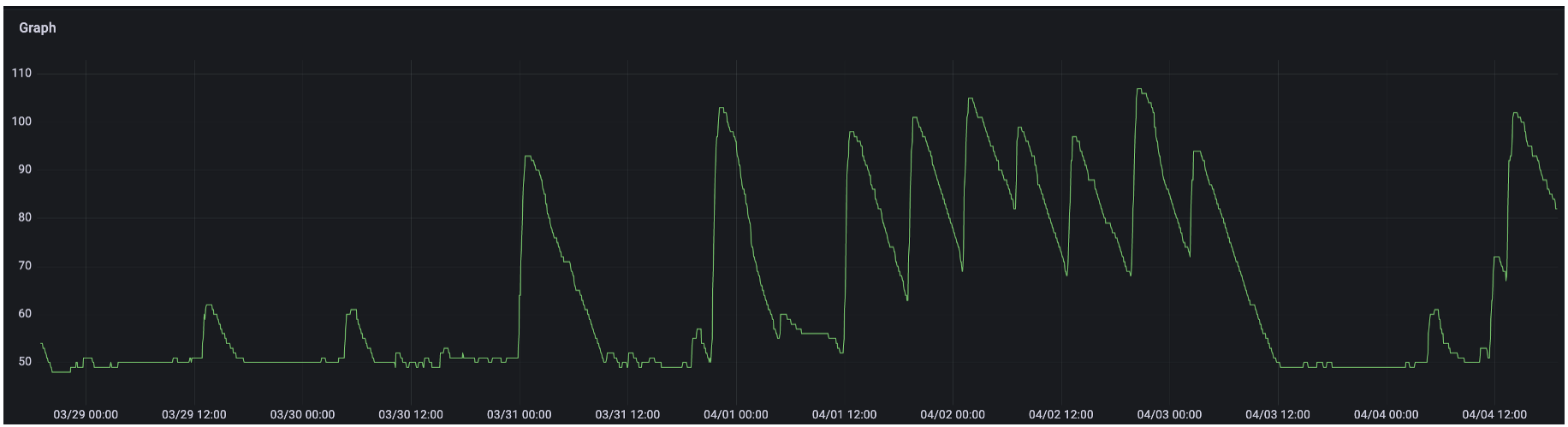
Automatically increase and reduce the cluster size

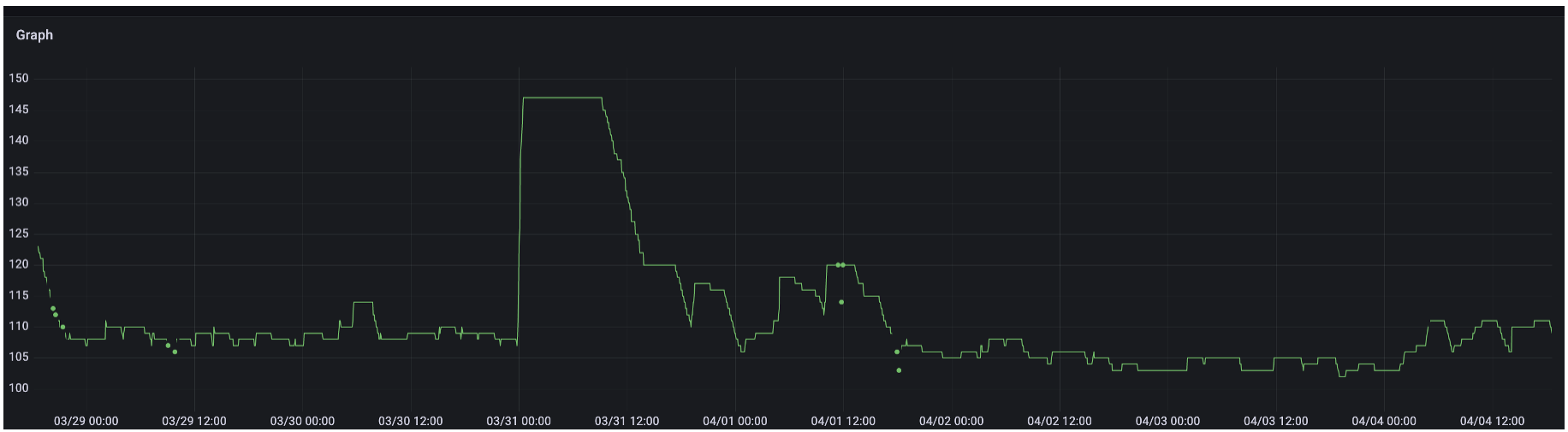
Based on CPU/memory requests

KUBERNETES CLUSTER AUTOSCALER

Max nodes managed at the cluster level

Savings: 30-50%


Graph
110
100
90
80
70
60
50
03/29 00:00
03/29 12:00
03/30 00:00
03/30 12:00
03/31 00:00
03/31 12:00
04/01 00:00
04/01 12:00
04/02 00:00
04/02 12:00
04/03 00:00
04/03 12:00
04/04 00:00
04/04 12:00


Graph
150
145
140
135
130
125
120
115
110
105
100
03/29 00:00
03/29 12:00
03/30 00:00
03/30 12:00
03/31 00:00
03/31 12:00
04/01 00:00
04/01 12:00
04/02 00:00
04/02 12:00
04/03 00:00
04/03 12:00
04/04 00:00
04/04 12:00

VERTICAL POD AUTOSCALER

Increasing/decreasing the resources for each pod

Requires **restart of pods** (automatic or on next start)

(next versions of Kubernetes will avoid it)

VPA

Only used in AEM **dev environments** to scale down if unused

Savings: 5-15%

HORIZONTAL POD AUTOSCALER

Creating more pods when needed

HPA

AEM scales on **CPU and http requests** per minute (rpm) metrics

⚠️ Do not use same metrics as VPA

CPU autoscaling is problematic

Periodic tasks can spike the CPU, **more pods do not help**

Spikes on **startup** can trigger a **cascading effect**

Savings: 50-75%

Easy to start in k8s, then optimize

Use patterns to decompose application: sidecars, init containers, new services,...

Resource optimization: tuning JVM CPU, memory, GC


nangulito


csanchez


angulito


carlossg