Your first week with EC2

(and other AWS things…)

AWS Summit 2013

Navigating the Cloud

Ryan Shuttleworth, Technical Evangelist

What are we going to cover?

Your first 5 days with EC2...

things you should know/think about up front ◀

some best practices for getting started ◀

essential technologies to dive into and get familiar with ◀

architectural principles you should immerse yourself in ◀

What are we going to cover?

Your first 5 days with EC2…

hear a ‘looking back at our first year’ customer story ◀

compressed into 5 days ◀

DAY 1

organise your house

Users & Roles

Start as you mean to go on

Secure your console with IAM roles

A little time spent now will save headaches later

Users & Roles

Start as you mean to go on

Secure your console with IAM roles

A little time spent now will save headaches later

Accounts & Billing

Create a structure that makes sense

Dev & Test accounts vs production

Consolidated billing

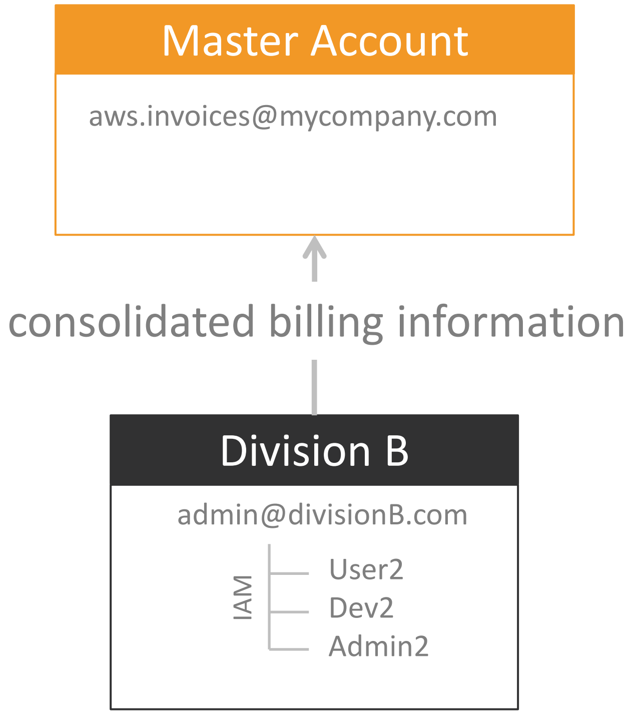
Resource tagging

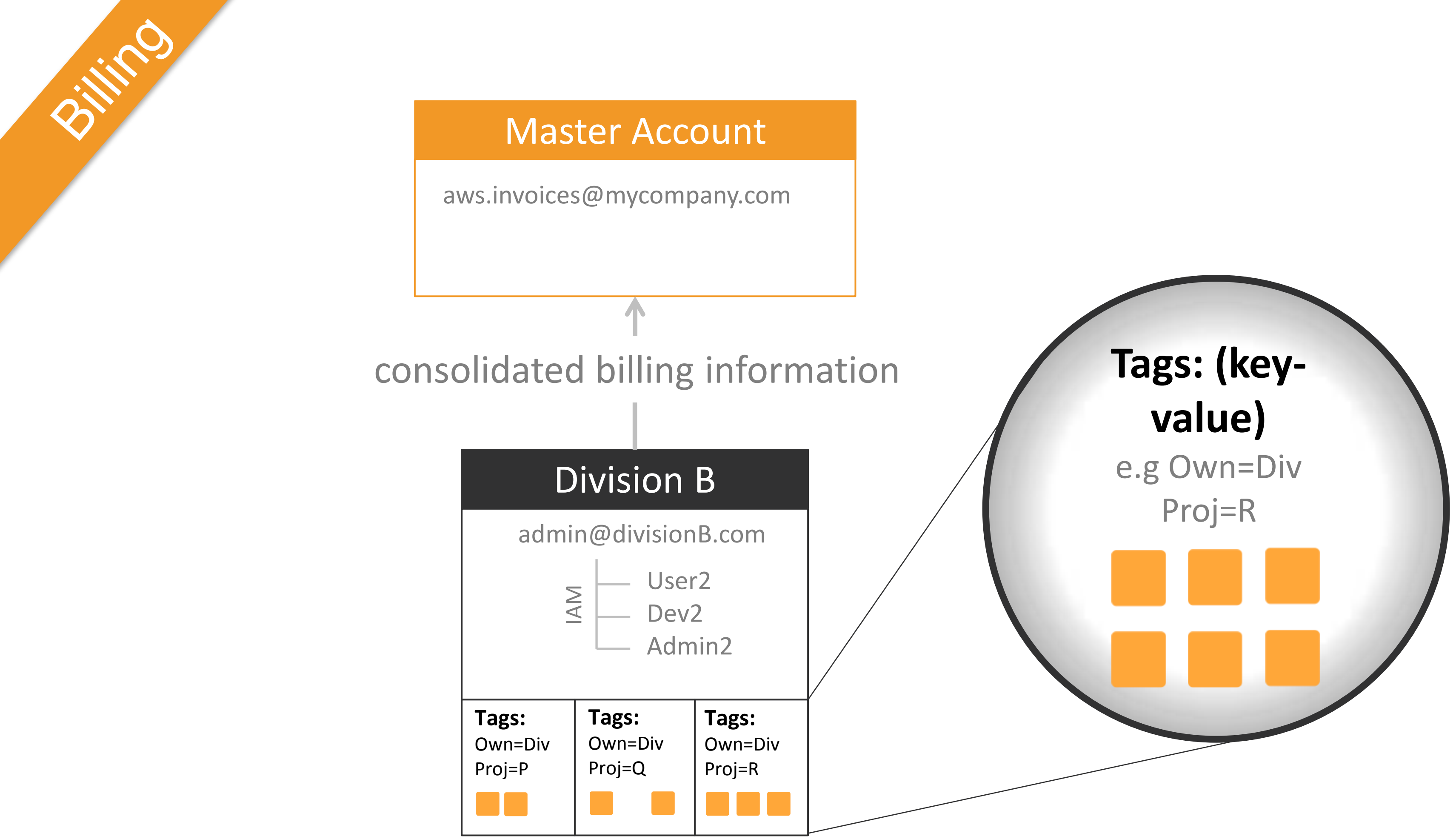
Billing

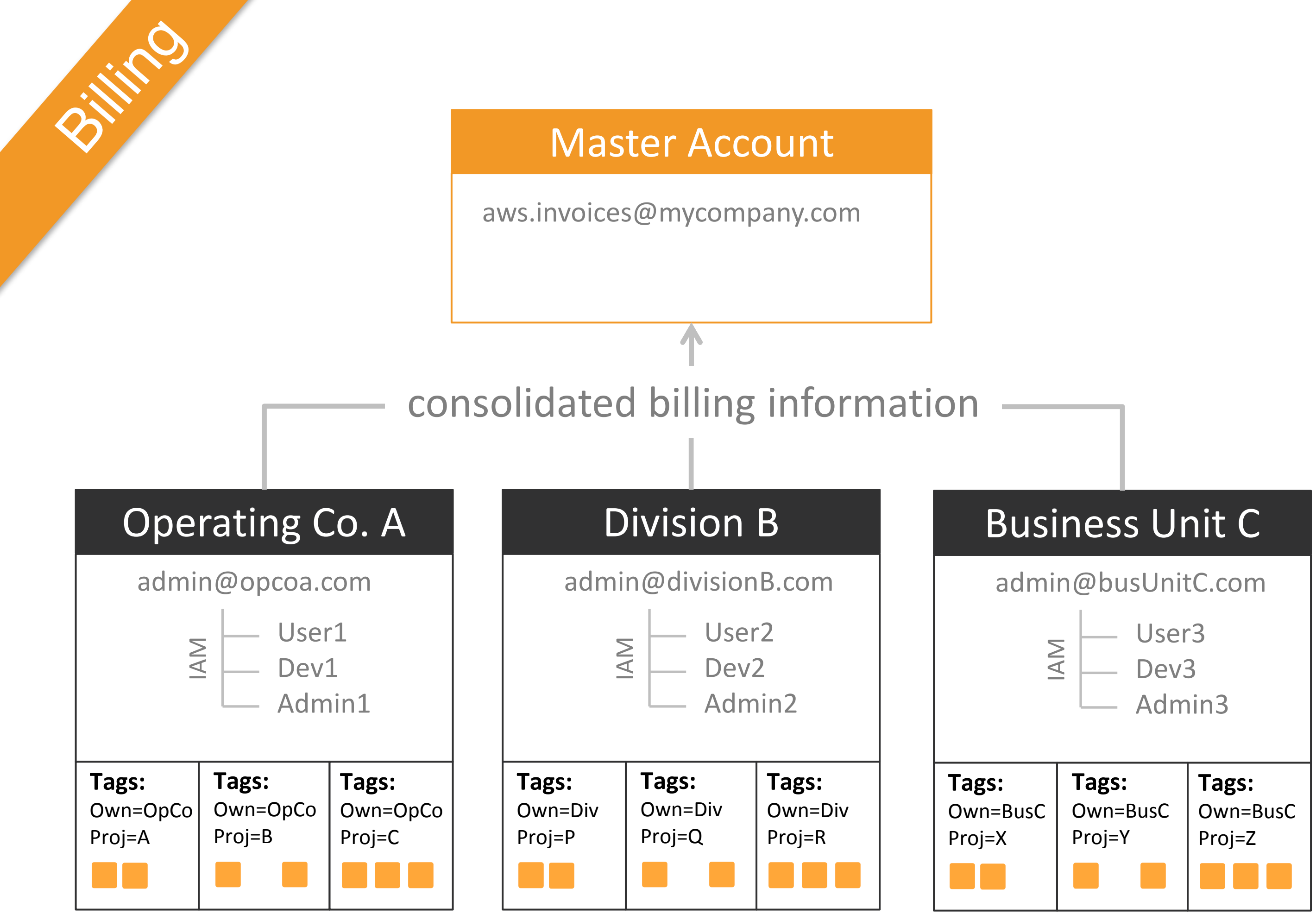
Master Account

aws.invoices@mycompany.com

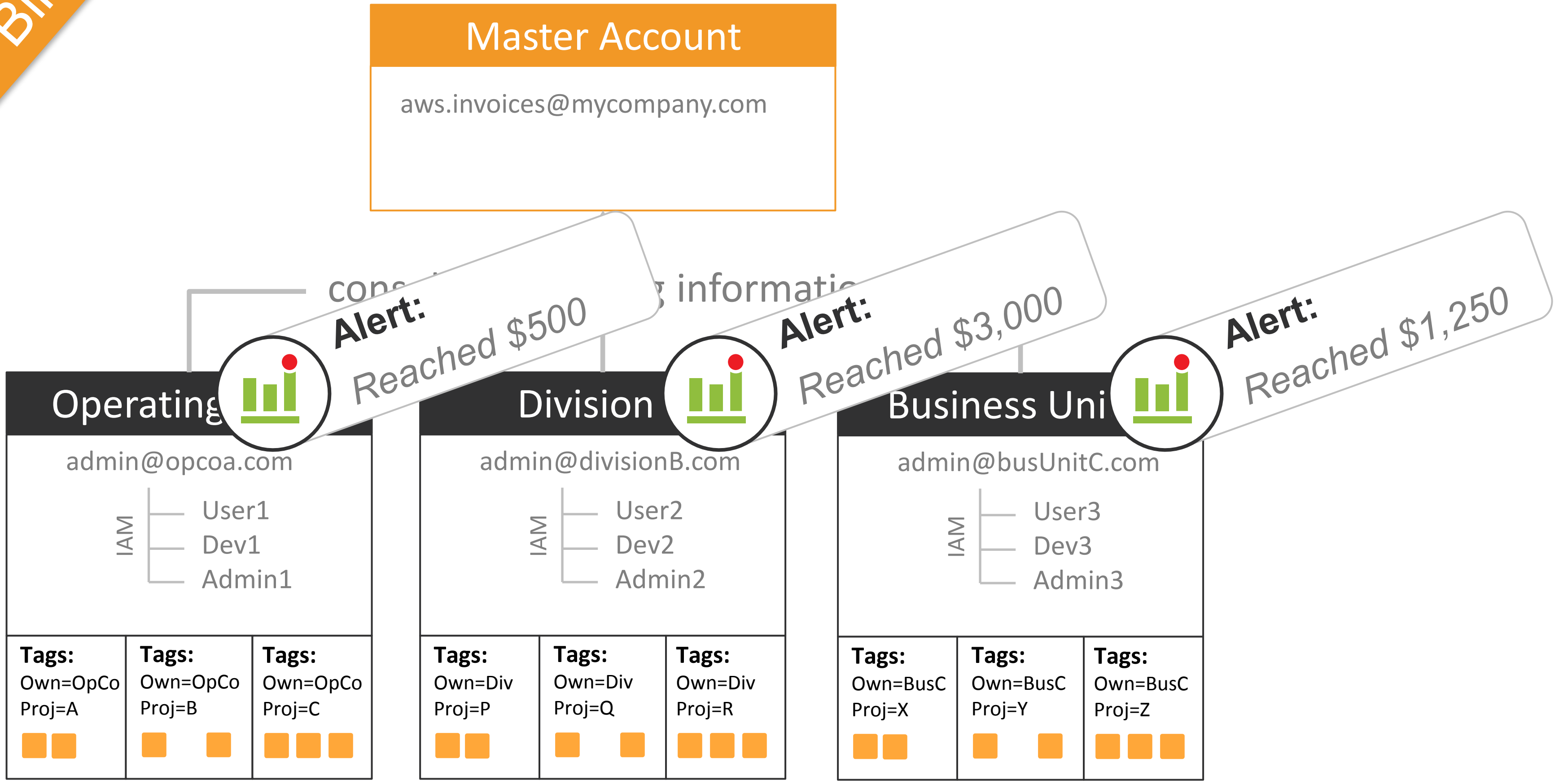
Billing


Master Account
aws.invoices@mycompany.com
consolidated billing information
Division B
admin@divisionB.com
IAM
User2
Dev2
Admin2

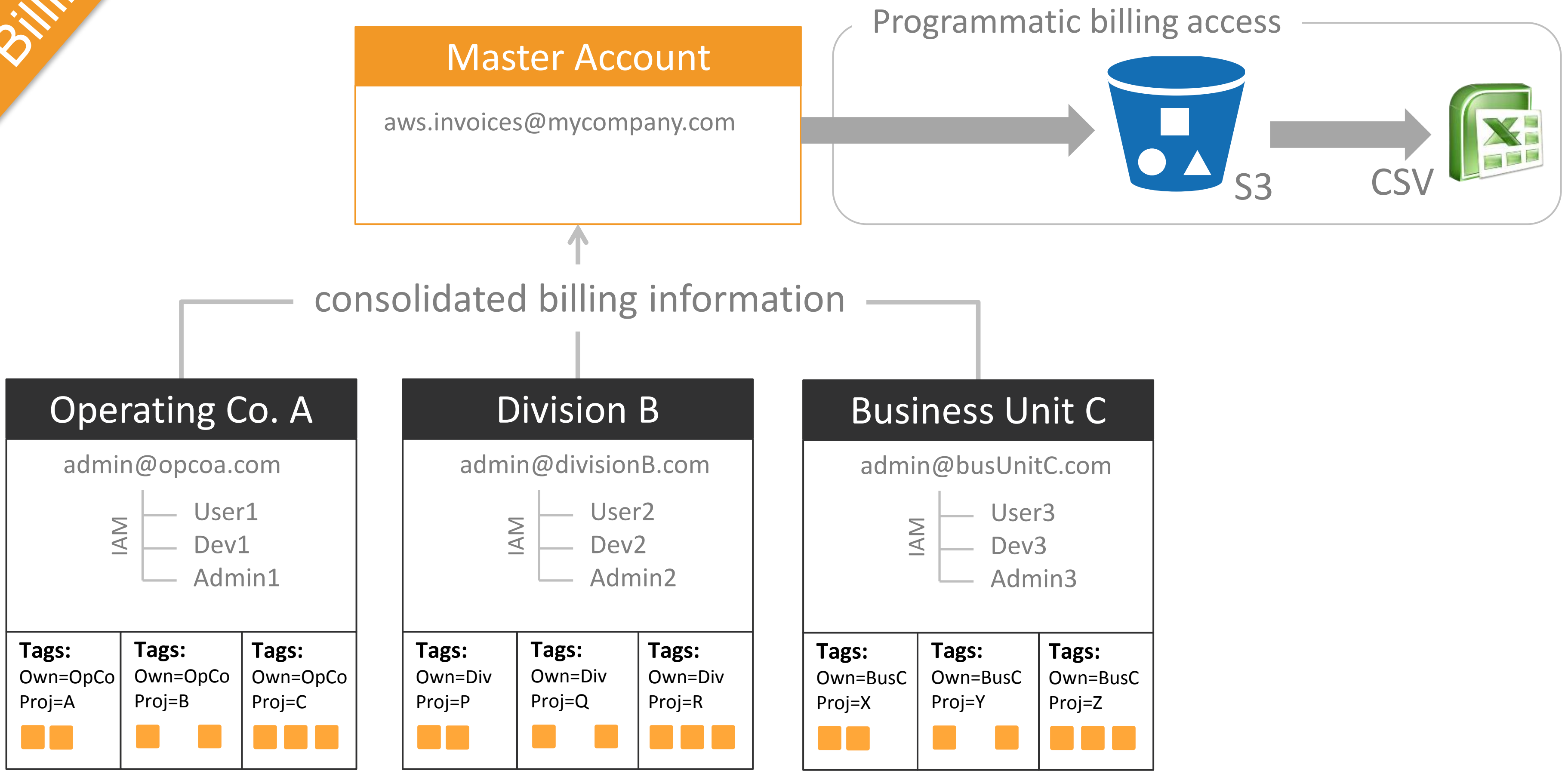

Billing
Master Account
aws.invoices@mycompany.com
consolidated billing information
Division B
admin@divisionB.com
IAM
User2
Dev2
Admin2
Tags: Own=Div Proj=P
Tags: Own=Div Proj=Q
Tags: Own=Div Proj=R
Tags: (key-value)
e.g Own=Div Proj=R


Billing
Master Account
aws.invoices@mycompany.com
consolidated billing information
Operating Co. A
admin@opcoa.com
IAM
User1
Dev1
Admin1
Tags: Own=OpCo Proj=A
Tags: Own=OpCo Proj=B
Tags: Own=OpCo Proj=C
Division B
admin@divisionB.com
IAM
User2
Dev2
Admin2
Tags: Own=Div Proj=P
Tags: Own=Div Proj=Q
Tags: Own=Div Proj=R
Business Unit C
admin@busUnitC.com
IAM
User3
Dev3
Admin3
Tags: Own=BusC Proj=X
Tags: Own=BusC Proj=Y
Tags: Own=BusC Proj=Z

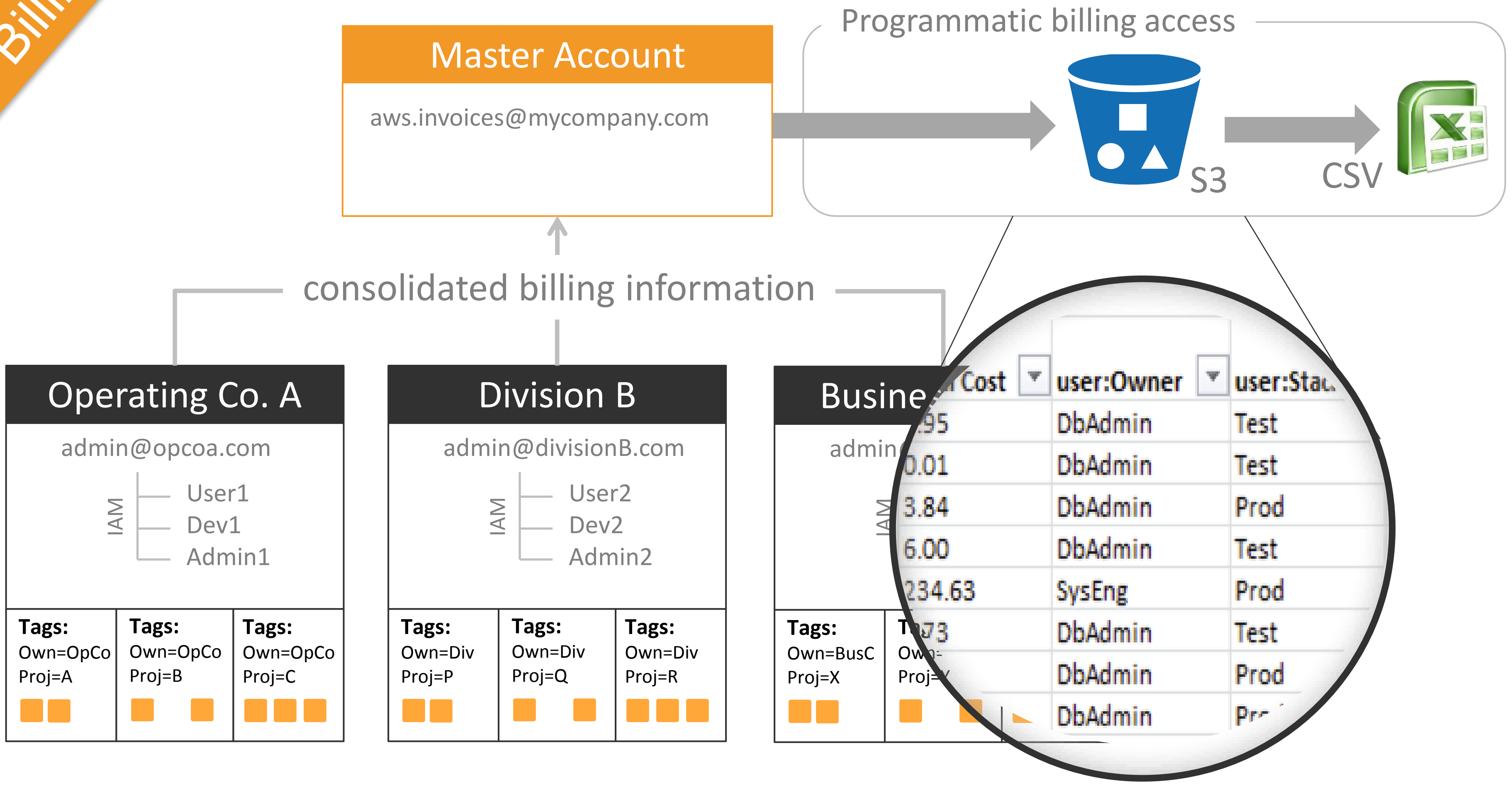
Billing


Master Account
aws.invoices@mycompany.com
Alert:
Reached $500
Alert:
Reached $3,000
Alert:
Reached $1,250
Operating
admin@opcoa.com
IAM
User1
Dev1
Admin1
Tags: Own=OpCo Proj=A
Tags: Own=OpCo Proj=B
Tags: Own=OpCo Proj=C
Division
admin@divisionB.com
IAM
User2
Dev2
Admin2
Tags: Own=Div Proj=P
Tags: Own=Div Proj=Q
Tags: Own=Div Proj=R
Business Uni
admin@busUnitC.com
IAM
User3
Dev3
Admin3
Tags: Own=BusC Proj=X
Tags: Own=BusC Proj=Y
Tags: Own=BusC Proj=Z

Billing


Programmatic billing access
Master Account
aws.invoices@mycompany.com
S3
CSV
consolidated billing information
Operating Co. A
admin@opcoa.com
IAM
User1
Dev1
Admin1
Tags:
Own=OpCo
Proj=A
Tags:
Own=OpCo
Proj=B
Tags:
Own=OpCo
Proj=C
Division B
admin@divisionB.com
IAM
User2
Dev2
Admin2
Tags:
Own=Div
Proj=P
Tags:
Own=Div
Proj=Q
Tags:
Own=Div
Proj=R
Business Unit C
admin@busUnitC.com
IAM
User3
Dev3
Admin3
Tags:
Own=BusC
Proj=X
Tags:
Own=BusC
Proj=Y
Tags:
Own=BusC
Proj=Z

Billing



Cost	user:Owner	user:Stac
.95	DbAdmin	Test
0.01	DbAdmin	Test
3.84	DbAdmin	Prod
6.00	DbAdmin	Test
234.63	SysEng	Prod
73	DbAdmin	Test
	DbAdmin	Prod
	DbAdmin	Pr

Secrets & Keys

Your front door keys

↑

Secrets & Keys

Your front door keys

Control access to your instances

Secrets & Keys

Key management strategy

Secrets & Keys

Your front door keys

Control access to your APIs

Control access to your instances

Use IAM Roles to distribute to instances

Key management strategy

DAY 2

learn the basics

Complete control

Elastic capacity

Flexible

What is EC2?

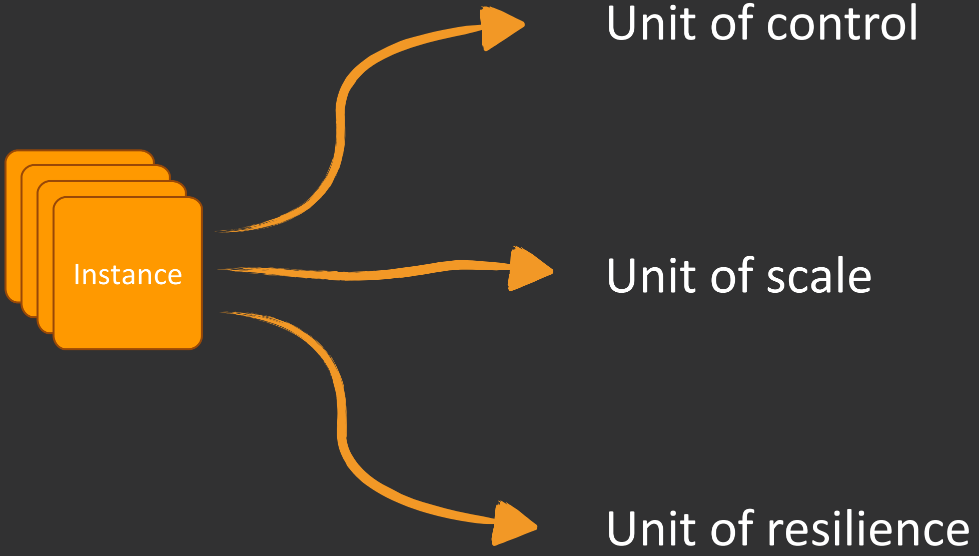
Reliable

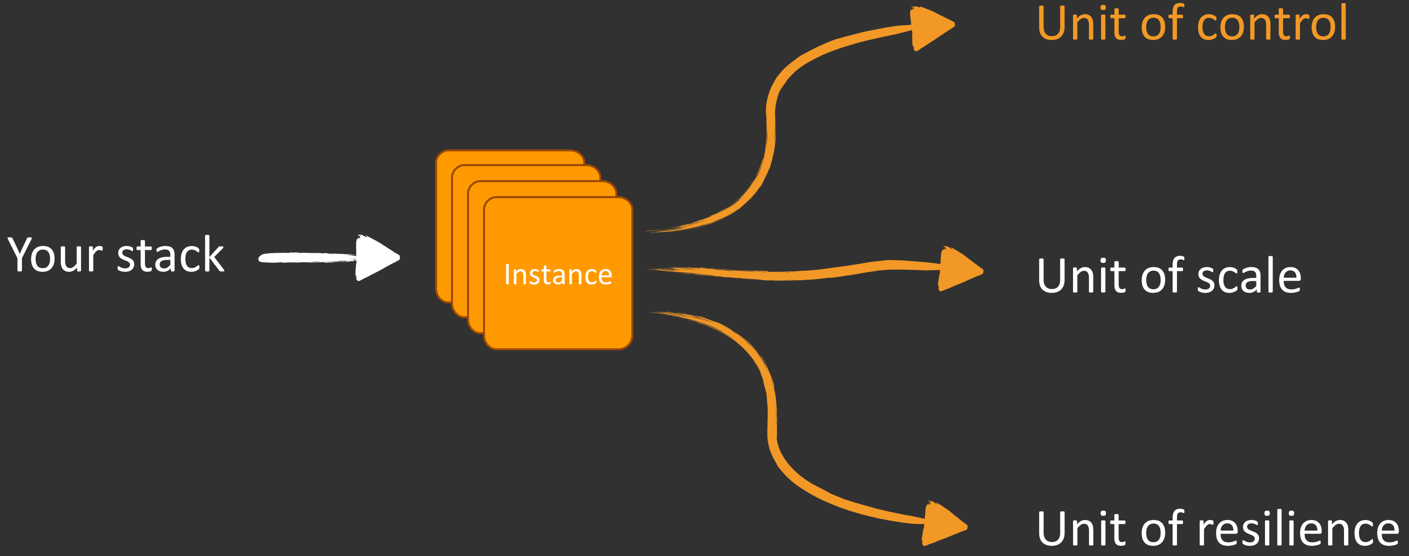
Secure

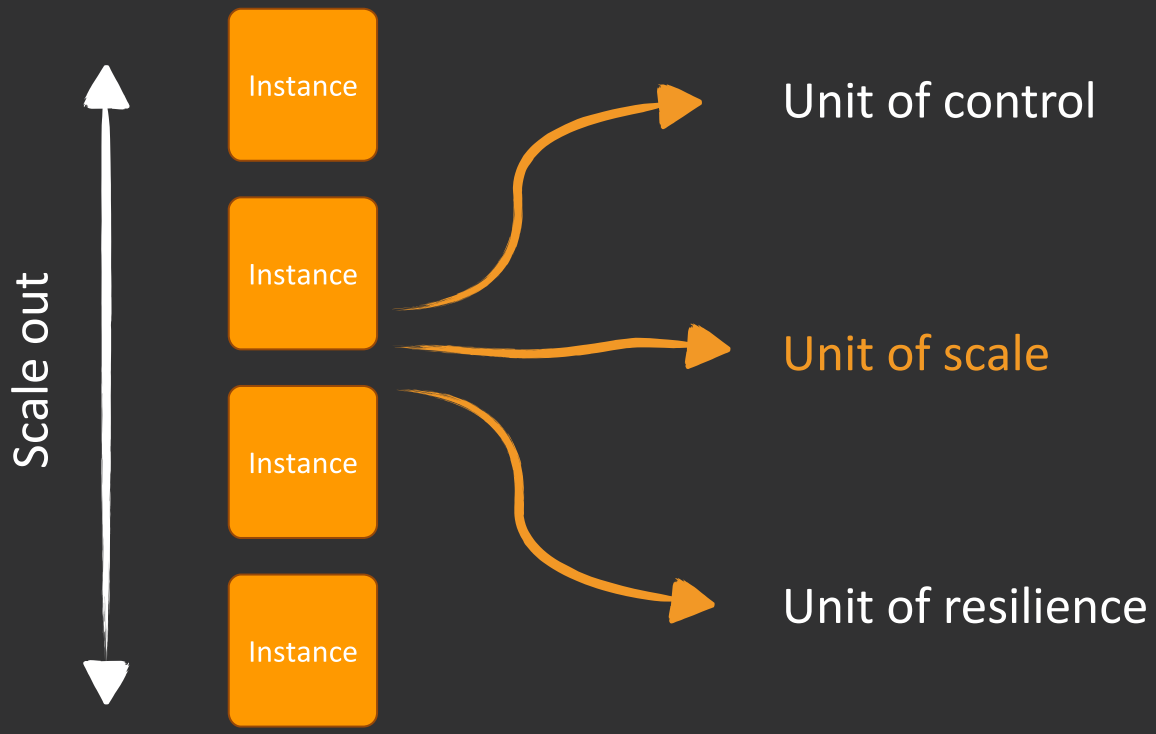
Inexpensive

Disposable compute

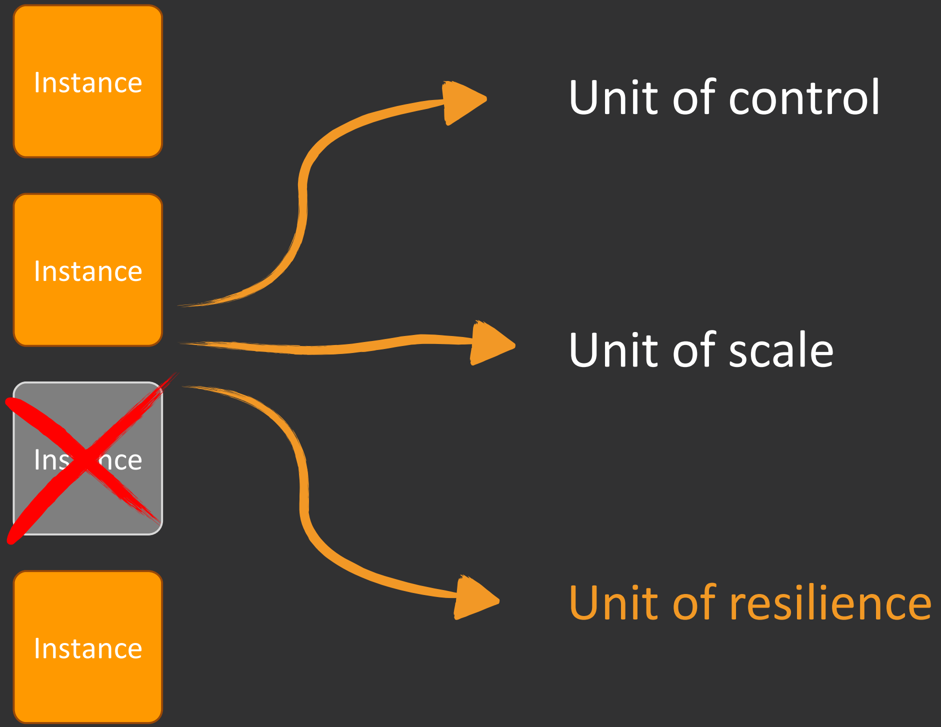
Instance

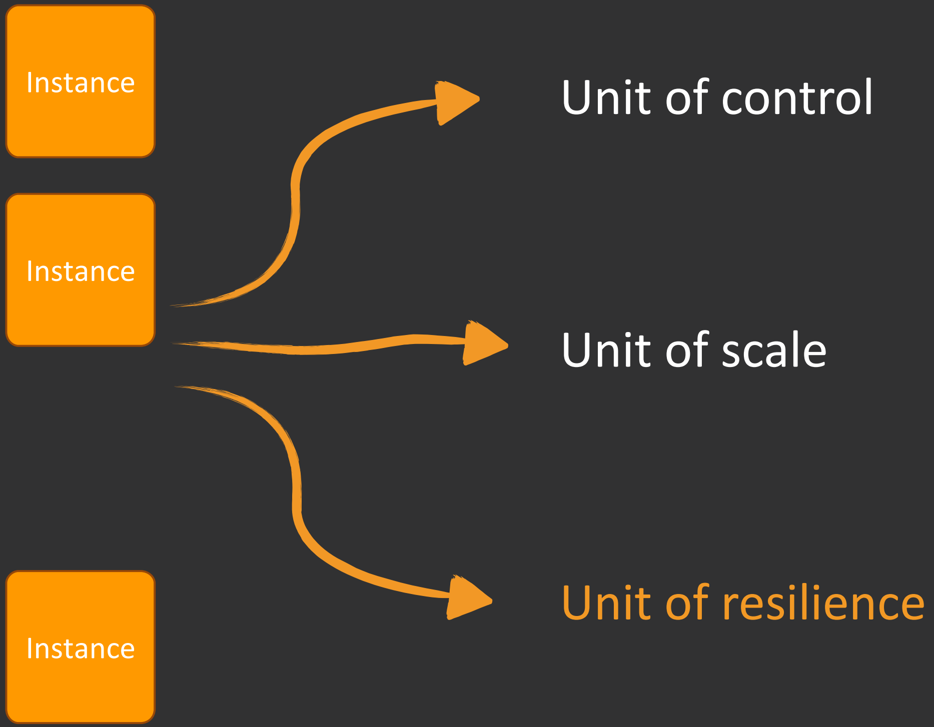

Instance
Unit of control
Unit of scale
Unit of resilience

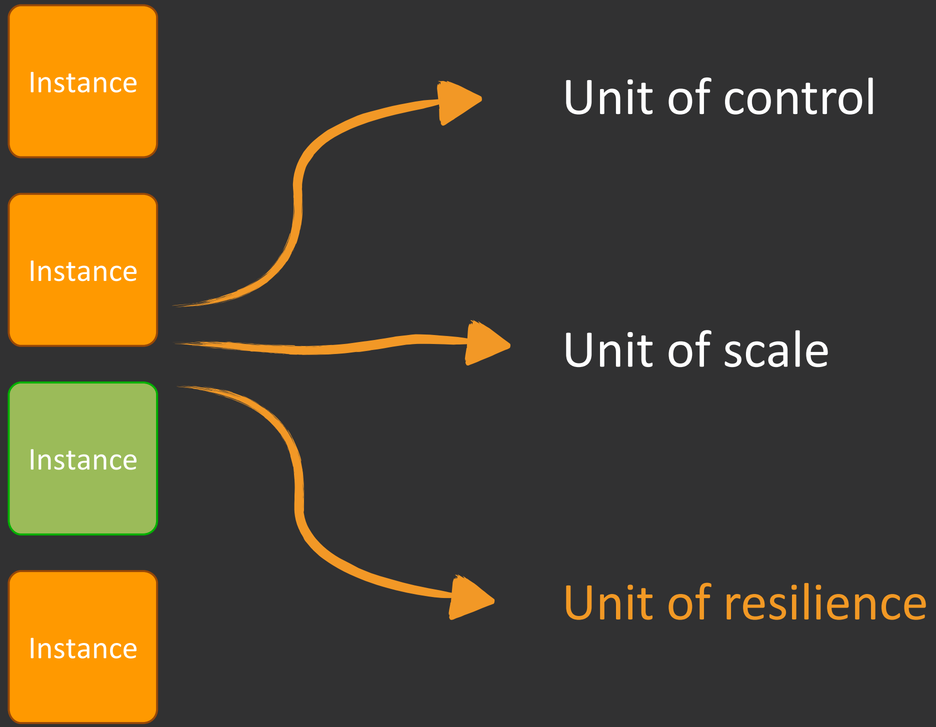

Your stack
Instance
Unit of control
Unit of scale
Unit of resilience



amazon web services


Instance
Instance
Instance
Instance
Unit of control
Unit of scale
Unit of resilience


Instance
Instance
Instance
Unit of control
Unit of scale
Unit of resilience


Instance
Instance
Instance
Instance
Unit of control
Unit of scale
Unit of resilience

Think differently

Compute is transient

Programmatic resources $\longrightarrow$ Treat your datacentre resources like code

Programmatic resources → Treat your datacentre resources like code

Distributed systems → Design for decoupled systems up front

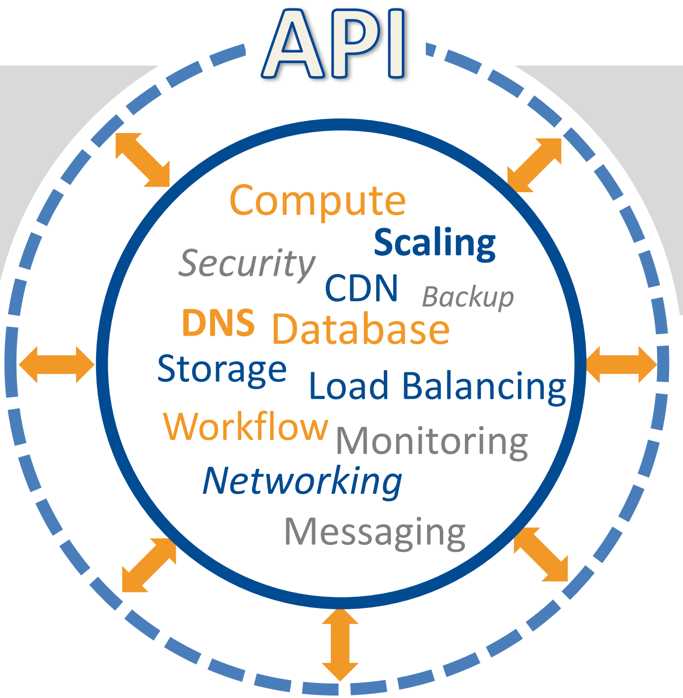
Programmatic resources	→	Treat your datacentre resources like code
Distributed systems	→	Design for decoupled systems up front
Late binding	→	Decide what it will run on at point of deployment

DAY 3

go wild with tools

Everything is programmable

Access everything via CLI, API or Console



Achieve the highest levels of automation sophistication with ease

```
$>  ec2-run-instances ami-54cf5c3d
    --instance-count 2
    --group webservers
    --key mykey
    --instance-type m1.small
```

CLI Tools

```
>>> import boto.ec2
>>> conn = boto.ec2.connect_to_region("us-east-1")
>>> conn.run_instances(
        'ami-54cf5c3d',
        key_name='mykey',
        instance_type='m1.small',
        security_groups=['webservers'])
```

Python boto

amazon web services

Resources created programmatically

Resources created programmatically

Configure automatically

Bootstrapping

Bake an AMI

Start an instance

Configure the instance

Create an AMI from your instance

Start new ones from the AMI

Bootstrapping

Bake an AMI

Start an instance

Configure the instance

Create an AMI from your instance

Start new ones from the AMI

```
$>   ec2-run-instances
       <your ami-id>
```

Bootstrapping

Bake an AMI	vs	Configure dynamically
Start an instance		Launch an instance
Configure the instance		Use metadata service and cloud-init to perform actions on instance when it launches
Create an AMI from your instance		
Start new ones from the AMI		

Bootstrapping

Bake an AMI

Build your base images and setup custom initialisation scripts

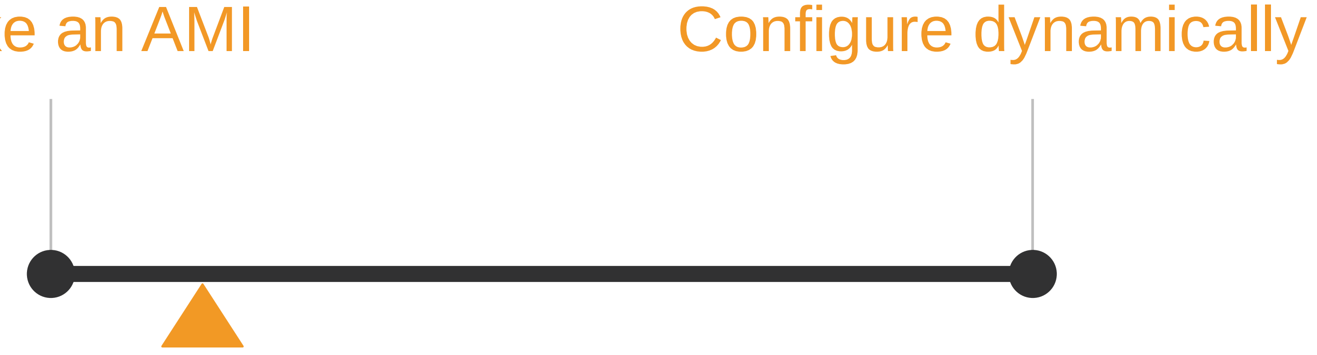
Maintain your 'golden' base

+

Configure dynamically

Use bootstrapping to pass custom information in and perform post launch tasks like pulling code from SVN

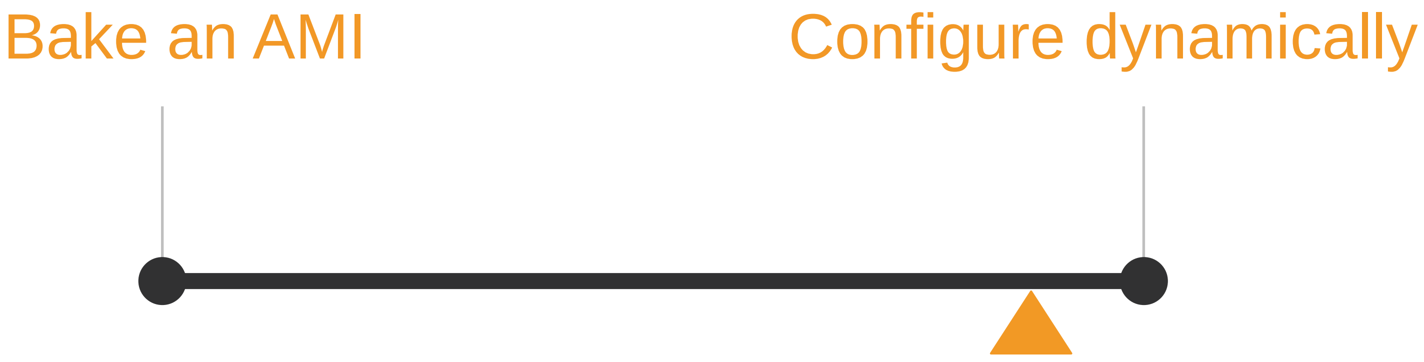
Bootstrapping



Time consuming configuration
(e.g startup time)

Static configurations
(e.g less change management)

Bootstrapping

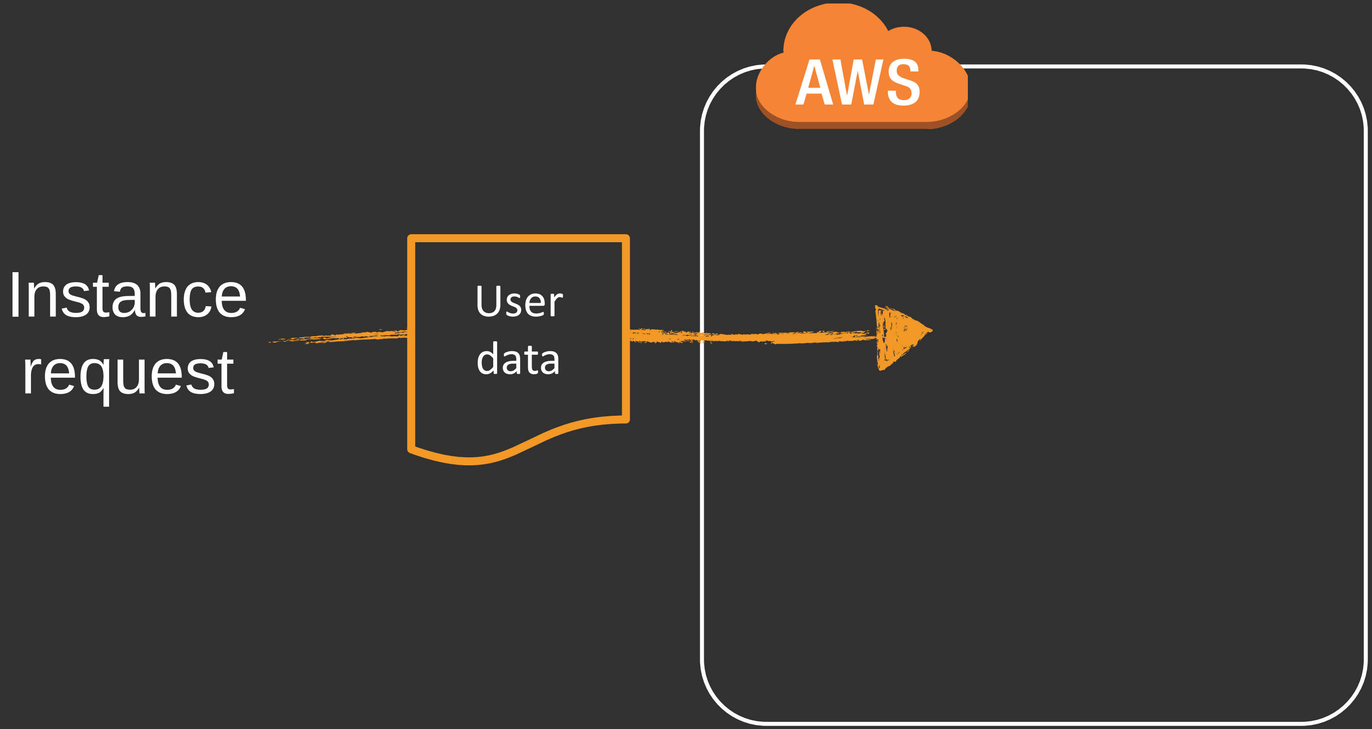


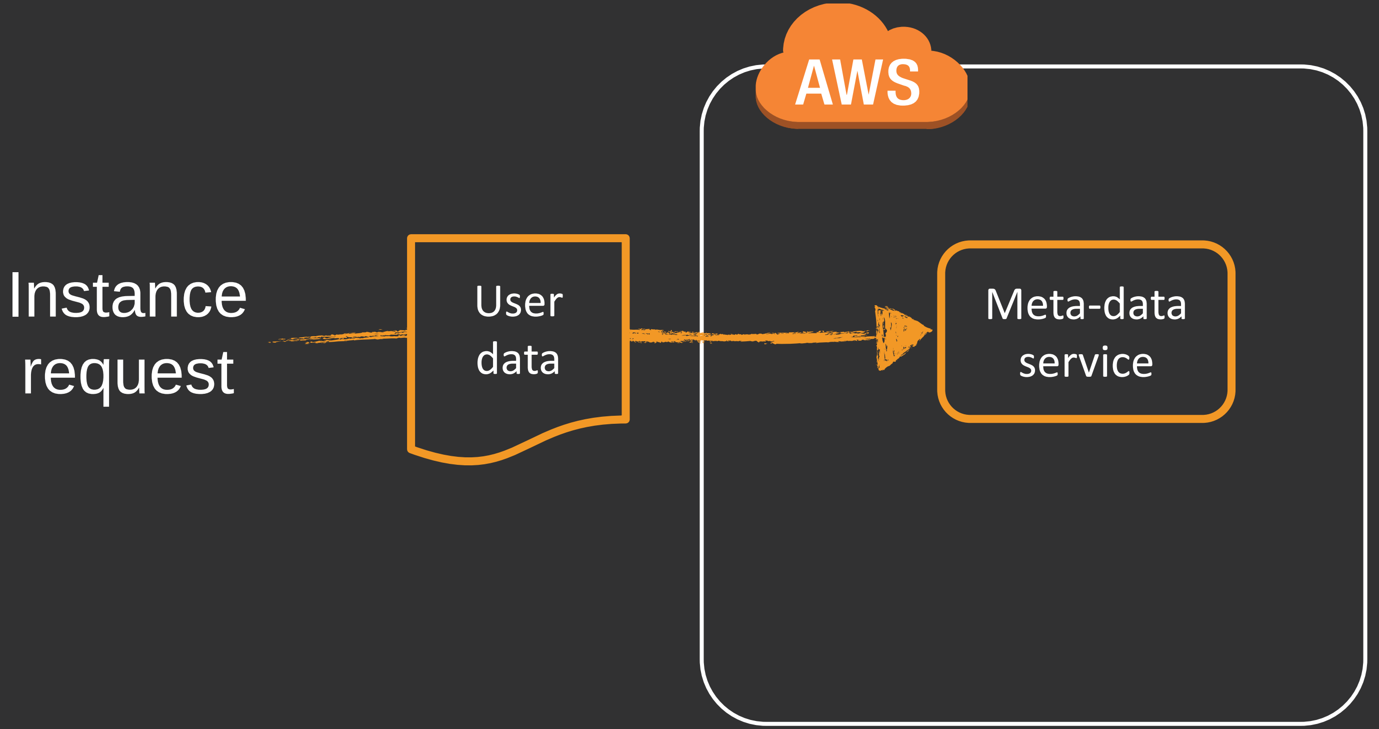
Continuous deployment
(e.g latest code)

Environment specific
(e.g dev-test-prod)

Goal is bring an instance up in a useful state

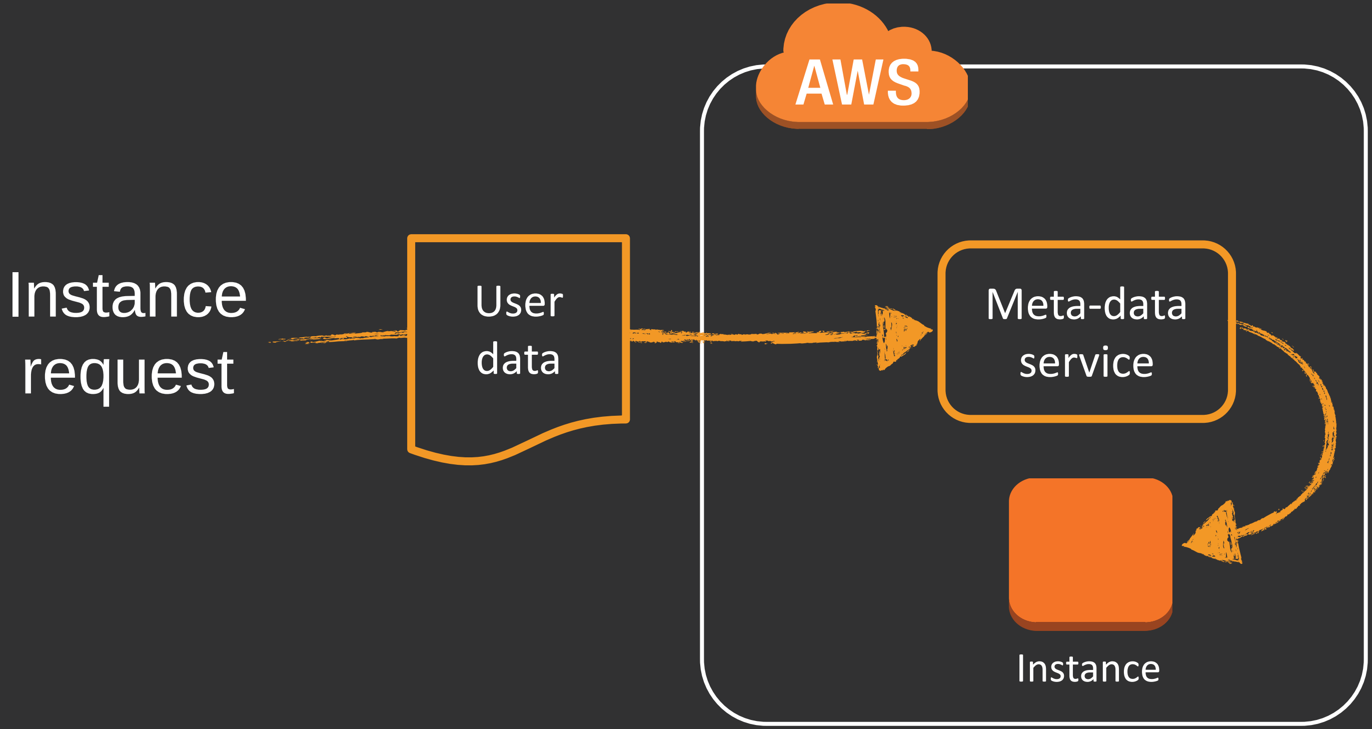
The balance will vary depending upon your application


Instance request
User data
AWS



User-data

User-data



Tip

Shell script in user-data will be executed on launch:

```
#!/bin/sh
yum -y install httpd php mysql php-mysql
chkconfig httpd on
/etc/init.d/httpd start
```

Tip

Amazon Windows EC2Config Service executes user-data on launch:

```
<script>dir > c:\test.log</script>

<powershell>any command that you can run</powershell>
```

AWS Powershell Tools

```
<powershell>
    Read-S3Object -BucketName myS3Bucket
    -Key myFolder/myFile.zip
    -File c:\destinationFile.zip
</powershell>
```

amazon web services

Why do this?

Automation
Less fingers, less mistakes

Security
Instances locked down by default

Availability
Drive higher availability with self-healing

Flexible
Shell, Powershell, CloudFormation ,Chef, Puppet, OpsWorks

Scale
Manage large scale deployments and drive autoscaling

Efficiency
Audit and manage your estate with less time & effort

DAY 4

try something new

Unconstrained

EC2 resources

Unconstrained

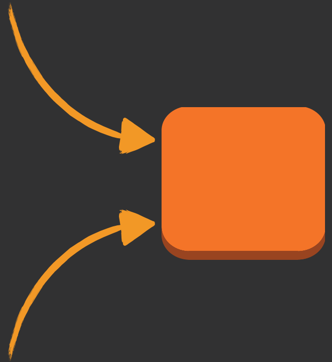
Complimentary services

My little instance

(created programmatically)

A bit of S3 code
(pulled down automatically from S3)

```
>>> from boto.s3.key import Key
>>> k = Key(bucket)
>>> k.key = 'foobar'
>>> k.set_contents_from_string('This is a test of S3')
```

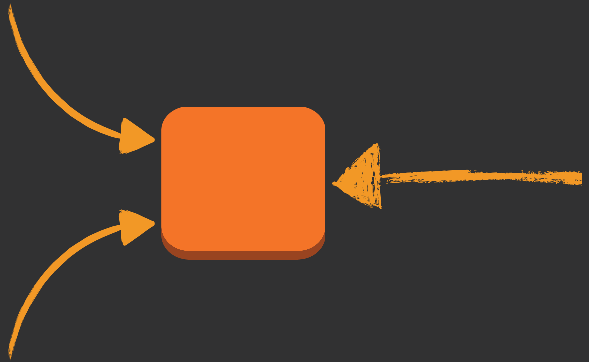


My little instance
(created programmatically)

A bit of S3 code

(installed automatically)

```
>>> from boto.s3.key import Key
>>> k = Key(bucket)
>>> k.key = 'foobar'
>>> k.set_contents_from_string('This is a test of S3')
```



UNLIMITED storage from my little instance

My little instance

(created programmatically)

Services instead of software

Removes undifferentiated heavy lifting

Services instead of software

Removes undifferentiated heavy lifting

S3 for object storage

SQS for queues

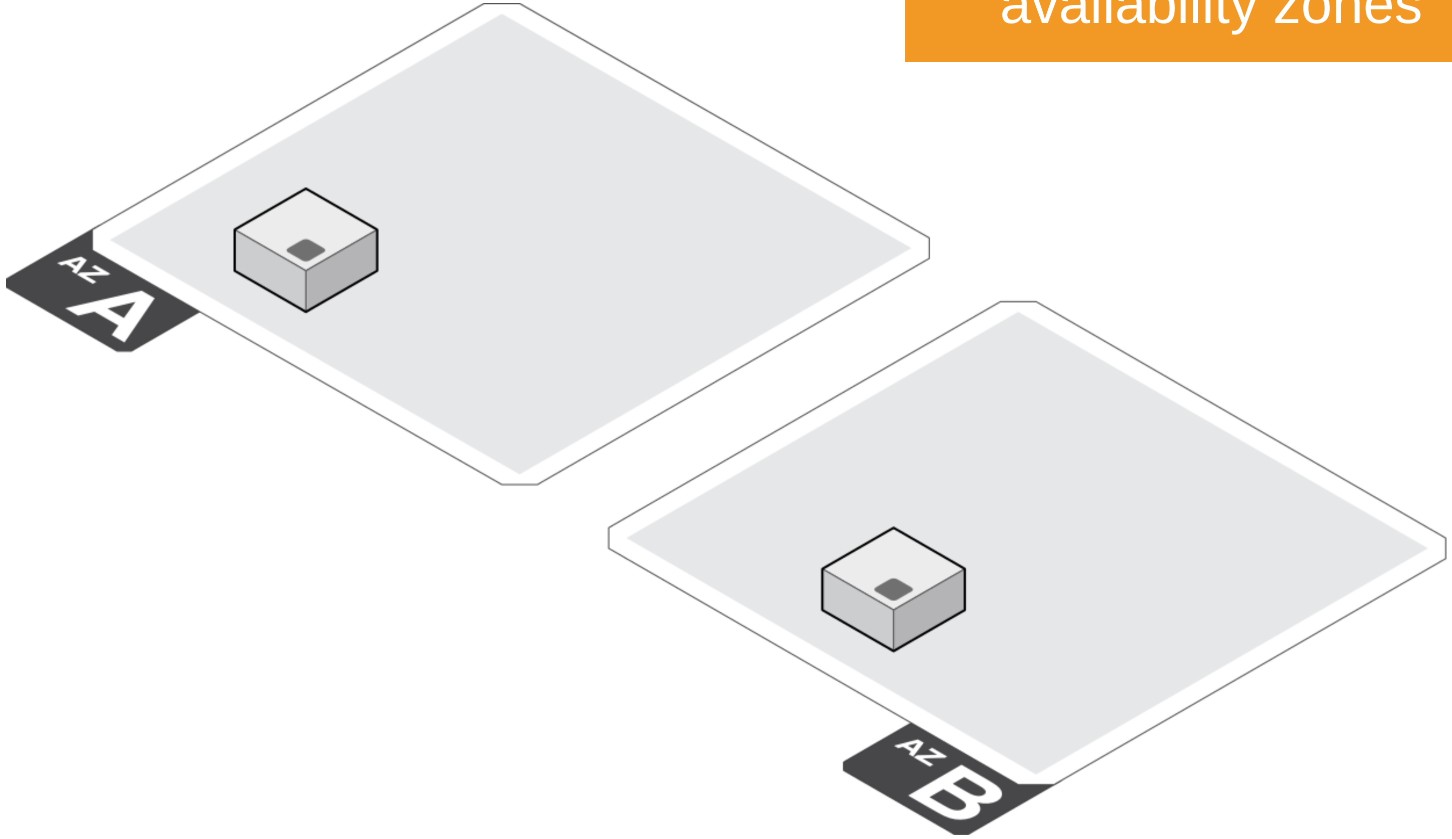
RDS for databases

CloudWatch for monitoring

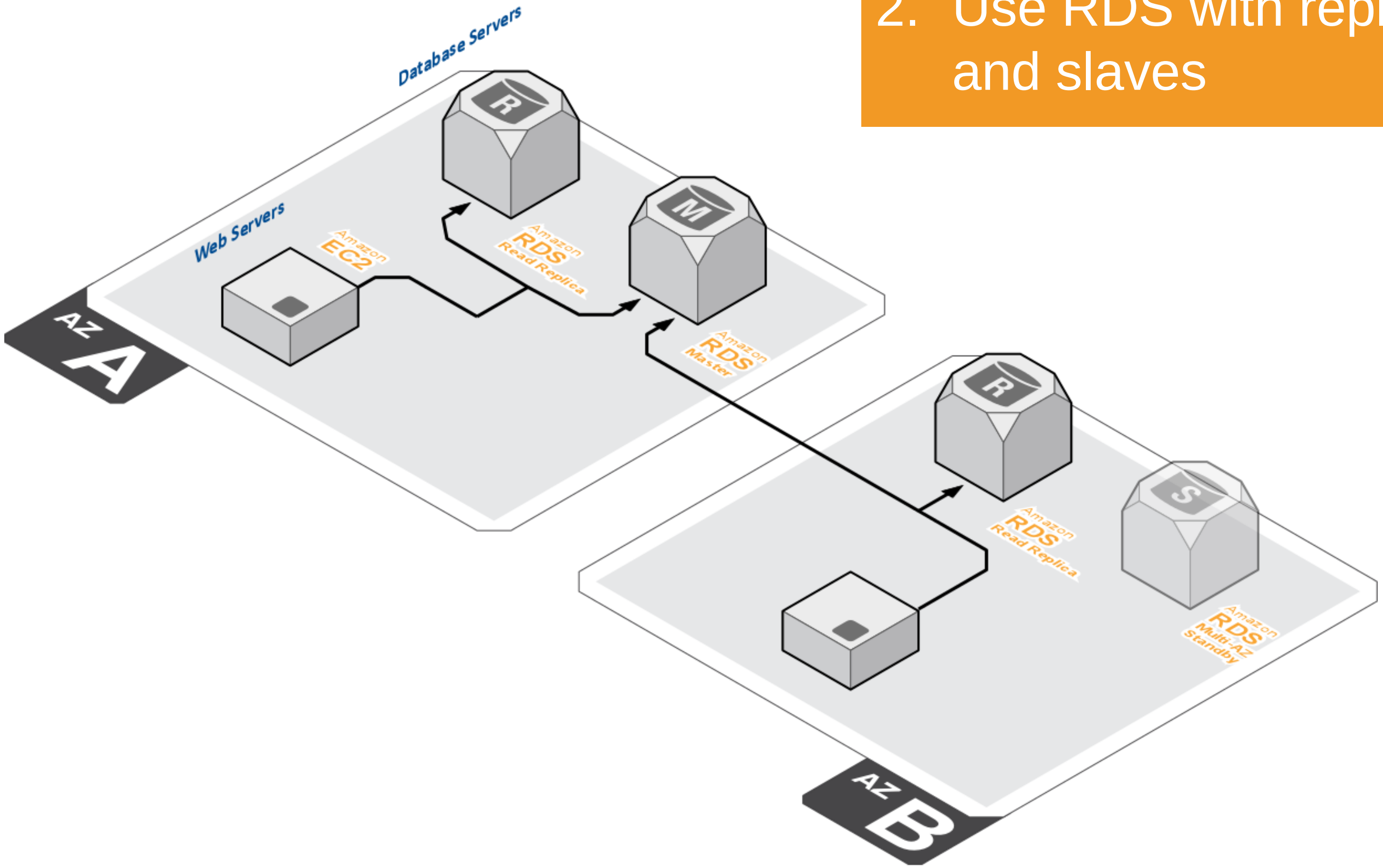
DAY 5

put something together

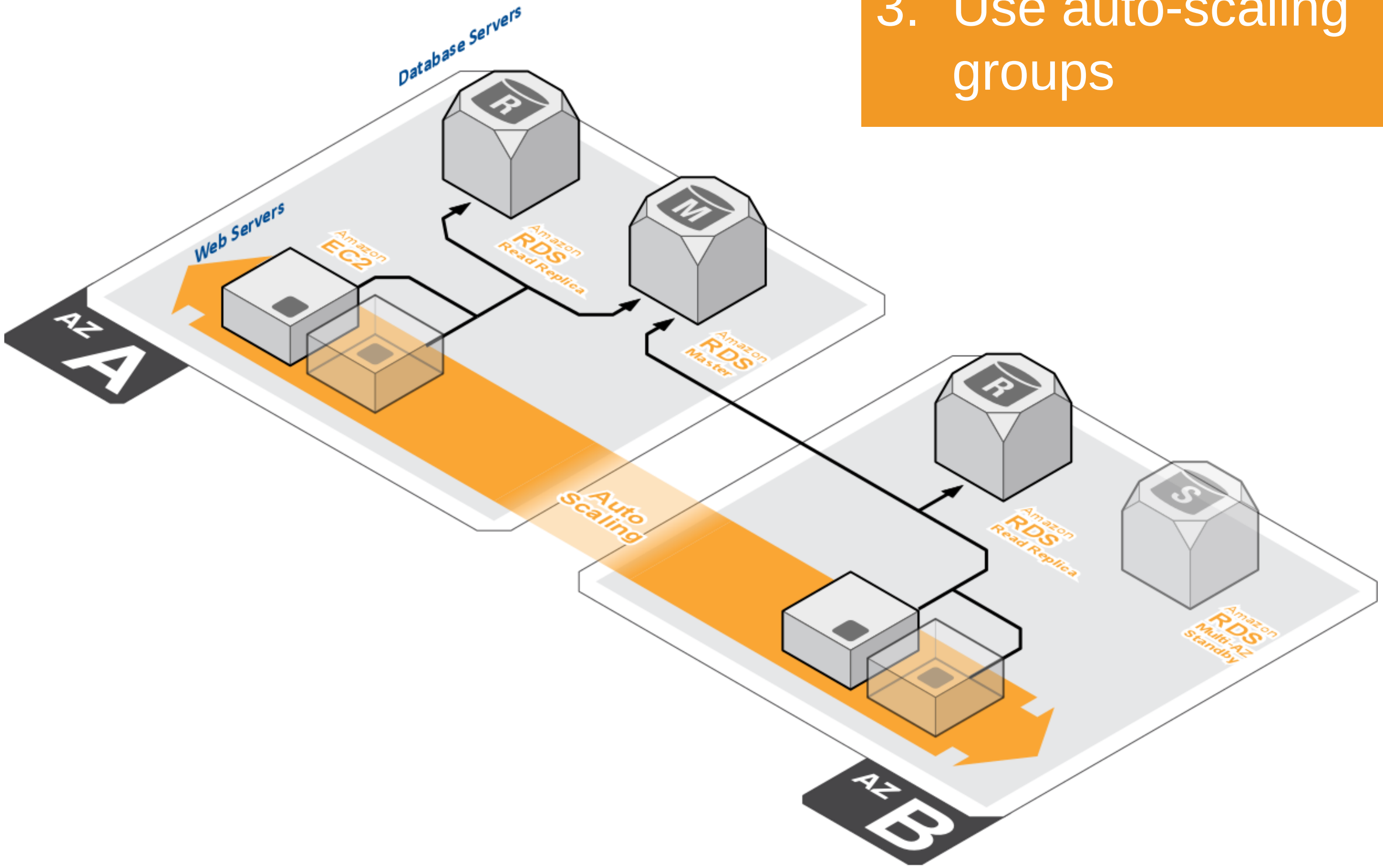
1. Use multiple availability zones



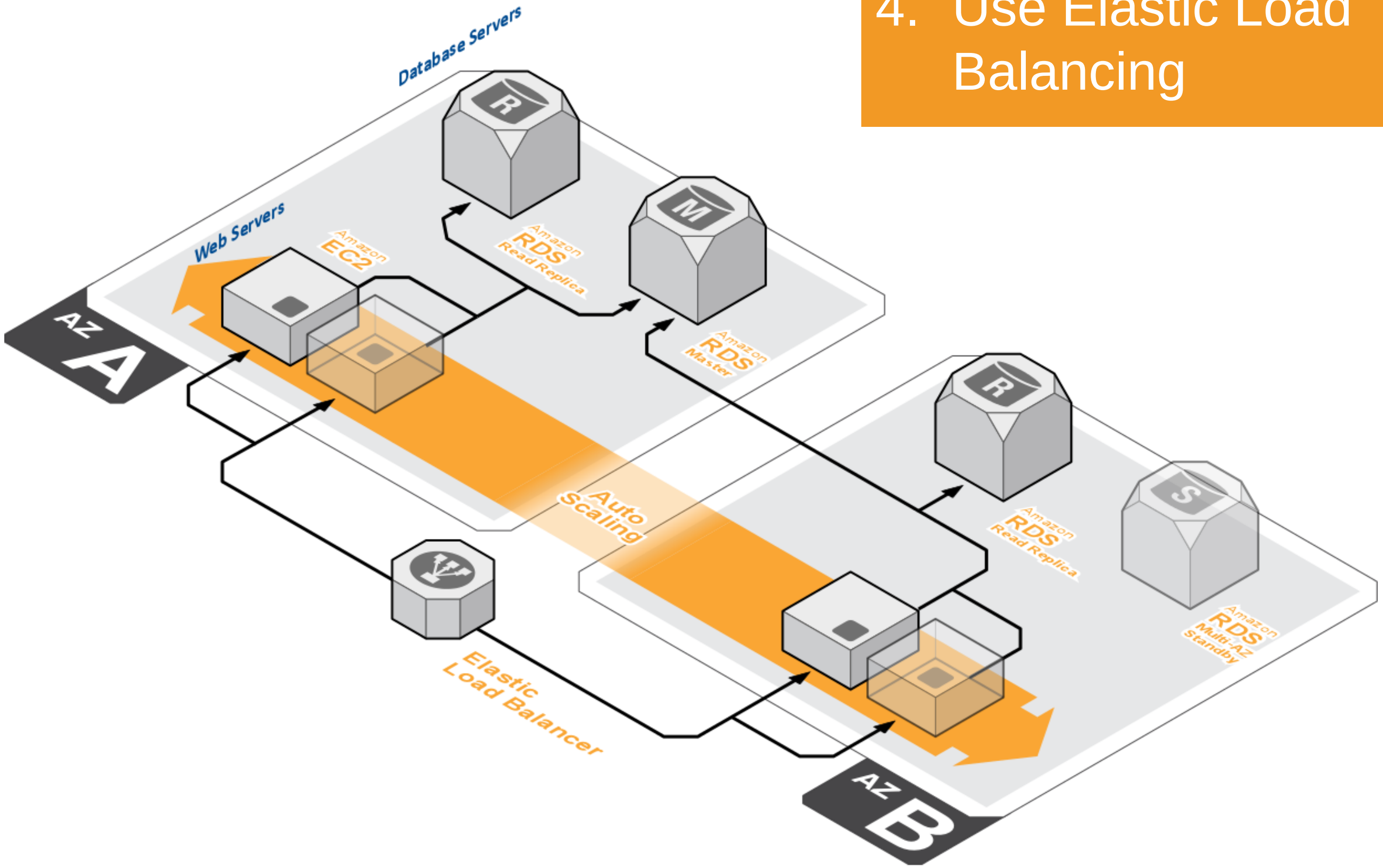
2. Use RDS with replicas and slaves



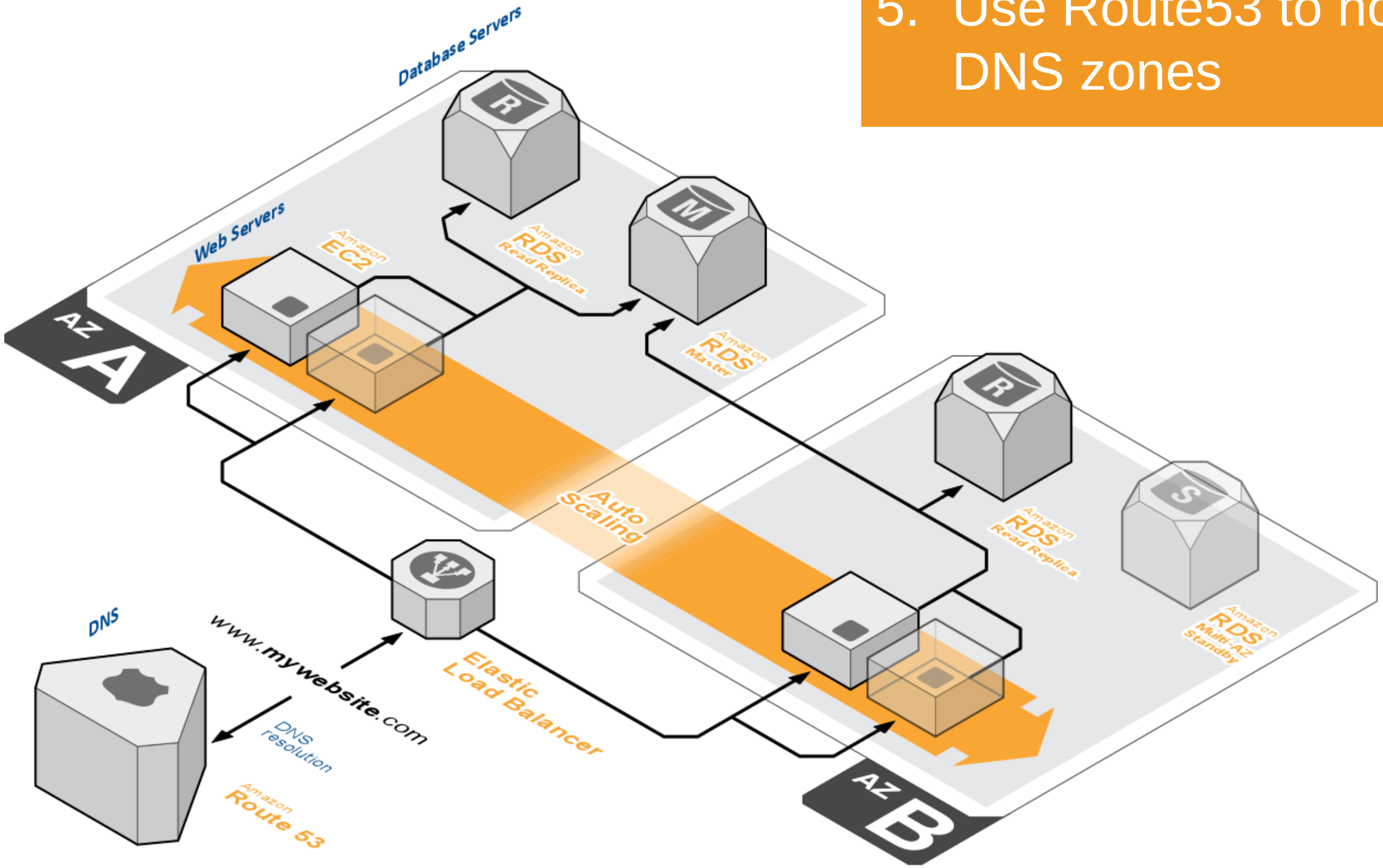
3. Use auto-scaling groups



4. Use Elastic Load Balancing



5. Use Route53 to host DNS zones



Find out more about model architectures in:

Building Web Scale Applications (bootstrapping track)

Architecting for High Availability (advanced track)

DAY 6

listen to Nick and Andy’s story

COMPLIANT PHONES
Protect, Comply, Empower. On demand

Amazon Web Services

One Year, in 5 days…

What do we do

- ➔ Record peoples mobile and fixed communications
  - – Voice, sms etc..
  - – Highly sensitive data
  - – Real time analytics and alerting
  - – Over 120 Financial Services organisations and other verticals

What is Important

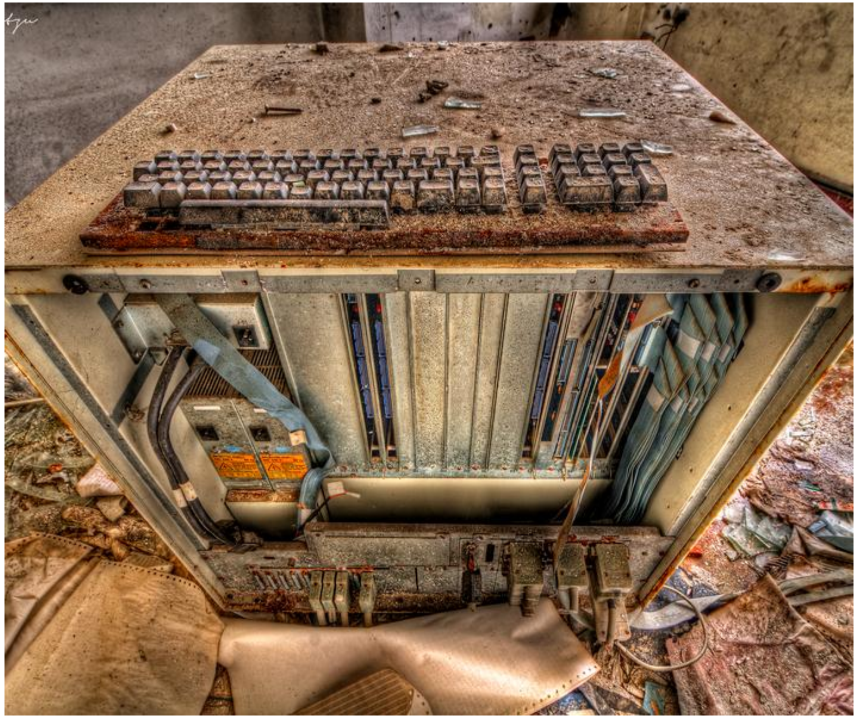
- → Security
- → Scalability
- → Flexibility
- → Global Platform

Day Zero

- → Recently Funded
- → Full Pipeline of customers
- → 6 Weeks to ‘go live’

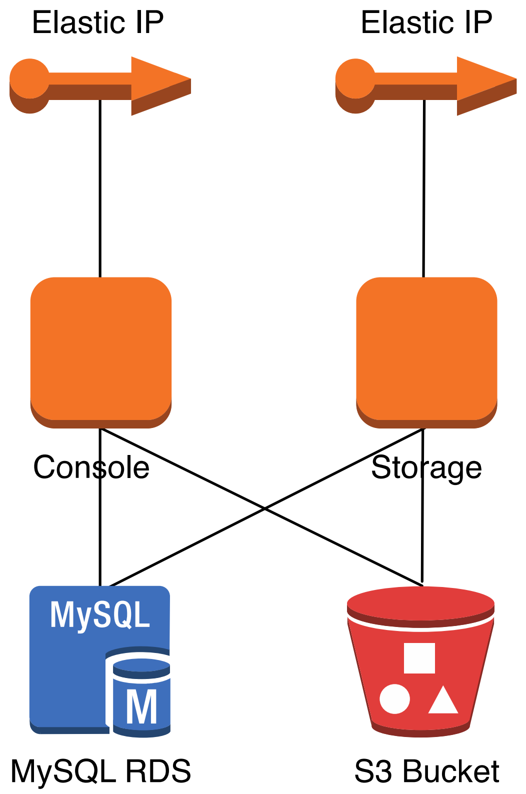
- → Web Console Service & Storage needed asap for demos

Day One – I found us a Server!



One “*slightly* fire damaged” server going cheap..

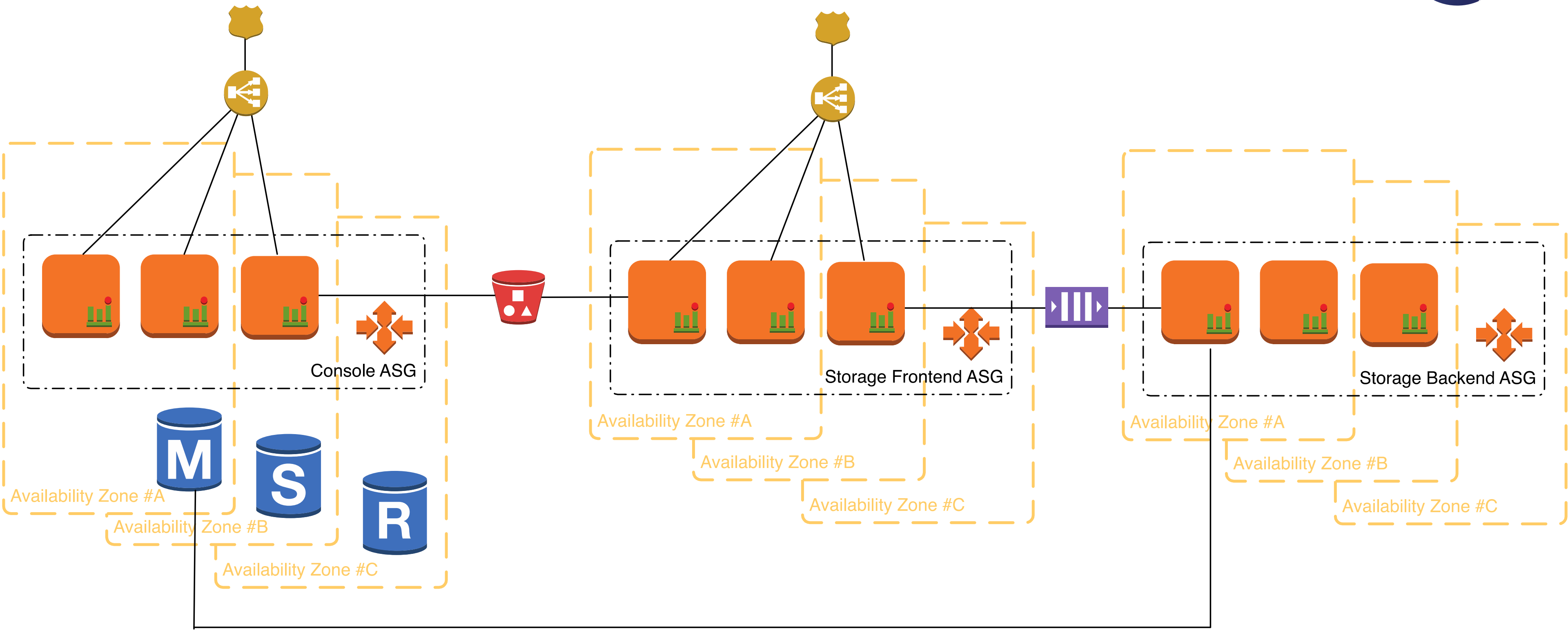
Day One – Getting Started



Day One – Lessons Learnt

- I’m dangerous on my own – I need a SysOps Guru
- Elastic IPs – Elastic Load Balancers & Route53
- RDS Monitoring – avoid that Sunday ‘brown alert’ moment, use cloud watch
- Single Instances… No
  - Stateless and multiple instances
- Decouple stuff, SQS, SWF, SNS

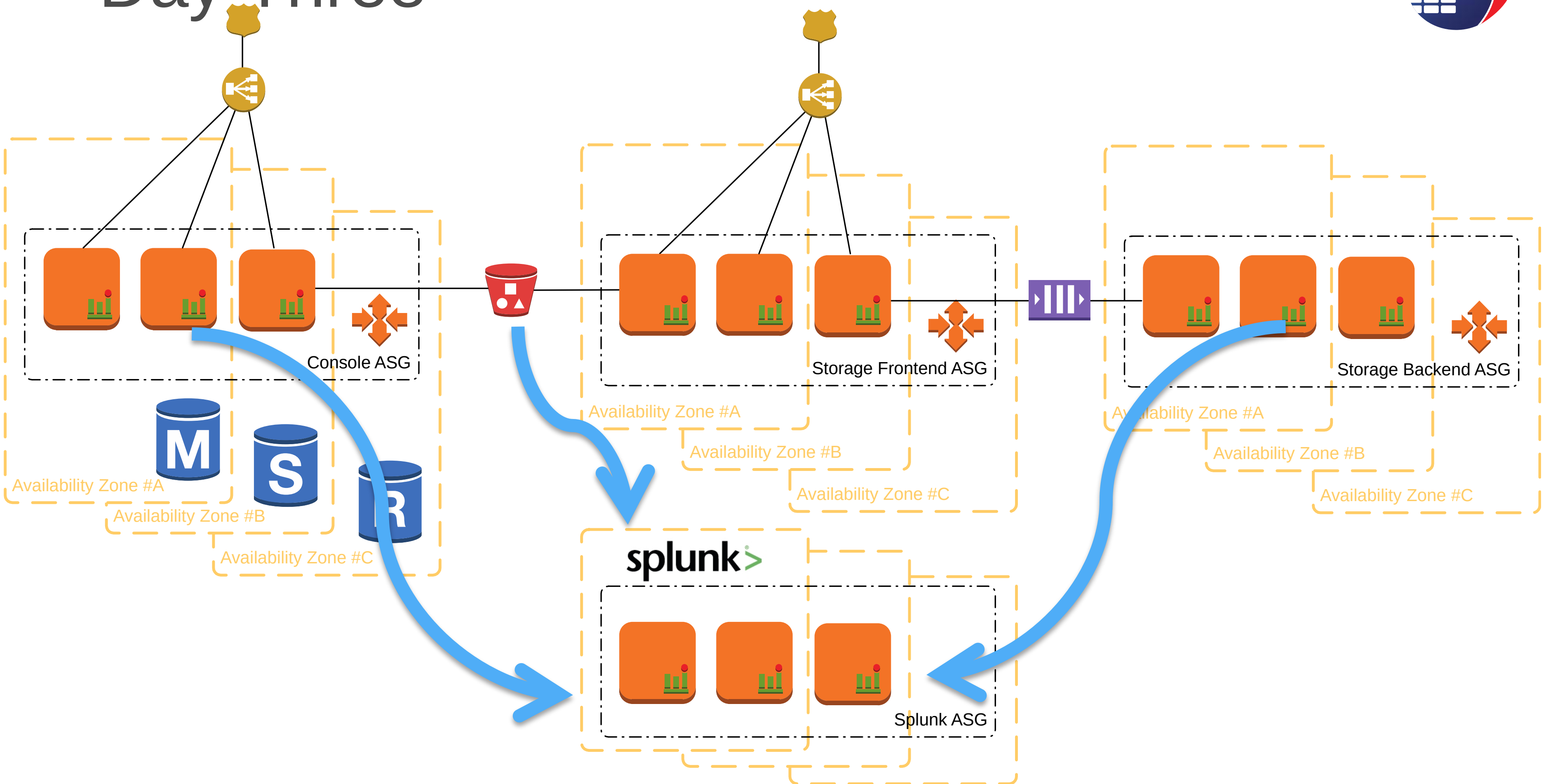
Day Two


Console ASG
Availability Zone #A
Availability Zone #B
Availability Zone #C
M
S
R
Storage Frontend ASG
Availability Zone #A
Availability Zone #B
Availability Zone #C
Storage Backend ASG
Availability Zone #A
Availability Zone #B
Availability Zone #C

Day Two – Lessons Learnt

- ➔ Larger Estate:
  - Autoscaling
  - More Logs (evidence)
  - More Metrics
  - More code, more services, more features.
- ➔ Devolve everything (where possible) to be HTTP – easier to scale
- ➔ Make things stateless – accept failure as routine

Day Three

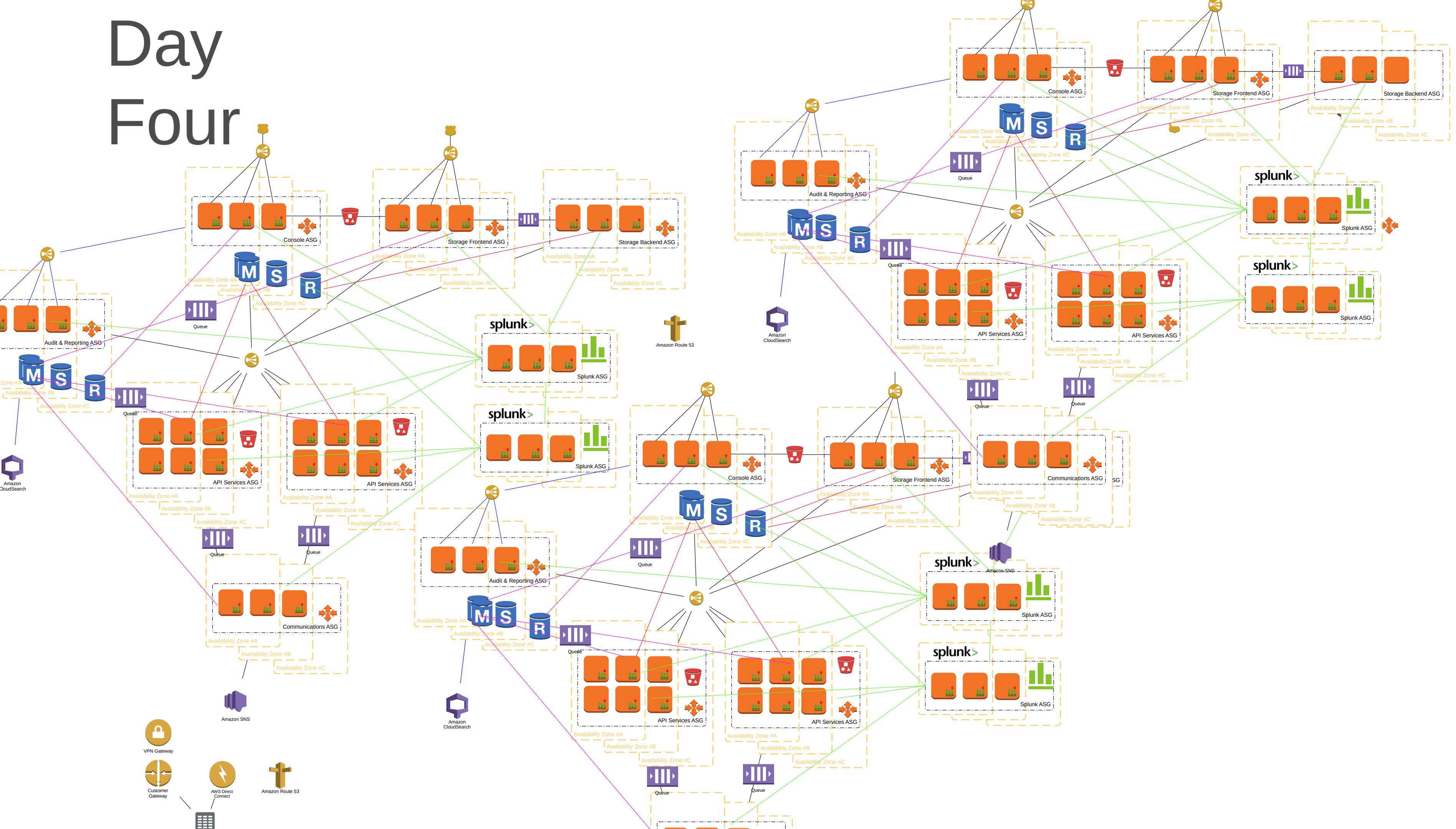


Console ASG
Availability Zone #A
Availability Zone #B
Availability Zone #C
M
S
R
Storage Frontend ASG
Availability Zone #A
Availability Zone #B
Availability Zone #C
Storage Backend ASG
Availability Zone #A
Availability Zone #B
Availability Zone #C
splunk>
Splunk ASG

Day Three – Lessons Learnt

- No human access to live
- Centralized Logging
  - We use Splunk, considered an employee in its own right
  - Consumes CloudWatch, S3 Logs, Application logs
  - Hand built alerts and filter by what is relevant
  - Evidence is key to diagnosis and resolution of any issues
- Amazon Support – cool story! Invaluable
- Think Big, you have a potential global platform at your fingertips.

Day Four



Console ASG
Storage Frontend ASG
Storage Backend ASG
Audit & Reporting ASG
API Services ASG
Communications ASG
Splunk ASG
splunk>
Availability Zone #A
Availability Zone #B
Availability Zone #C
M
S
R
Queue
Amazon CloudSearch
Amazon Route 53
Amazon SNS
VPN Gateway
Customer Gateway
AWS Direct Connect

Day Four– Lessons Learnt

- Check your growth – Easy to run more than you need.
- Document and diagram your system – roguse instances.
- Use Consolidated Billing
  - Separate accounts for Live & Test.
  - Get it off the credit card – Invoiced Billing
- Use as many of the tools as you can. - Don’t re-invent the wheel, exploit the full ecosystem.
- Read the blogs, announcements, examples, and talk to the AWS SA’s

Day Five

- Check out trusted advisor
  - What's your score?
- Leave work early!
  - You have a Global fault tolerant self healing system.
  - It notifies you if there is a problem, and then resolves itself.
  - You can keep an eye on your logs from the Pub.
- On a beer mat, do your reserved instance calculations
  - Buy reserved instances on Monday!

Lessons Learnt

- DevOps – tightly coupled development and systems teams = rapid evolution
- Building it right (evolution) allowed us to take time out and not be worried by failure
- Failure doesn't have to be all bad, if you expect it.
- Automation of testing, release and the deployment process, removes the risk from human mistake.

COMPLIANT PHONES

Protect, Comply, Empower. **On demand**

www.compliantphones.com

Summary

Lessons learned…

Monitoring, auto-scaling, de-coupling ◀
Stateless, expect failure ◀
Lock-down live, AWS support rocks, think big! ◀
Consolidated billing, follow the blog ◀
Reserved instances ◀

aws.typepad.com