The AWS DevOps combo

I hope you're hungry!

Julien Simon, Principal Technical Evangelist
julsimon@amazon.fr
@julsimon

amazon web services

Thousands of teams

× Microservice architecture

× Continuous delivery

× Multiple environments

= 50 million deployments a year

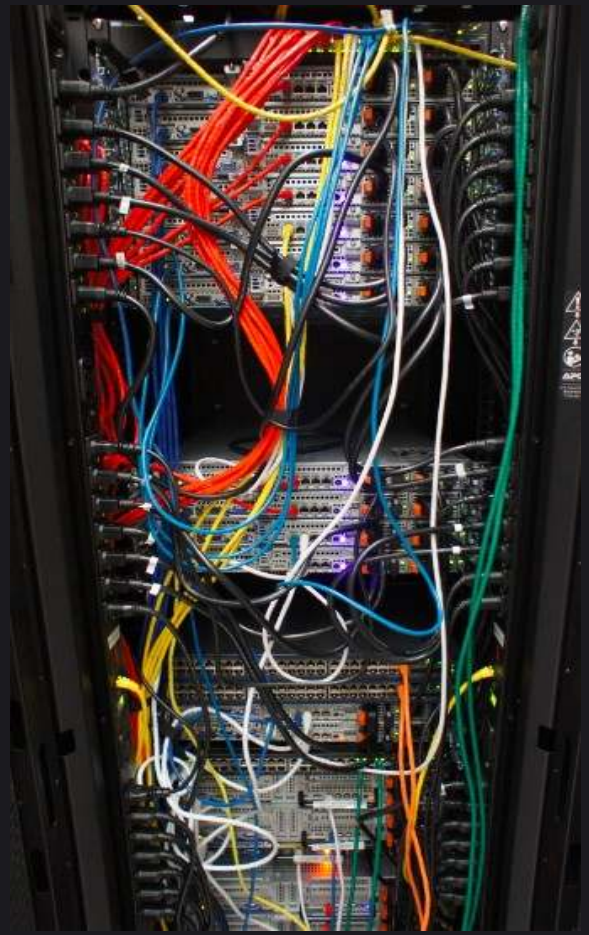
(1.5 deployment every second)

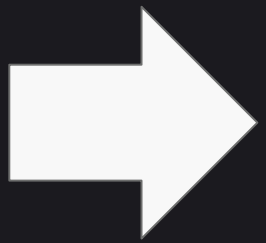
The AWS DevOps menu

- Infrastructure as code: AWS CloudFormation
- Configuration management: AWS OpsWorks
  - Won't cover it today
  - 3 words: "Managed Chef server" ☺
- Continuous Integration & Deployment: AWS Code*
- Container management: Amazon ECS & ECR

AWS CloudFormation

The problem



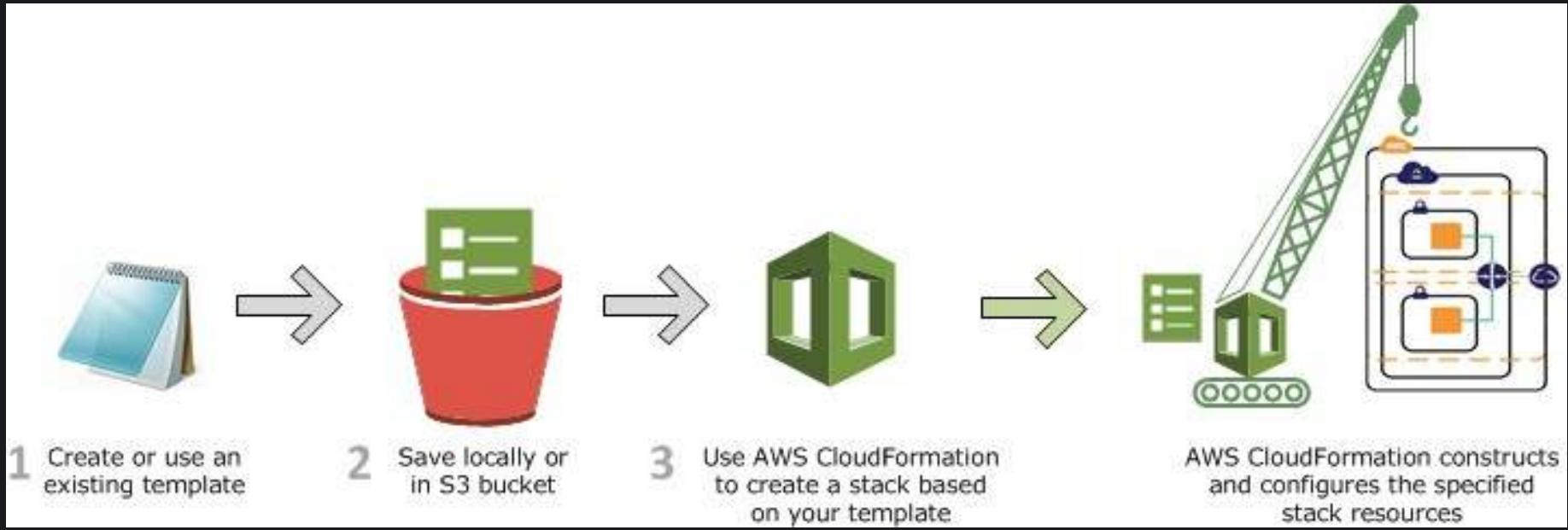


```
"Conditions" : {
  "HaveNoOtherRoles"  : { "Fn::Equals" : [{"Ref" : "OtherRoles"}, ""]},
  "HaveEbs"           : { "Fn::Not" : [{ "Fn::Equals" : [{"Ref" : "EbsVolumeSize"}, "0"]}] },
  "HaveEbsSnapshotId" : { "Fn::Not" : [{ "Fn::Equals" : [{"Ref" : "EbsSnapshotId"}, ""]}] },
  "HaveAdditionalTagKey": { "Fn::Not" : [{ "Fn::Equals" : [{"Ref" : "AdditionalTagKey"}, ""]}] },
  "HaveAdditionalTagValue": { "Fn::Not" : [{ "Fn::Equals" : [{"Ref" : "AdditionalTagValue"}, ""]}] },
  "HaveSSL": { "Fn::Not" : [{ "Fn::Equals" : [{"Ref" : "SSLPort"}, "0"]}] },
  "IsHTTP" : { "Fn::Equals" : [{"Ref" : "ElbProtocol"}, "HTTP"]},
  "HaveSpotPrice" : { "Fn::Not" : [{ "Fn::Equals" : [{"Ref" : "SpotPrice"}, ""]}]}
},
"Resources": {
  "AutoScalingGroup": {
    "Type": "AWS::AutoScaling::AutoScalingGroup",
    "UpdatePolicy" : {
      "AutoScalingRollingUpdate" : {
      "MaxBatchSize" : "1",
      "MinInstancesInService" : "0",
      "PauseTime" : "PT15M",
      "WaitOnResourceSignals": "true"
      }
    },
    "Properties": {
      "LaunchConfigurationName": { "Ref": "LaunchConfig" },
      "LoadBalancerNames": [ { "Ref": "ElasticLoadBalancer" } ],
      "MinSize": { "Ref": "MinPoolSize" },
      "MaxSize": { "Ref": "MaxPoolSize" },
      "AvailabilityZones": { "Fn::FindInMap": ["AZConfig", "AvailabilityZones", "all"] },
      "VPCZoneIdentifier": { "Ref": "EC2SubnetsIds" },
      "Tags" : [
        { "Fn::If": [
            "HaveAdditionalTagKey",
            {
              "Key" : { "Ref": "AdditionalTagKey" },
              "Value": {
                "Fn::If": [
                  "HaveAdditionalTagValue",
                  {"Ref": "AdditionalTagValue"},
                  ""
                ]
              },
              "PropagateAtLaunch": "true"
            },
            {"Ref" : "AWS::NoValue"}
          ]
        },
        { "Key" : "Name", "Value" : { "Fn::Join" : [ "-", [ { "Ref" : "ServiceName"}, {"Ref" : "EnvironmentName"
        { "Key" : "cost", "Value" : { "Ref" : "Cost" }, "PropagateAtLaunch": "true" },
        { "Key" : "environment", "Value": { "Ref" : "EnvironmentName"}, "PropagateAtLaunch": "true" }
      ]
    }
  }
```

AWS CloudFormation

- Fundamental service used to automate deployment and configuration of AWS resources (VPC, EC2, RDS, etc.)
- Infrastructure as code: versionable, auditable, testable
- https://aws.amazon.com/cloudformation/
- Pricing: no extra charge ☺

AWS CloudFormation



CloudFormation template

- JSON or YAML document which describes a configuration to be deployed in an AWS account
  - Resources, Parameters, Outputs, etc.
- When deployed, refers to a stack of resources
- Not a script, a document

Some use cases for AWS CloudFormation

- Used internally by many AWS products (Elastic Beanstalk, ECS, etc.)
- Building as many environments as you need
  - Dev, staging, pre-production, production
  - Same architecture, different sizing → template + parameters
- Deploying in a different region
- Green / blue deployments
- Disaster Recovery

Managing AWS CloudFormation with the CLI

```
$ aws cloudformation validate-template --template-body
file://template.json

$ aws cloudformation create-stack --template-body
file://template.json --stack-name MyTemplate --region eu-
west-1

$ aws cloudformation get-template --stack-name MyTemplate

$ aws cloudformation update-stack --stack-name MyTemplate
--template-body file://template.json

$ aws cloudformation delete-stack --stack-name MyTemplate
```

Demo

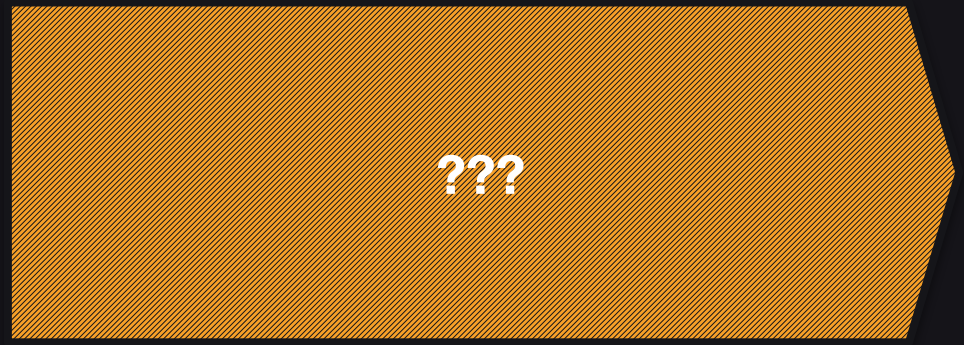
Starting stuff, updating it, deleting it, yeah!

AWS Code*

The problem

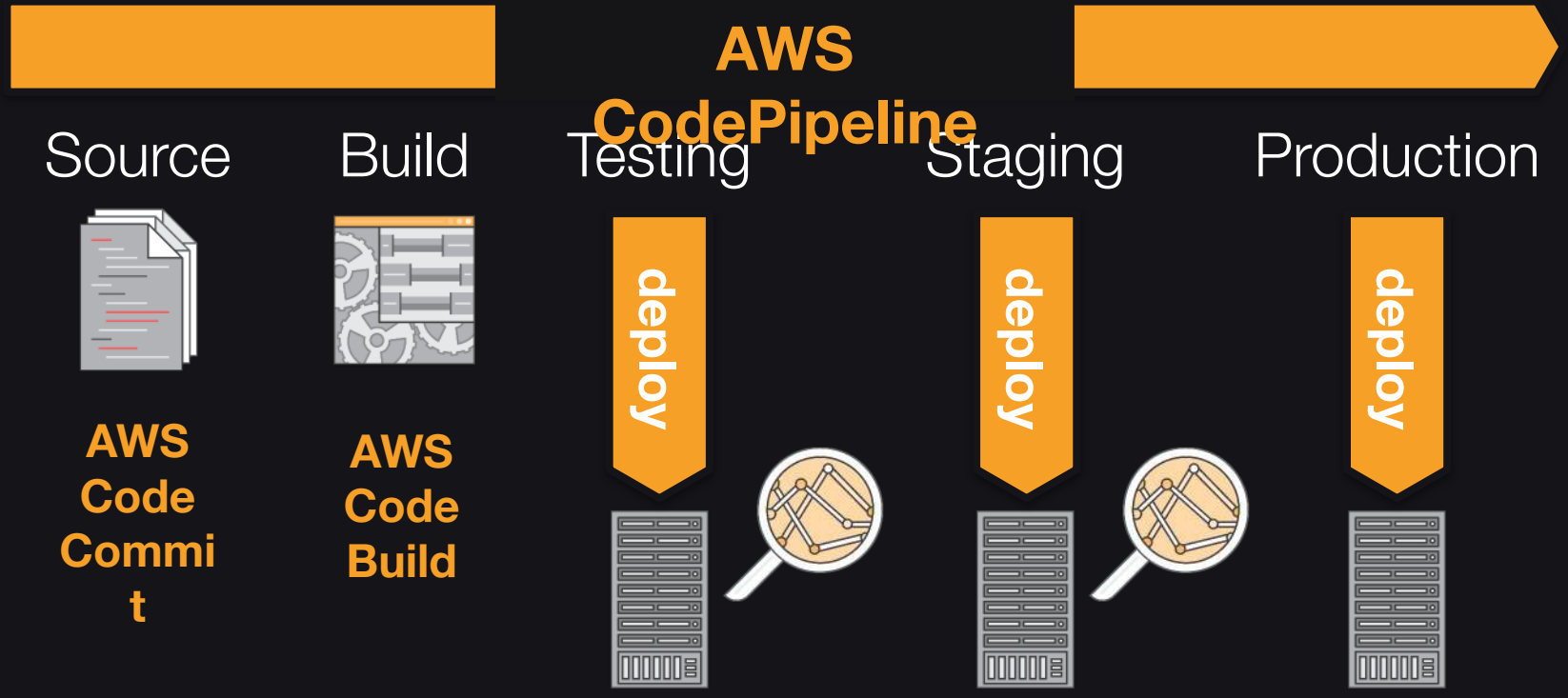
developers

services

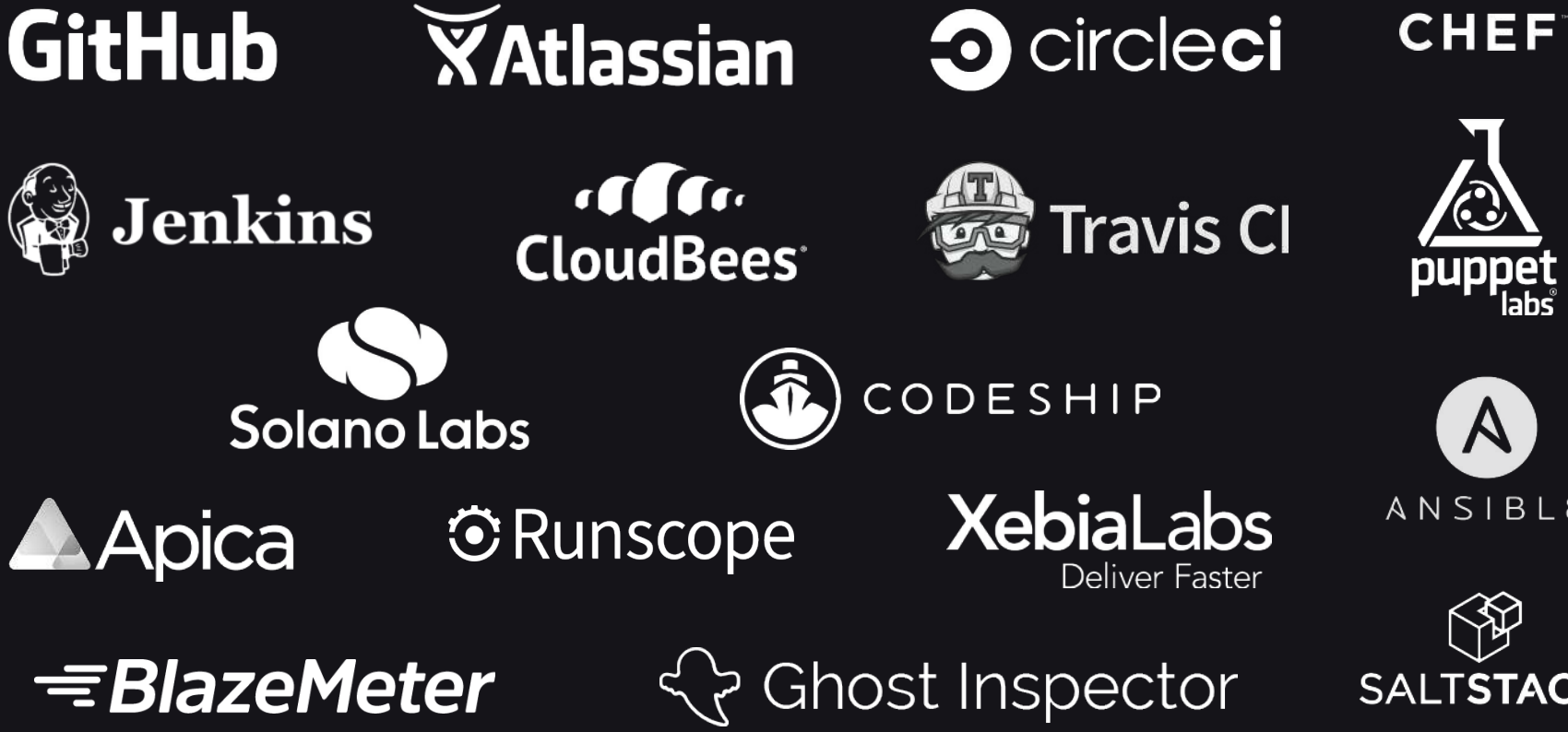


delivery pipeline

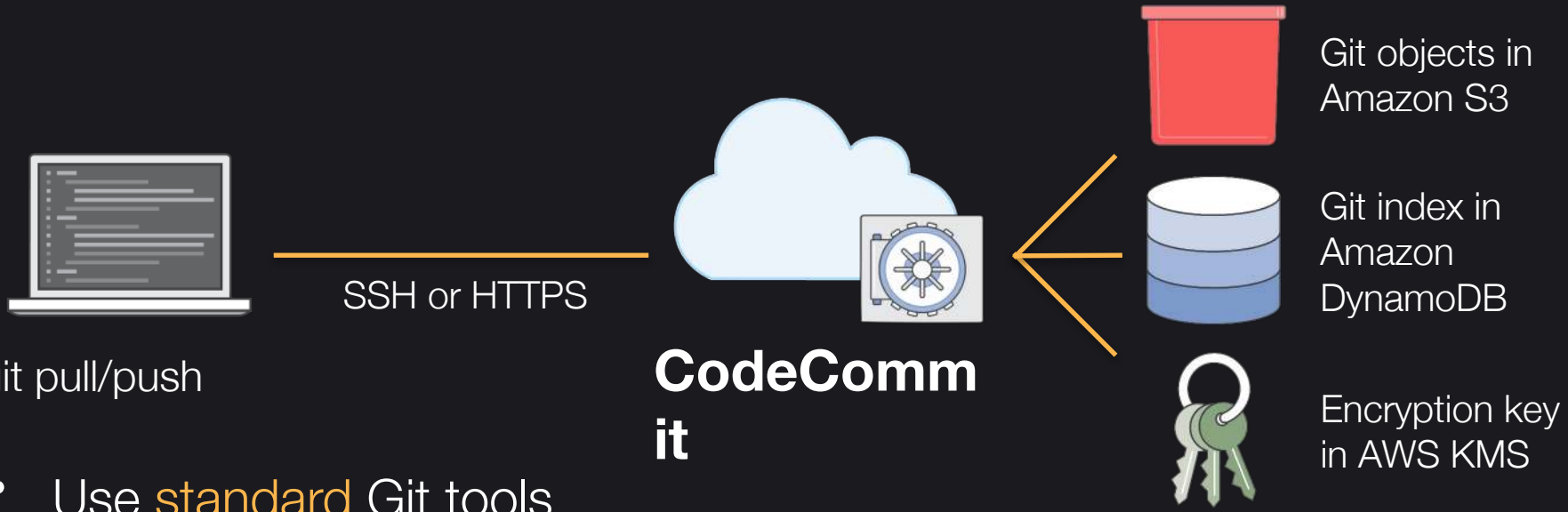
Setting up a delivery pipeline



AWS Code* partners



AWS CodeCommit

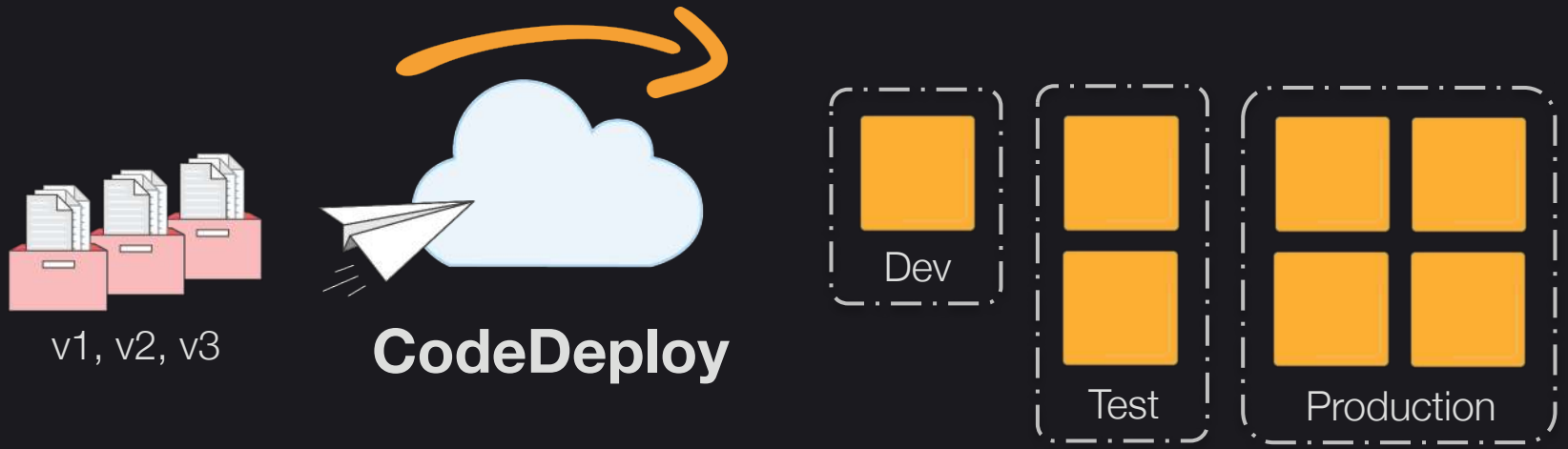


- Use standard Git tools
- Scalability, availability and durability of Amazon S3
- Encryption at rest with customer-specific keys
- Pricing: first 5 users free, then $1 / user / month
- https://aws.amazon.com/codecommit/

AWS CodeBuild

- New service launched at re:Invent 2016
- Managed build environments (Linux only)
- Pull sources from Github, S3 or CodeCommit
- Build on an AWS-provided image or on your Docker container
- Supported environments : “base”, Android, Java, Go, Python, Ruby, Go, Docker
- Build commands: inline or in buildspec.yml file
- Pricing starts at $0.005 per minute (free tier available)
- https://aws.amazon.com/codebuild/

AWS CodeDeploy



- Easy and reliable deployments (zero downtime, rollbacks)
- Scale with ease (support for Auto Scaling groups)
- Deploy to any server (Linux / Windows, EC2 / on-premise)
- Pricing : no extra charge for EC2
- https://aws.amazon.com/codedeploy/

AWS CodePipeline

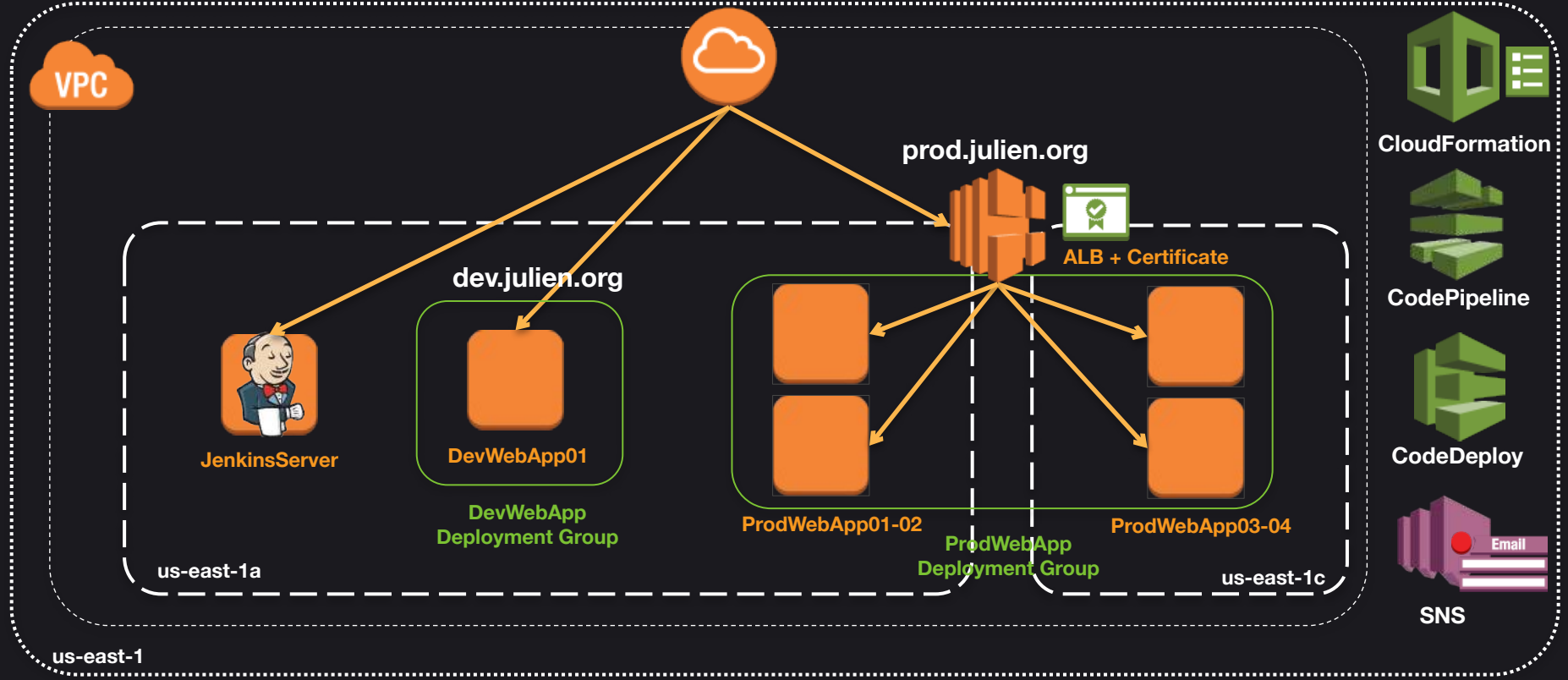
Source	Build	Deploy	Approve	Deploy
1) Pull	1) Build 2) Unit test	1) Deploy 2) QA	SNS	1) Deploy canary 2) Deploy prod

- Define stages: Source, Build, Test, Deploy, Invoke, Approve
- Connect to best-of-breed tools
- Accelerate your release process
- Consistently verify each release
- Pricing: $1 / active pipeline / month
- https://aws.amazon.com/codepipeline/

AWS Code* demo

Source (GitHub) → Build (Jenkins) → Deploy Dev (CodeDeploy) → Approve (SNS Email) → Deploy Prod (CodeDeploy)

Code
+ appspec.yml
+ scripts



CloudFormation

CodePipeline

CodeDeploy

SNS

APPROVAL NEEDED: AWS CodePipeline app-name-Pipeline for action My_Approval

Email JS 11:15

Hello,

The following Approval action is waiting for your response:

--Pipeline Details--

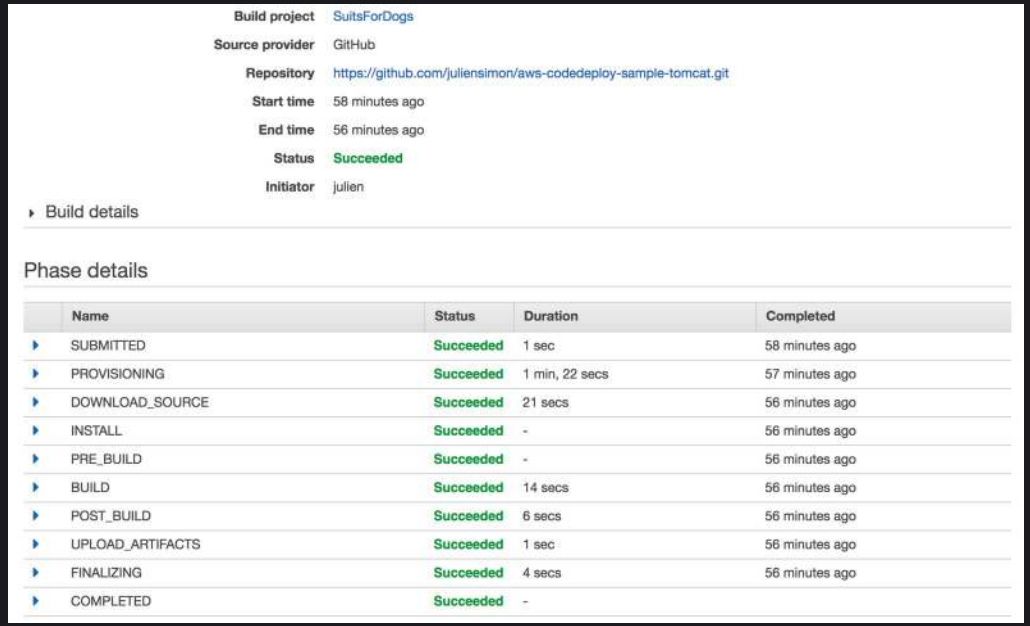
Pipeline name: app-name-Pipeline
Stage name: Approval
Action name: My_Approval
Region: us-east-1

--Approval Details--

Content to review: http://dev.julien.org
Approve or reject: https://console.aws.amazon.com/codepipeline/home?region=us-east-1#/view/app-name-Pipeline/Approval/My_Approval/approve/0bba2e19-4c19-4a74-87de-cdc1f9612613
Additional information: Please review this deployment
Deadline: This review request will expire on 2016-10-12T09:15Z

Sincerely,
Amazon Web Services

Building our app with CodeBuild



Build project	SuitsForDogs
Source provider	GitHub
Repository	https://github.com/juliensimon/aws-codedeploy-sample-tomcat.git
Start time	58 minutes ago
End time	56 minutes ago
Status	Succeeded
Initiator	julien

▸ Build details

Phase details

Name	Status	Duration	Completed
SUBMITTED	Succeeded	1 sec	58 minutes ago
PROVISIONING	Succeeded	1 min, 22 secs	57 minutes ago
DOWNLOAD_SOURCE	Succeeded	21 secs	56 minutes ago
INSTALL	Succeeded	-	56 minutes ago
PRE_BUILD	Succeeded	-	56 minutes ago
BUILD	Succeeded	14 secs	56 minutes ago
POST_BUILD	Succeeded	6 secs	56 minutes ago
UPLOAD_ARTIFACTS	Succeeded	1 sec	56 minutes ago
FINALIZING	Succeeded	4 secs	56 minutes ago
COMPLETED	Succeeded	-	

buildspec.yml

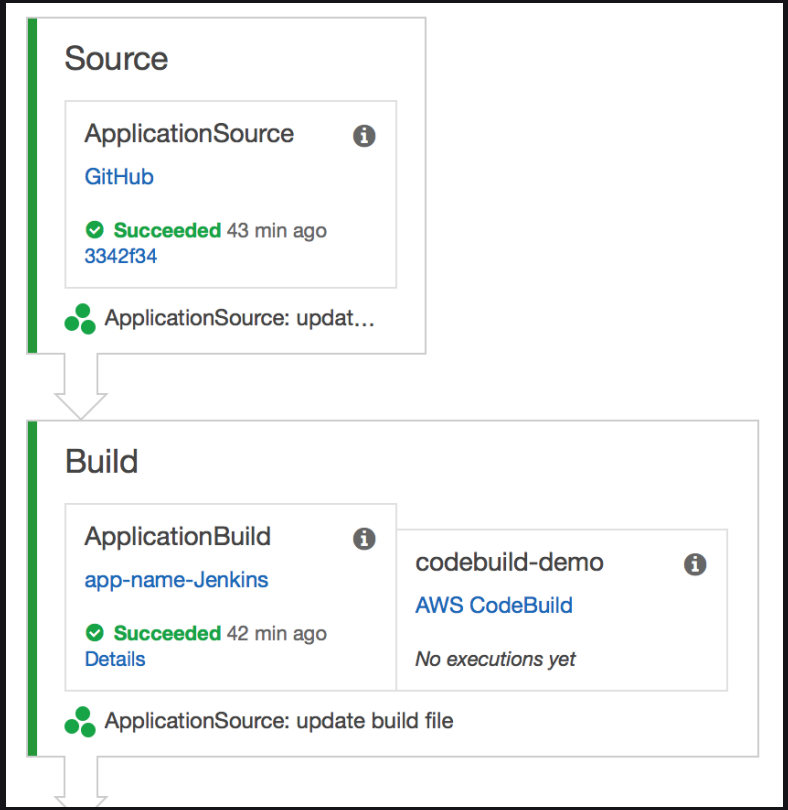
```
version: 0.1

phases:
  build:
    commands:
      - echo Build started on `date`
      - mvn test
  post_build:
    commands:
      - echo Build completed on `date`
      - mvn package
artifacts:
  files:
    - target/SampleMavenTomcatApp.war
```

Adding CodeBuild to the pipeline

You can run multiple builds in parallel

- Split the CI process
- Build a debug version
- Build for multiple targets
- …



Amazon ECS and ECR

The problem

Given a certain amount of processing power and memory,

how can we best manage an arbitrary number of apps running in Docker containers?

Modern cluster orchestration

Amazon EC2 Container Service (ECS)

- https://aws.amazon.com/ecs/
- Pricing: no extra charge

Amazon EC2 Container Registry (ECR)

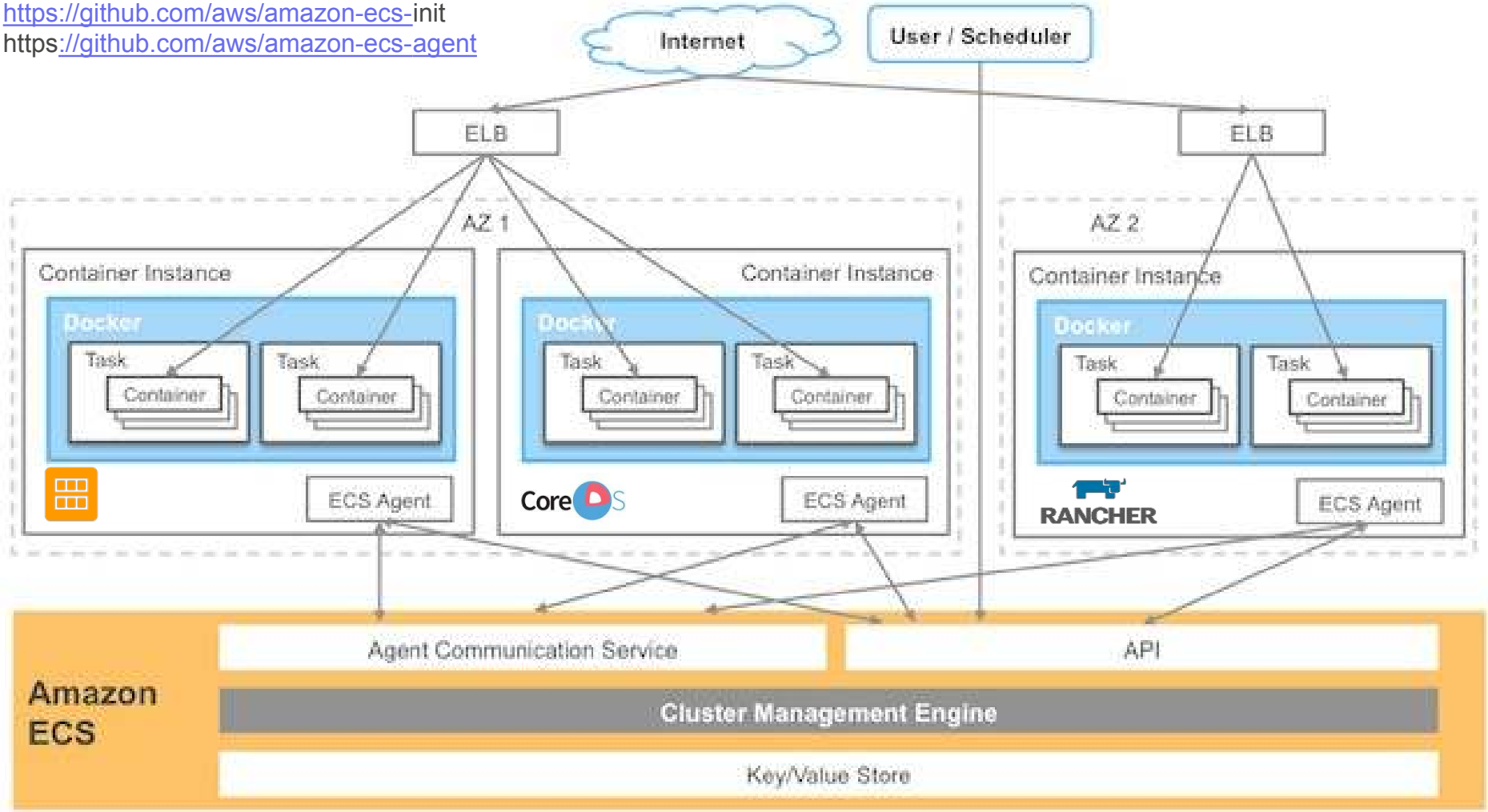
- https://aws.amazon.com/ecr/
- Pricing: $0.10 / GB / month + outgoing traffic

Distributed state management

Scalable scheduling

Built-in high availability

https://github.com/aws/amazon-ecs-init
https://github.com/aws/amazon-ecs-agent



http://www.allthingsdistributed.com/2015/07/under-the-hood-of-the-amazon-ec2-container-service.html

Managing Docker images with ECR

https://github.com/awslabs/ecs-demo-php-simple-app

```
$ aws ecr create-repository --repository-name php-simple-app
    --region us-east-1

$ aws ecr get-login --region us-east-1

<run docker login command provided as output>

$ docker build –t php-simple-app .

$ docker tag php-simple-app:latest
ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com/php-simple-app:latest

$ docker push ACCOUNT_ID.dkr.ecr.us-east-1.amazonaws.com php-simple-app:latest
```

Demo: Amazon ECS ‘Hello World’

https://github.com/aws/amazon-ecs-cli

```
$ ecs-cli configure --cluster myCluster --region eu-west-1
$ ecs-cli up --keypair myKey --capability-iam –size 3
```

```
$ ecs-cli compose service up
$ ecs-cli compose service ps
$ ecs-cli compose service scale 3
```

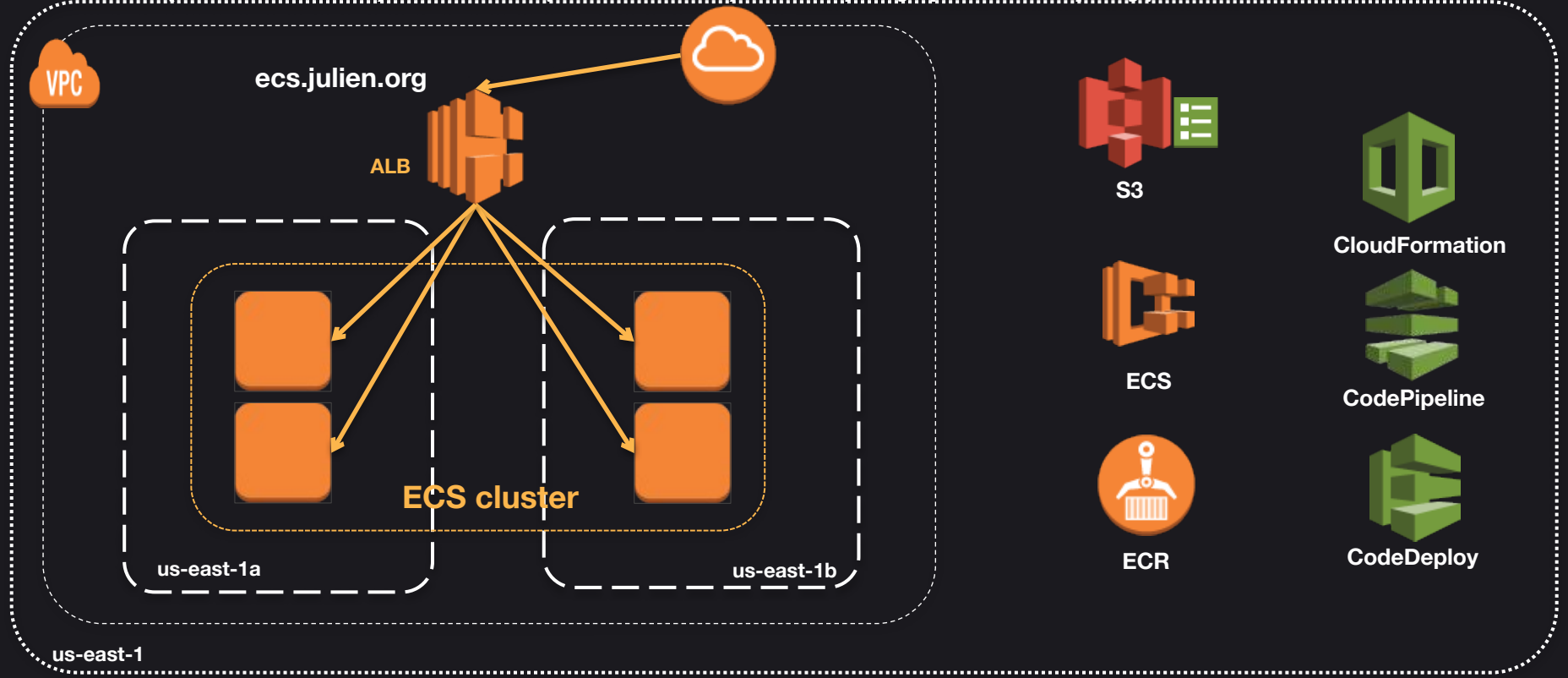
```
$ ecs-cli compose service stop
$ ecs-cli compose service delete
$ ecs-cli down myCluster --force
```

Compose file

```
php-demo:
  image: ACCOUNT_ID.dkr.ecr.us-
east-1.amazonaws.com/php-simple-app
  cpu_shares: 100
  mem_limit: 134217728
  ports:
    - "80:80"
  entrypoint:
    - "/usr/sbin/apache2"
    - "-D"
    - "FOREGROUND"
```

Demo: Continuous Deployment on Amazon ECS

Source (GitHub + S3) → Build (CodeBuild) → Deploy (CodeDeploy)



Closing words

- Automation is a key factor in technical & business agility
- You can use the same tools as Amazon.com!
- Zero dev infrastructure to purchase & manage
- Minimal cost (none for CF, CodeDeploy and ECS)
- Compatible with your existing CI/CD tools

- Get started and tell us what you think ☺
  - http://aws.amazon.com/free
  - http://console.aws.amazon.com/

Resources

https://blogs.aws.amazon.com/application-management

https://blogs.aws.amazon.com/application-management/post/Tx2CIB02ZO05ZII/Explore-Continuous-Delivery-in-AWS-with-the-Pipeline-Starter-Kit

https://aws.amazon.com/about-aws/whats-new/2016/11/aws-codepipeline-introduces-aws-cloudformation-deployment-action/

https://aws.amazon.com/fr/blogs/compute/continuous-deployment-to-amazon-ecs-using-aws-codepipeline-aws-codebuild-amazon-ecr-and-aws-cloudformation/

http://www.allthingsdistributed.com/2014/11/amazon-ec2-container-service.html

http://www.allthingsdistributed.com/2015/04/state-management-and-scheduling-with-ecs.html

http://www.allthingsdistributed.com/2015/07/under-the-hood-of-the-amazon-ec2-container-service.html

Tons of re:Invent videos on Youtube!

More content you may like

Deep Dive on Continuous Delivery
https://www.youtube.com/watch?v=Py0DmiIkxHM
Running Docker clusters on AWS
https://www.youtube.com/watch?v=_fwVuC672Ck

YouTube: https://www.youtube.com/user/juliensimonfr/
Slideshare: http://fr.slideshare.net/JulienSIMON5/

Ευχαριστώ !

See you in May at DevIt ☺
http://devitconf.org

Julien Simon, Principal Technical Evangelist
julsimon@amazon.fr
@julsimon