Continuous Development in the Cloud

DevOps, Continuous Delivery, Microservices, Docker and Clouds

15 December, 2017. Free University of Bozen-Bolzano, Italy

migrating IDM's big data platform to RDS and Elastic Beanstalk

IDM SÜDTIROL ALTO ADIGE

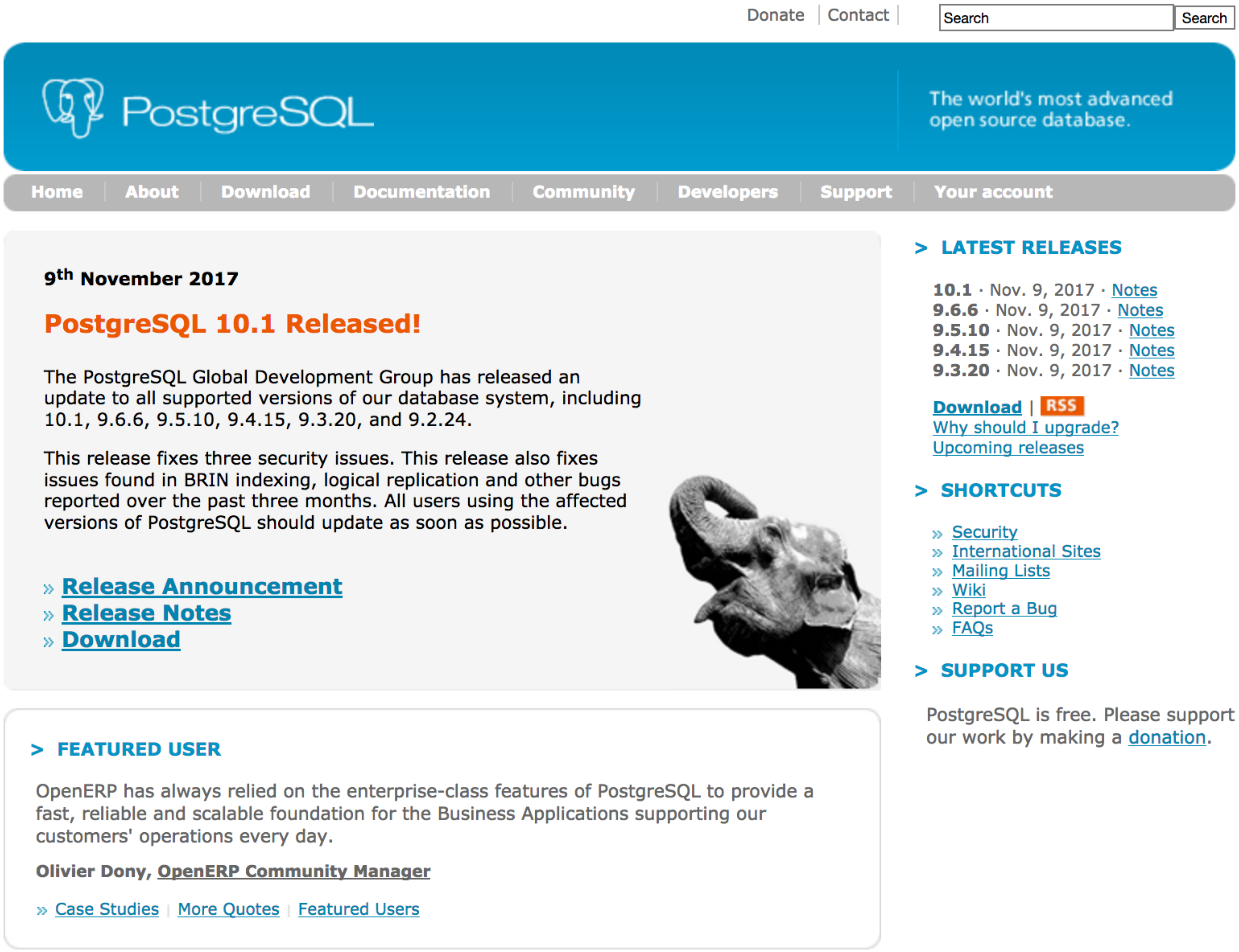
Chris Mair · www.1006.org

Outline

- some random praise about PostgreSQL - 1 slide
- some random info about AWS - 27 (will skip a few ;)
- IDM's Big Data Platform (BDP) - 3
- BDP migration experience - 10

some random praise
about **PostgreSQL**

PostgreSQL



9th November 2017

PostgreSQL 10.1 Released!

The PostgreSQL Global Development Group has released an update to all supported versions of our database system, including 10.1, 9.6.6, 9.5.10, 9.4.15, 9.3.20, and 9.2.24.

This release fixes three security issues. This release also fixes issues found in BRIN indexing, logical replication and other bugs reported over the past three months. All users using the affected versions of PostgreSQL should update as soon as possible.

» **Release Announcement**
» **Release Notes**
» **Download**

> FEATURED USER

OpenERP has always relied on the enterprise-class features of PostgreSQL to provide a fast, reliable and scalable foundation for the Business Applications supporting our customers' operations every day.

Olivier Dony, OpenERP Community Manager

» Case Studies | More Quotes | Featured Users

> LATEST RELEASES

10.1 · Nov. 9, 2017 · Notes
9.6.6 · Nov. 9, 2017 · Notes
9.5.10 · Nov. 9, 2017 · Notes
9.4.15 · Nov. 9, 2017 · Notes
9.3.20 · Nov. 9, 2017 · Notes

Download | RSS
Why should I upgrade?
Upcoming releases

> SHORTCUTS

» Security
» International Sites
» Mailing Lists
» Wiki
» Report a Bug
» FAQs

> SUPPORT US

PostgreSQL is free. Please support our work by making a donation.

some random info
about **AWS**

context

- 2003 - Amazon: "let's split datacenter operation from retail operation"

- 2006 - Amazon Web Services (AWS) starts as a public service; first big service: **S3** storage with HTTP-API

- 2008 - Chris discovers AWS and feels like a kid in a candy store :)

- 2016 - AWS is a 13E9 $/y revenue business and huge (e.g. Netflix, uses 800000 ± 20% cores in **EC2**)

S3, EC2, WTF?

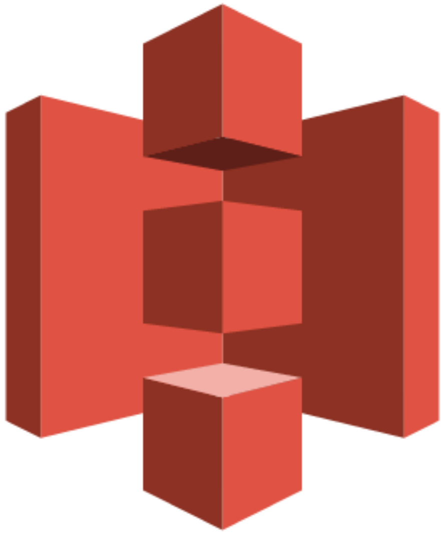
- **S3** stands for **S**imple **S**torage **S**ervice

- it is a **web service** (i.e. accessed through https) to store and retrieve data

  →

  this explains the name **AWS** (Amazon Web Services)

- lots of feature: data is stored redundantly for high durability, can do versioning, can be used for static web hosting, etc...



so, how does one use this?

- all services can be accessed as **web services**, i.e. via an open API (usually REST, earlier also SOAP)

- there are **SDK** for all sorts of programming languages and environments (Java, Node, etc...)

- there is also an AWS **CLI**, called aws

- there is a **web UI**, which is ideal for discovering things and administrate small fleets of services

about the AWS CLI

- AWS CLI is a command line client to access all of AWS services.
- it is written in Python and released on GitHub under the Apache License.
- it is in Debian's stable repo (package awscli).
- credentials are stored in ~/.aws/credentials

let's try S3 (with the CLI)

- **store** something:

```
[chris@pluto ~]$ cat hello.txt
Hello S3!
[chris@pluto ~]$ aws s3 cp hello.txt s3://1006.org/
upload: ./hello.txt to s3://1006.org/hello.txt
```

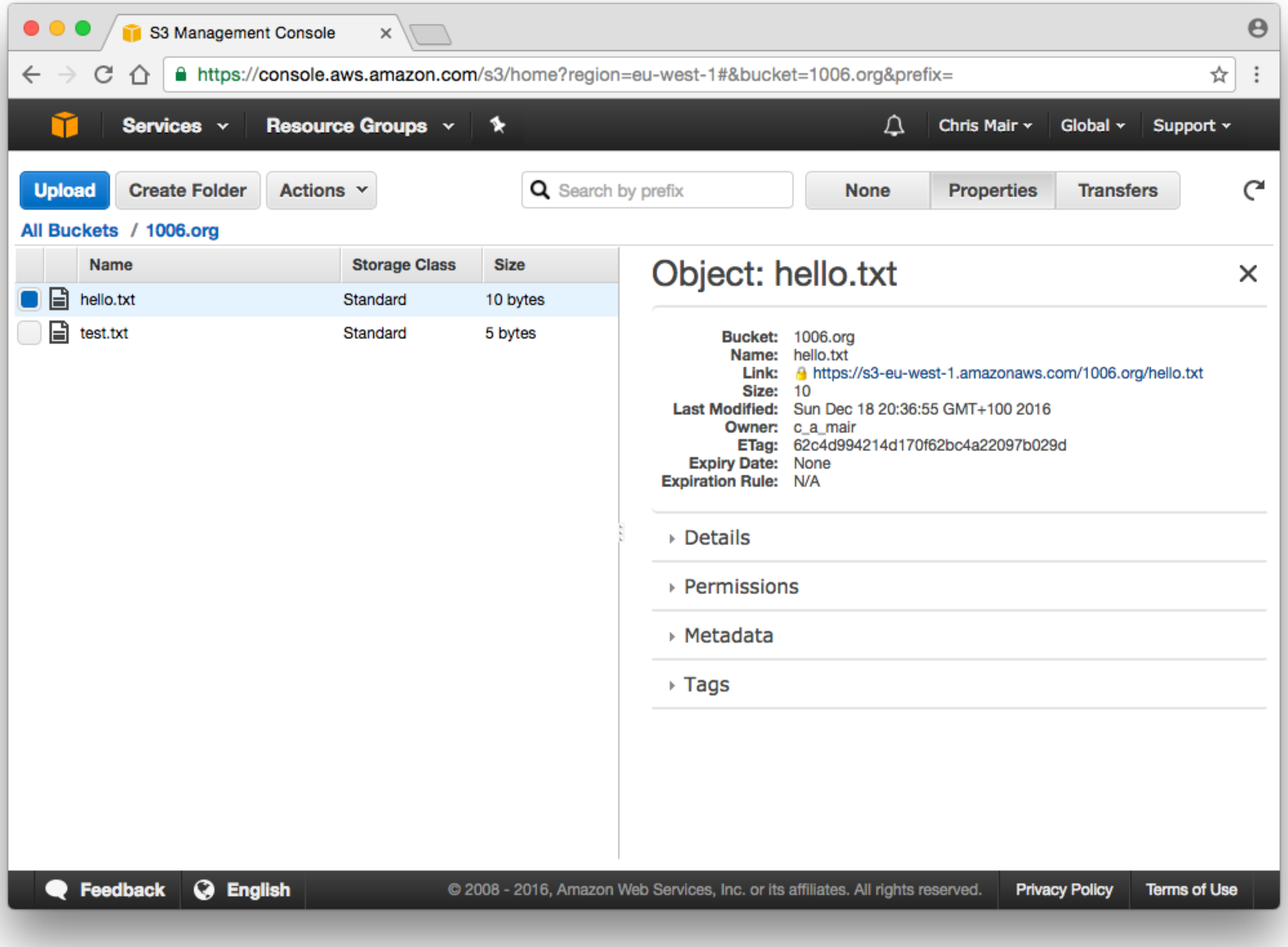
1006.org is a "bucket"

hello.txt is an "object"

- **retrieve** it somewhere else:

```
earth:~ chris$ aws s3 cp s3://1006.org/hello.txt .
download: s3://1006.org/hello.txt to ./hello.txt
earth:~ chris$ cat hello.txt
Hello S3!
```

Web UI, you said?



Name	Storage Class	Size
hello.txt	Standard	10 bytes
test.txt	Standard	5 bytes

Object: hello.txt

Bucket:	1006.org
Name:	hello.txt
Link:	https://s3-eu-west-1.amazonaws.com/1006.org/hello.txt
Size:	10
Last Modified:	Sun Dec 18 20:36:55 GMT+100 2016
Owner:	c_a_mair
ETag:	62c4d994214d170f62bc4a22097b029d
Expiry Date:	None
Expiration Rule:	N/A

how much $$$?

- funny that you ask, because **S3** is one of the more complex examples :)
  - $0.023 per GB month of (standard) storage in Ireland
  - $0.005 per 1000 stores
  - $0.004 per 10000 retrieves
  - $0.090 per GB of outgoing traffic to internet (free to AWS)
- lots of options (opt into infrequent access, reduced redundancy or large access times to reduce the cost)

argh!!! I'll never know how much I'm going to pay!

- don't worry, most services are easier to account for or have a fixed cost per time unit (unlike **S3**); my experience is that a typical bill is almost completely made of costs that can be easily computed/estimated

- the key point is flexibility and agility; you pay what you consume (or what you provisioned); it is very easy to scale up (or more often down!) and it's very easy to run stuff on things that would have been completely unaccessible to small companies just a few years ago

so...?

- I picked **S3** as an example because it was (sort of) the first service in AWS. And **S3 is** actually a web service

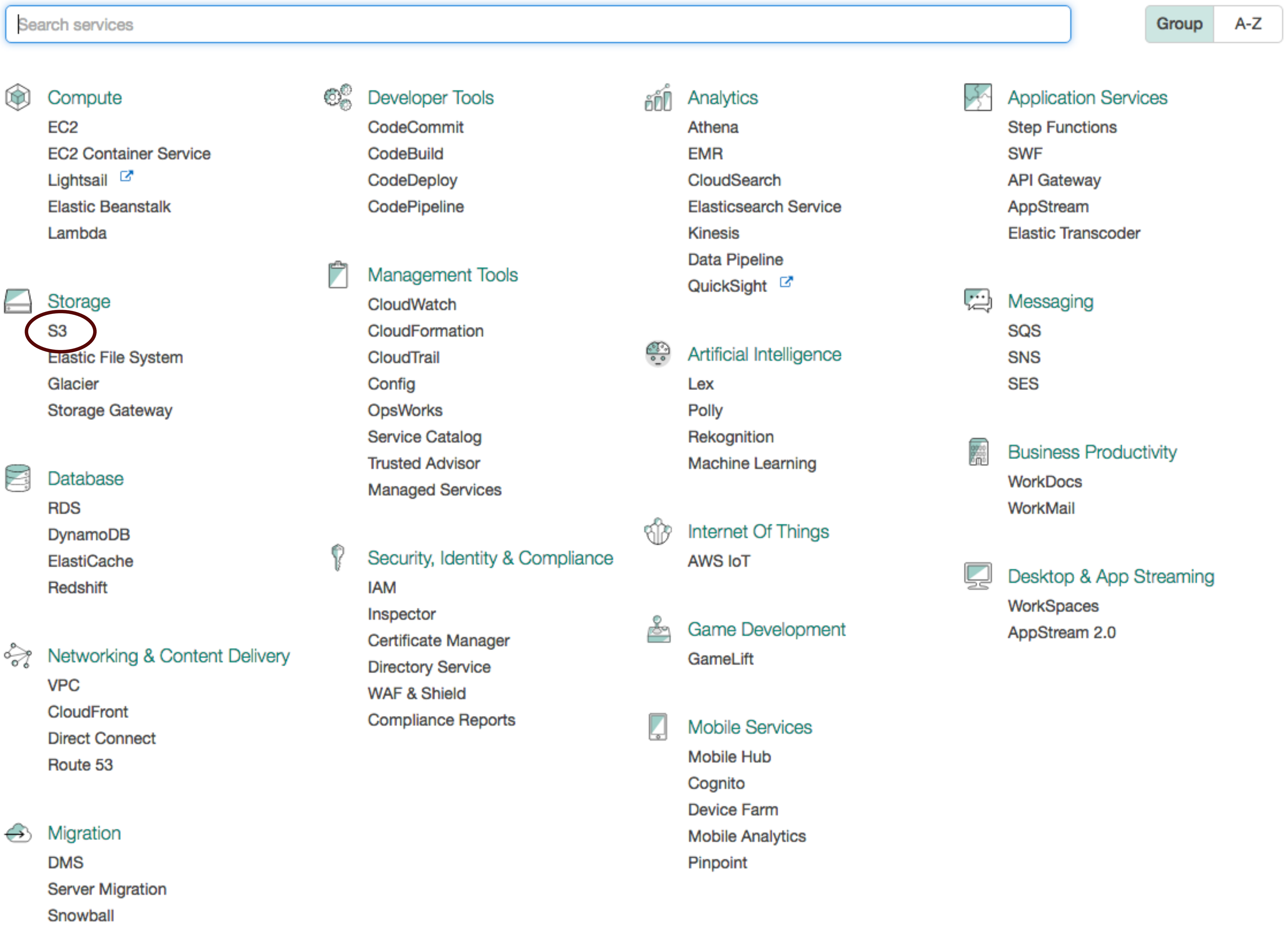
- during the last 10 years, AWS' offerings have grown to span a huge range: web services, leased resources (all kinds of hardware, VMs, storage, networking, software...)

- some of these have a fixed cost per month, others are charged by hour, or by the second, AWS Lambda even charges in units of 100ms (!) - you need to look into the **specific service**

- some confusion comes from the name AWS. A somewhat better name would be Amazon Big Candy Store for System Administrators and Devops or something like that :)

what are all these services, then?

you are here :)



Compute
- EC2
- EC2 Container Service
- Lightsail
- Elastic Beanstalk
- Lambda

Storage
- S3
- Elastic File System
- Glacier
- Storage Gateway

Database
- RDS
- DynamoDB
- ElastiCache
- Redshift

Networking & Content Delivery
- VPC
- CloudFront
- Direct Connect
- Route 53

Migration
- DMS
- Server Migration
- Snowball

Developer Tools
- CodeCommit
- CodeBuild
- CodeDeploy
- CodePipeline

Management Tools
- CloudWatch
- CloudFormation
- CloudTrail
- Config
- OpsWorks
- Service Catalog
- Trusted Advisor
- Managed Services

Security, Identity & Compliance
- IAM
- Inspector
- Certificate Manager
- Directory Service
- WAF & Shield
- Compliance Reports

Analytics
- Athena
- EMR
- CloudSearch
- Elasticsearch Service
- Kinesis
- Data Pipeline
- QuickSight

Artificial Intelligence
- Lex
- Polly
- Rekognition
- Machine Learning

Internet Of Things
- AWS IoT

Game Development
- GameLift

Mobile Services
- Mobile Hub
- Cognito
- Device Farm
- Mobile Analytics
- Pinpoint

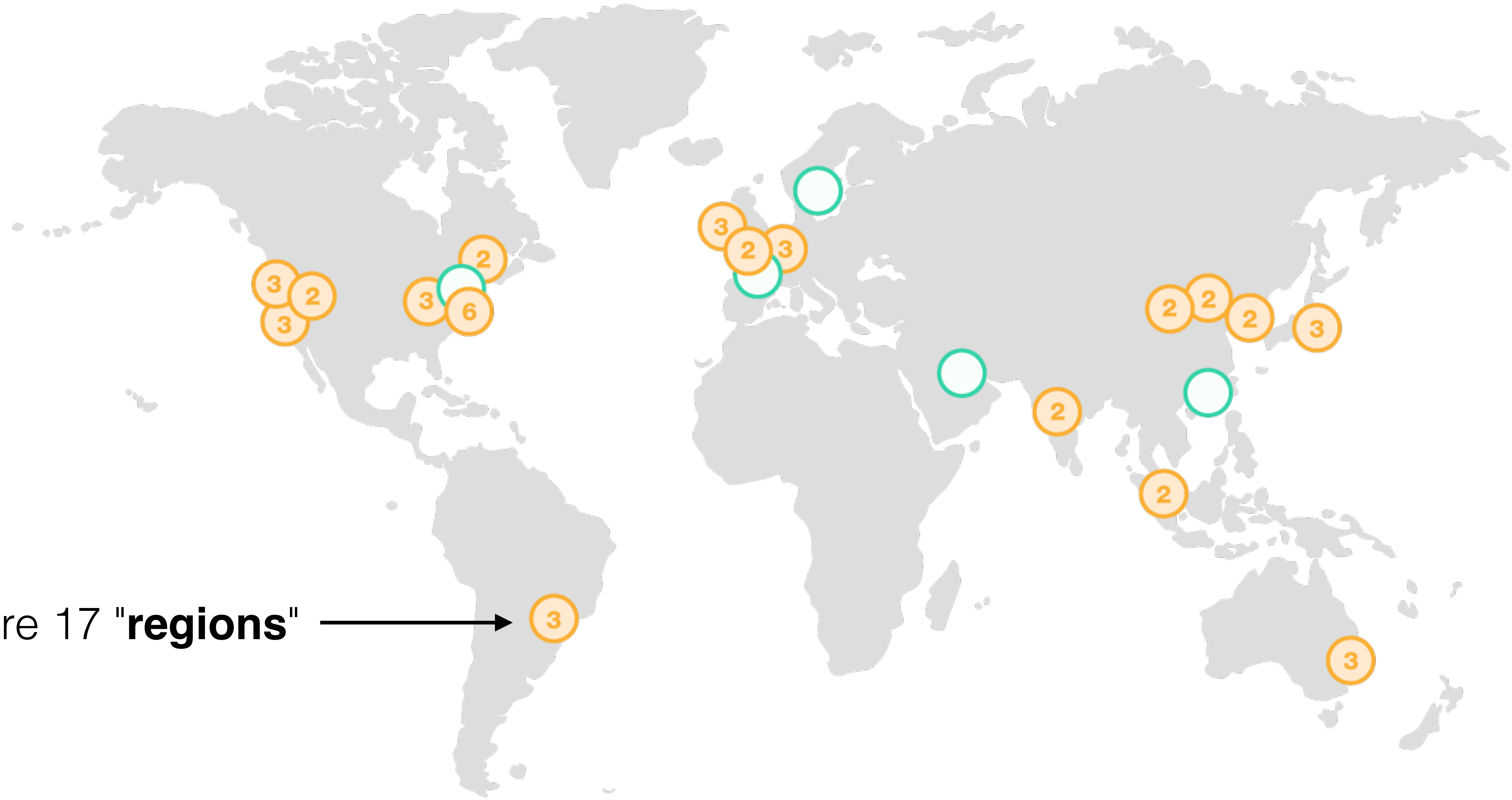
Application Services
- Step Functions
- SWF
- API Gateway
- AppStream
- Elastic Transcoder

Messaging
- SQS
- SNS
- SES

Business Productivity
- WorkDocs
- WorkMail

Desktop & App Streaming
- WorkSpaces
- AppStream 2.0

where is all this? 1/2



there are 17 "**regions**" →

each region has at least two independent data centers ("**availability zones**" - "**AZ**s")

where is all this? 2/2

- you need to pick the region where you want to use the service (except billing, which is always in the US).

- some services with high durability are always spanned across two AZs (e.g. **S3**), for some you pick the AZ too (e.g. **EBS**) and for some you choose if you want them in a single AZ or replicated into a second one (e.g. **RDS**)

- it's not true that you don't know where your data is - you actually do (at least the city ;).

IaaS 1/2

- let's look into **IaaS** (**I**nfrastructure **a**s **a** **S**ervice) offerings

- at the base of IaaS is **VPC** ("**v**irtual **p**rivate **c**loud") - this is software defined networking, with (private) subnets, gateways, ACLs and routing

- in a new account there is a standard configuration with a subnet for each AZ and default options, which is enough for basic use

- today, you always use VPC! older accounts still have a simplified network setup now called "classic" now; I mention this because you might see it in tutorials...

IaaS 2/2

- the service most directly associated with IaaS is **EC2** (**e**lastic **c**ompute **c**loud).

- in EC2 you can launch VMs, called **instances** that are **billed by the second**.

- instances will get a private IP on the subnet in the AZ they run in; to access them from the internet, you can allocate public IP addresses, called "**elastic IPs**" that can be dynamically attached to instances

- firewall/forwarding rules between elastic IPs and your instances can be defined using **security groups**

EC2

- to run an instance in **EC2**, besides a zillion details, you:
  - pick the OS
  - pick the instance type
  - choose the AZ/subnet
  - add block storage
  - configure a security group and elastic IP

EC2 - the OS

- **EC2** can run any OS from an **AMI** (**A**mazon **M**achine **I**mage) - it uses Xen internally.

- the default OS is Amazon Linux (a nice rolling release CentOS-like distribution), all the usual distributions are available in the AMI market place at no additional cost (e.g. Debian, CentOS)

- in the market place there are also AMIs for which you need to pay an extra cost per hour (e.g. Red Hat, Microsoft stuff, etc..)

- you can easily make you own AMI; you can upload an image, or you can configure a running instance, shut it down and clone its disk into an AMI

EC2 - the instance type

- you choose an instance type according to CPU generation, number of cores ("**vCPUs**"), RAM, and possibly extra hardware such as directly attached block devices ("instance disks") or GPUs

- it's not possible to freely choose the number of cores and RAM; However there are more than 80 instance types, so there is some choice...

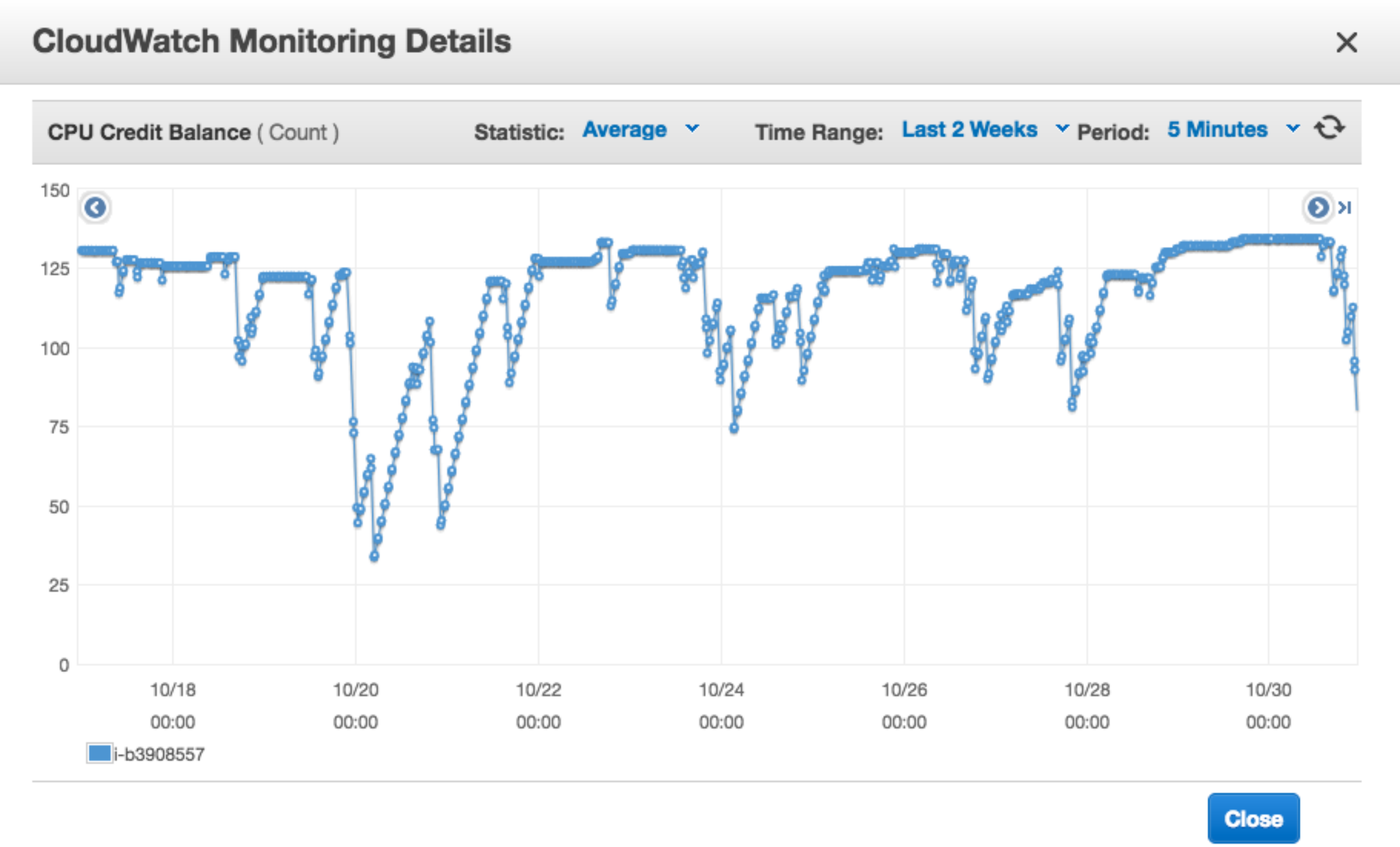
- Instances are classified in families, general purpose families are **t2** and **m4**. The **c4** family has more cores and the **r4** family has more RAM.

EC2 - the t2 family 1/2

- the t2 family is of particular interest, because it has fractional cores!
- the way this works is that you get 1-8 cores
  - at full speed for some time or
  - at a fraction of the speed for all the time or
  - if you choose so, also, at full speed for all the time (with extra costs)
- in a lot of use cases (I'd dare to say most) the first mode is very useful! unlike other vendors that overcommit cores, in EC2 you get a clear metric showing your allowed core usage ("CPU credits")

EC2 - the t2 family 2/2

- example: capybara is a machine in the PostgreSQL build farm; It is of type t2.micro (10% of a CPU); it runs new builds of PostgreSQL when they become available; since the duty cycle is less than 10%, the builds run at full CPU speed (so the PostgreSQL devs get their results in time), but I still only pay for a t2.micro instance



CloudWatch Monitoring Details

CPU Credit Balance (Count)

Statistic: Average

Time Range: Last 2 Weeks

Period: 5 Minutes

how much $$$?

- prices range from small to huge:

  $0.0063 / hour ~ $4.54 / month for the t2.nano instance
  ...
  $16.006 / hour ~ $11524.32 / month for the x1.32xlarge instance

- the trick is that you still pay the large by the second too. You probably wouldn't run in AWS if you needed such a machine all the time. On the other hand, you can get it for a week of business hours when you compute something huge for $800 (50 hours)

- besides on-demand pricing, you can get rebates (typically 30% if you commit for a year) and you can place requests on a spot market of unused capacity (this is fun :); on the spot market the beast goes for $ 2.74 in Ireland and $ 1.56 in Virginia while I'm writing this...

EBS - the block storage 1/2

- EC2 instances have RAM and CPUs and are networked through VPC. So what's missing? Disks!

- disks are provides by sub-service of **EC2**: **EBS** (**e**lastic **b**lock **d**evices). **EBS volumes** are network attached storage that provide between 1 GB and 16 TB of storage (freely settable); they can be attached and detached from running instances and appear dynamically under `/dev/xvd[fghij...];` the root volume (`/dev/xvda`) is the one that's created initially created from the AMI

EBS - the block storage 2/2

- there are different types of storage: **magnetic**, **gp2**, **io1**, **st1** and **sc1**; they differ in the underlying technology, access time, bandwidth (burstable or provisioned) and cost; I prefer:
  - **gp2** - SSD, bandwidth scales with size and is burstable, cost depends only on size, I/O is free
    @ $0.11 GB month in Ireland
  - **sc1** - magnetic, cost depends only on size, I/O is free, performance is slowish, minimum size is 0.5 TB
    @ $ 0.028 per GB month in Ireland
- EBS volumes have RAID-1 class durability and can be snapshotted; snapshots are incremental and durable (two AZs).

yes, but what about scaling?

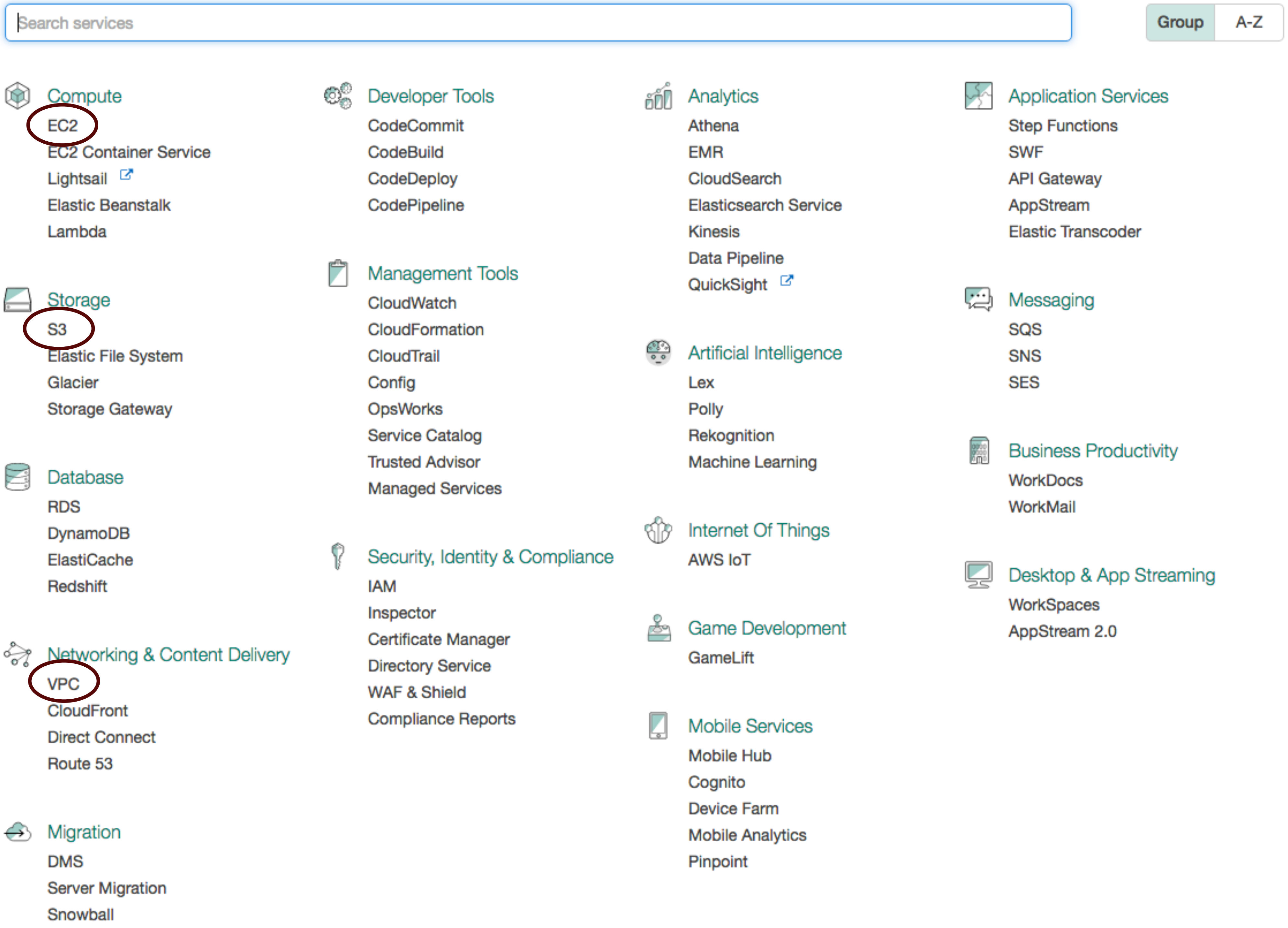
- isn't everything elastic here?!

- not quite: in **EC2** you run instances that are fixed sizes (RAM/CPUs); however, since all the instance data is on **EBS** volumes (or other persistent file system storage such as **EFS**), you can shut an instance down, change its type and boot it up again; this gives about 1 minute of downtime ("vertical scaling").

- EC2 does also **load balancing** (at the IP or application level), which puts more instances behind an IP; you can define a minimum and maximum number of instances and define criteria that would trigger automatic scaling of the number of instances ("horizontal scaling"); this would not get you and downtime (unless the number of instance goes to 0).

uhm, so we covered AWS now?

nope ;)



Search services

Group A-Z

Compute
- EC2
- EC2 Container Service
- Lightsail
- Elastic Beanstalk
- Lambda

Storage
- S3
- Elastic File System
- Glacier
- Storage Gateway

Database
- RDS
- DynamoDB
- ElastiCache
- Redshift

Networking & Content Delivery
- VPC
- CloudFront
- Direct Connect
- Route 53

Migration
- DMS
- Server Migration
- Snowball

Developer Tools
- CodeCommit
- CodeBuild
- CodeDeploy
- CodePipeline

Management Tools
- CloudWatch
- CloudFormation
- CloudTrail
- Config
- OpsWorks
- Service Catalog
- Trusted Advisor
- Managed Services

Security, Identity & Compliance
- IAM
- Inspector
- Certificate Manager
- Directory Service
- WAF & Shield
- Compliance Reports

Analytics
- Athena
- EMR
- CloudSearch
- Elasticsearch Service
- Kinesis
- Data Pipeline
- QuickSight

Artificial Intelligence
- Lex
- Polly
- Rekognition
- Machine Learning

Internet Of Things
- AWS IoT

Game Development
- GameLift

Mobile Services
- Mobile Hub
- Cognito
- Device Farm
- Mobile Analytics
- Pinpoint

Application Services
- Step Functions
- SWF
- API Gateway
- AppStream
- Elastic Transcoder

Messaging
- SQS
- SNS
- SES

Business Productivity
- WorkDocs
- WorkMail

Desktop & App Streaming
- WorkSpaces
- AppStream 2.0

we covered ~ 10% of S3 and EC2
and barely mentioned VPC

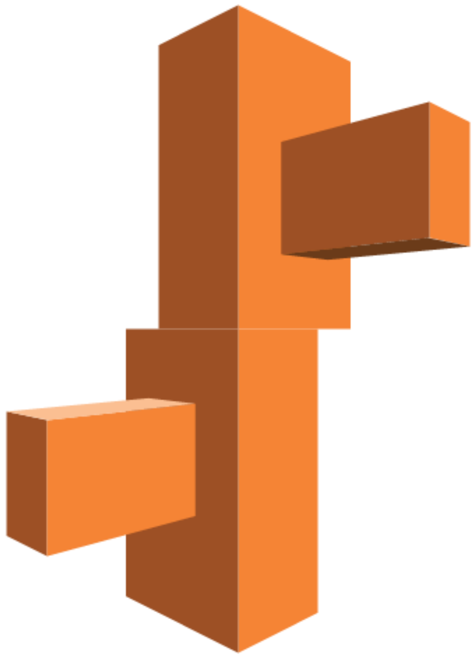
quick view at:
Elastic Beanstalk

- **Elastic Beanstalk** (EB) is a service for automatically deploying EC2 instances with your preferred application servers; among others, one with Java/Tomcat is available

- you can store a WAR-file in S3 and ask EB to launch one or more EC2 instances with Tomcat(s) serving your WAR-files

- Elastic Beanstalk has no extra cost with respect to the IAAS resources you're using, in fact you have full access to the underlying EC2 instances (root)

- there are two operation modes: "single instance" or "load balancing and autoscaling"



quick view at:
RDS

- **RDS** is a collection of relational database management servers as a service: PostgreSQL is available

- it is separated from EC2, one has no access to the underlying resources such as instances and disks

- the user just sees the endpoint running the PostgreSQL service on port 5432, that is fully managed by AWS

- RDS has a custom price structure

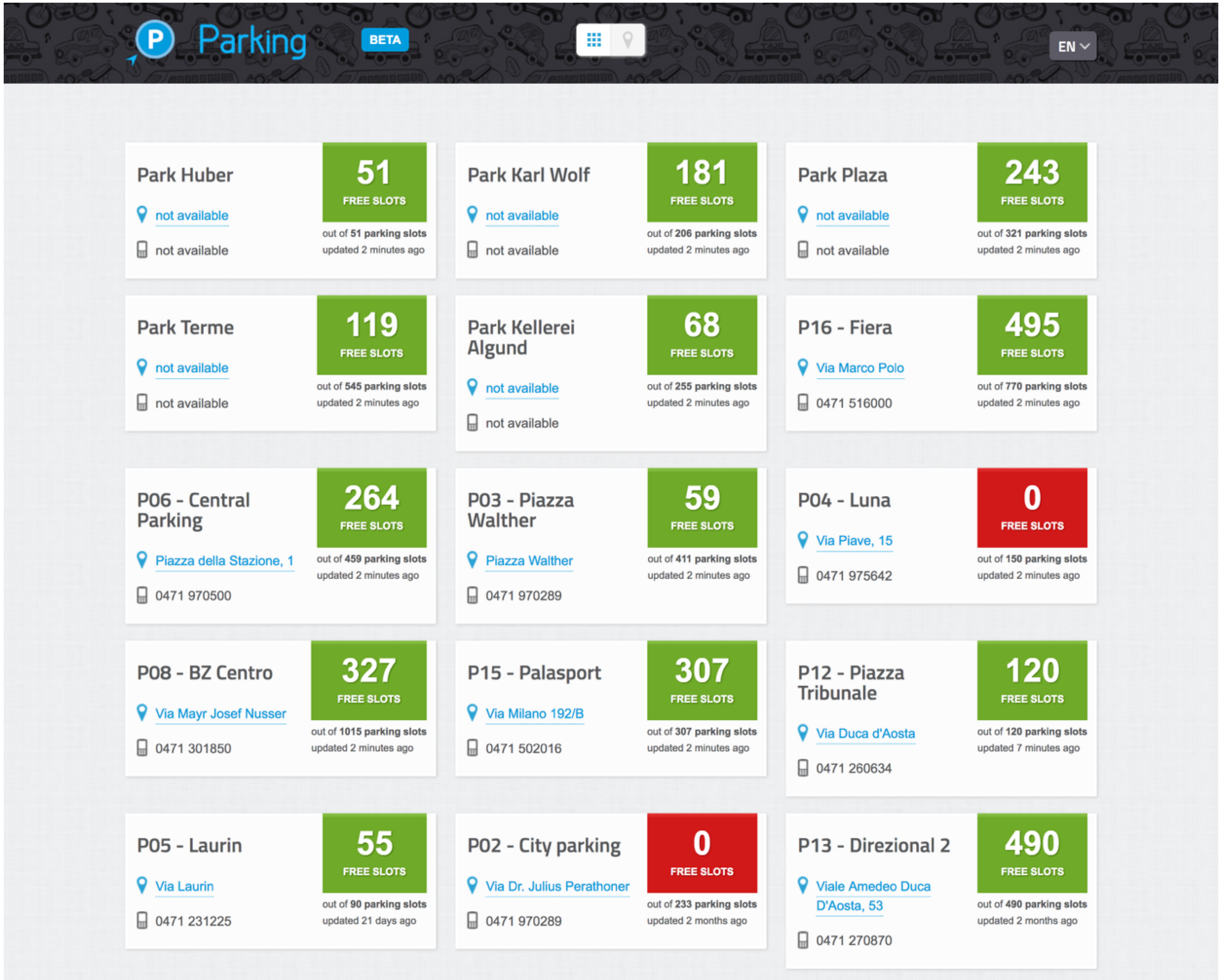
- as in EB, there are two configurations (single instance) and real high availability (two servers running in two AZ)

IDM's Big Data Platform (BDP)

the platform

- is a collection of data sources, data bases, analytics and front-ends that started in the realm of **smart mobility**, currently it is being expanded to cover also **smart tourism**
- various projects at IDM contributed (and contribute) to build the platform:
  - completed project Integreen (www.integreen-life.bz.it) and
  - current project BrennerLEC (brennerlec.life)
  - various other projects, e.g. real-time public transport data
- these projects aim to collect meteo/environment/traffic/parking data by polling sensors and other data sources, run forecasting and analytics and offer front-ends to the public such as:
  - www.parking.bz.it
  - traffic.bz.it
  - mobility.meran.eu and many more...

example: parking.bz.it



Park Huber
51 FREE SLOTS
not available
out of 51 parking slots
updated 2 minutes ago
not available

Park Karl Wolf
181 FREE SLOTS
not available
out of 206 parking slots
updated 2 minutes ago
not available

Park Plaza
243 FREE SLOTS
not available
out of 321 parking slots
updated 2 minutes ago
not available

Park Terme
119 FREE SLOTS
not available
out of 545 parking slots
updated 2 minutes ago
not available

Park Kellerei Algund
68 FREE SLOTS
not available
out of 255 parking slots
updated 2 minutes ago
not available

P16 - Fiera
495 FREE SLOTS
Via Marco Polo
out of 770 parking slots
updated 2 minutes ago
0471 516000

P06 - Central Parking
264 FREE SLOTS
Piazza della Stazione, 1
out of 459 parking slots
updated 2 minutes ago
0471 970500

P03 - Piazza Walther
59 FREE SLOTS
Piazza Walther
out of 411 parking slots
updated 2 minutes ago
0471 970289

P04 - Luna
0 FREE SLOTS
Via Piave, 15
out of 150 parking slots
updated 2 minutes ago
0471 975642

P08 - BZ Centro
327 FREE SLOTS
Via Mayr Josef Nusser
out of 1015 parking slots
updated 2 minutes ago
0471 301850

P15 - Palasport
307 FREE SLOTS
Via Milano 192/B
out of 307 parking slots
updated 2 minutes ago
0471 502016

P12 - Piazza Tribunale
120 FREE SLOTS
Via Duca d'Aosta
out of 120 parking slots
updated 7 minutes ago
0471 260634

P05 - Laurin
55 FREE SLOTS
Via Laurin
out of 90 parking slots
updated 21 days ago
0471 231225

P02 - City parking
0 FREE SLOTS
Via Dr. Julius Perathoner
out of 233 parking slots
updated 2 months ago
0471 970289

P13 - Direzional 2
490 FREE SLOTS
Viale Amedeo Duca D'Aosta, 53
out of 490 parking slots
updated 2 months ago
0471 270870

the technologies

- chosen core technologies are:
    - RDBMS: **PostgreSQL**
    - Application Server: **Tomcat**
    - Programming Language: **Java**
- currently, large part of the smart mobility side of the BDP have been written (or are being ported) to these technologies and are running in (or being migrated to) AWS - quick current stats:
    - 4 Tomcat Servers in Elastic Beanstalk
    - 2 PostgreSQL Servers in RDS with ~ 50 GB data (growing fast with new data and more services being imported)

BDP migration experience

why migrate

- IDM used to host the BDP on three VMs; there was some "gap" between the **administrators** (that looked after the OS) and the **developers** (that looked after their data and code)

    - example: at one point, called upon investigating a PostgreSQL slow-down, I discovered that the Linux OOM-killer had slain one important PostgreSQL process - PostgreSQL filled the log complaining, but nobody caught it

- there was a **flexibility/scaling** problem: there was no way to quickly adapt the resources to the actual need

    - example: there were not enough resources readily available to have a copy of some data and an extra Tomcat when the forecasting model was ported over to Java

expectations

- the new hosting platform should:
    - be **flexible/scalable** in a "cloud" way, i.e. it should be possible to quickly modify resources or deploy new resources and remove them as well
    - be as **admin-less** as possible, i.e. the developers shouldn't worry about OS, OS updates and security, network and firewall details, etc.
    - be as **open** as possible, i.e. developers should deploy vs standard PostgreSQL and standard Tomcat - not be tight into a proprietary platform as much as possible

what we used for **PostgreSQL**

- **RDS/PostgreSQL** is a **fully managed** PostgreSQL service deploying standard PostgreSQL (AWS has also custom variants, which we do not want to use) - fully managed means the user has no access to the server instance and has no super user rights
    - AWS takes care of **high availability** (cluster of two machines in separate availability zones)
    - AWS takes care of **OS security and updates** with no downtime (thanks to the high availability option)
    - AWS takes care of **backups** (snapshots and PITR)
    - AWS takes care of **monitoring** and **alerting**
    - **instance type** (CPU/RAM/bandwidth) can be **modified** with a reboot (~ 1 minute downtime), **disk space** can be **increased** with no down time

what we used for **Tomcat**

- **Elastic Beanstalk** is a **somewhat thin layer** above EC2, it's sort of PAAS-ish, but **not** fully managed

- AWS prepares (and keeps up to date) images with Amazon Linux and your preferred application service (in our case this is Tomcat 8); AWS keeps your application files (stored in S3)

- you still have root access to the instance (if you wish) and you can deploy custom scripts in the bundle executed when a new instance is created

- we currently use the **single instance model**:
    - AWS takes care of (re)launching an instance that is not available, but we don't have true high availability
    - AWS does update their image, but it's not deployed automatically
    - (of course these could be mitigated by going to the "load balancing and autoscaling" model...)

some workarounds were needed in EB...

1/4 - the "scale down" problem

- the BDP platform is split into quite some WAR-files - launching an instance for each of them would be somewhat wasteful
- what you can do is "**bundle**" the application into a ZIP file and deploy it as a whole
- this is somewhat mitigated by automatising everything using the EB command line tool
- still it is somewhat annoying that the developer needs to think about bundling WAR-files and sizing & matching instances

some workarounds were needed in EB...

2/4 - the RAM problem

- AWS sneakily sets **no swap space** on their Tomcat EB images

- BDP has some parts that need very few CPU cycles, so the smaller t2 instances are an ideal fit ... except concerning the amount of RAM used by Tomcat

- so we added a script to create swap space to the custom script, which avoids out of memory problems in the smaller t2 instances

some workarounds were needed in EB...

3/4 - the "where is my file system?" problem

- ideally there should be **no file system persistency** :)
- IDM custom apps are written to store everything in the database (the JDBC URL is stored in the environment)
- however, some third party app needed a persistent file system
- what we did is use a network file system (NFS) and mount it via custom scripts
- AWS offers NFS as a service (called EFS) with zero administration needs - so this comes in handy here

some workarounds were needed in EB...

4/4 - the "HTTPS" problem

- in the single instance mode of EB with different WAR-files serving different hostnames from a single IP there is no obvious way to use **HTTPS**

- the easiest workaround we found is using (abusing?) AWS's load balancers:
    - they can be used with a single endpoint, but support multiple hostnames via SNI
    - as a side effect, AWS handles the certificate for you for free (they're their own CA)

conclusions 1/2

- we are completely satisfied with **RDS** :)
- RDS provides lot of things (high availability - monitoring - alerting) that do not come out of the box with PostgreSQL
- one of the few things PostgreSQL is not very good in, is isolating users; as RDS scales linearly in price we found that running independent databases in separated instances resolves some load related problems we had
- still we're running 100% Open Source PostgreSQL, so there is no vendor lock in

conclusions 2/2

- we are reasonably satisfied with **EB**, but we needed some custom scripting and a few workarounds (some mild vendor lock-in looms somewhere there)

- one might argue that 3 of the 4 workarounds where only needed because we insisted on saving money and run a lot of apps on a few single instance Tomcats ;)

- however, at this point, AWS could improve EB a lot by having a better way to automatically map WAR-files to instances and improve handling of HTTPS thanks :)