



Docker & AWS

Fabrizio Ciacchi

Explained simple for PHP Developers :)



Who am I?

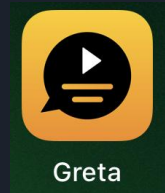
Fabrizio Ciacchi / @ciacchi / ciacchi.it

Lead Software Developer @ Greta & Starks Apps GmbH.

Worked previously in Rocket Internet and Spryker Systems

PHP Developer from more than 10 years

I'm working with Symfony from version 2+





I want to tell a story...

... how I started the migration of Greta App:

- Legacy Application on Hosted Server

I have been requested to:

- Create the whole AWS deployment system
- Migrate to a new Symfony Application
- Ensure security and best practices



Step One / Clean-up

- Clean-up your passwords and conf (use .env)
- Clean-up your data - you should have 'demo' data for your dev and staging machines. It also facilitate tests and development
 - Put DB and Content in different repositories
- Update your application:
 - Latest version of PHP, MySQL and other softwares
 - Latest version of your framework
 - If you use it, check and update your composer
- Move your cache and tmp files to Redis (or other storage)



Step two / Dockerize

- Very simple:
 - A virtual machine with a configuration file
- Why we use it?
 - Faster - it depends from the host system
 - Configuration based - it removes a lot of Admin/DevOPs issues
 - Reproducible environment
- What to use:
 - Bitnami Nginx
 - Bitnami PHP
 - Official MySQL
 - Bitnami has a lot of packages (Laravel, Redis, etc)
 - And... [nibirrayy/docker-smtp](#)



Step three / Still Docker

Why those packages?

- Bitnami packages are highly configurable
- MySQL package allows to import SQL files

Also remember:

- Use Docker-hub
- port mapping is to expose ports outside
- You'll probably need an SMTP server
- Use mounted volumes (ie: composer)
- Avoid custom Docker images (you'll have to compile your own image)

AWS

▼ All services			
Compute	Developer Tools	Machine Learning	Mobile
EC2	CodeStar	Amazon SageMaker	AWS Amplify
Lightsail	CodeCommit	Amazon Comprehend	Mobile Hub
ECR	CodeBuild	AWS DeepLens	AWS AppSync
ECS	CodeDeploy	Amazon Lex	Device Farm
EKS	CodePipeline	Machine Learning	
Lambda	Cloud9	Amazon Polly	
Batch	X-Ray	Rekognition	
Elastic Beanstalk		Amazon Transcribe	
Serverless Application Repository	Customer Enablement	Amazon Translate	AR & VR
	AWS IQ	Amazon Personalize	Amazon Sumerian
	Support	Amazon Forecast	
	Managed Services	Amazon Textract	Application Integration
Storage		AWS DeepRacer	Step Functions
S3	Robotics		Amazon EventBridge
EFS	AWS RoboMaker		Amazon MQ
FSx		Analytics	Simple Notification Service
S3 Glacier		Athena	Simple Queue Service
Storage Gateway	Blockchain	EMR	SWF
AWS Backup	Amazon Managed Blockchain	CloudSearch	
		Elasticsearch Service	Customer Engagement
Database		Kinesis	Amazon Connect
RDS	Satellite	QuickSight	Pinpoint
DynamoDB	Ground Station	Data Pipeline	Simple Email Service
ElastiCache		AWS Glue	
Neptune	Management & Governance	AWS Lake Formation	Business Applications
Amazon Redshift	AWS Organizations	MSK	Alexa for Business
Amazon QLDB	CloudWatch		Amazon Chime
Amazon DocumentDB	AWS Auto Scaling	Security, Identity, & Compliance	WorkMail
	CloudFormation	IAM	
Migration & Transfer	CloudTrail	Resource Access Manager	End User Computing
AWS Migration Hub	Config	Cognito	WorkSpaces
Application Discovery Service	OpsWorks	Secrets Manager	AppStream 2.0
Database Migration Service	Service Catalog	GuardDuty	WorkDocs
Server Migration Service	Systems Manager	Inspector	WorkLink
AWS Transfer for SFTP	Trusted Advisor	Amazon Macie	
Snowball	Control Tower	AWS Single Sign-On	Internet of Things
DataSync	AWS License Manager	Certificate Manager	IoT Core
	AWS Well-Architected Tool		Amazon FreeRTOS

- So many services!!!
- Don't get overwhelmed
- You might need a CC (ok you have also free tiers)

EC2 - your 'machine'

The first thing to do is to set-up your Staging machine.

Why Staging?

- First you want to start simple
- Also can be used as 'base' for Production



The screenshot shows the AWS Marketplace listing for the Amazon ECS-Optimized Amazon Linux 2 AMI. It includes the AWS logo, the product name, a star rating of 0, the version 2.0.20190709, and the publisher Amazon Web Services. The pricing is listed as \$0.0047 to \$38.688/hr, including EC2 charges and other AWS usage fees. A 'Free tier eligible' badge is present. The operating system details are Linux/Unix, Amazon Linux 2.0.20181017, 64-bit (x86) Amazon Machine Image.

	Family	Type	vCPUs	Memory (GiB)
<input type="checkbox"/>	General purpose	t2.nano	1	0.5
<input type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1
<input type="checkbox"/>	General purpose	t2.small	1	2
<input checked="" type="checkbox"/>	General purpose	t2.medium	2	4

Configure Security Groups & Roles

Type ⓘ	Protocol ⓘ	Port Range ⓘ
SSH	TCP	22
HTTP	TCP	80
HTTPS	TCP	443

Add Rule

Additionally you might want the IAM role

[AmazonS3FullAccess](#)

At the end you need to generate and download a PEM certificate!

Create role

Review

Provide the required information below and review this role before you create it.

Role name*

Use alphanumeric and '+,=,@-_' characters. Maximum 64 characters.

Role description

Allows EC2 instances in an ECS cluster to access ECS.

Maximum 1000 characters. Use alphanumeric and '+,=,@-_' characters.

Trusted entities

AWS service: ec2.amazonaws.com

Policies



AmazonEC2ContainerServiceforEC2Role [↗](#)



Set-up Application + S3 Bucket

Clone your repository(-ies) in `/home/ec2-user` (do not store GIT credentials)

But sym-link them under ie: `/opt/app`

S3 Bucket is nearly infinite storage:

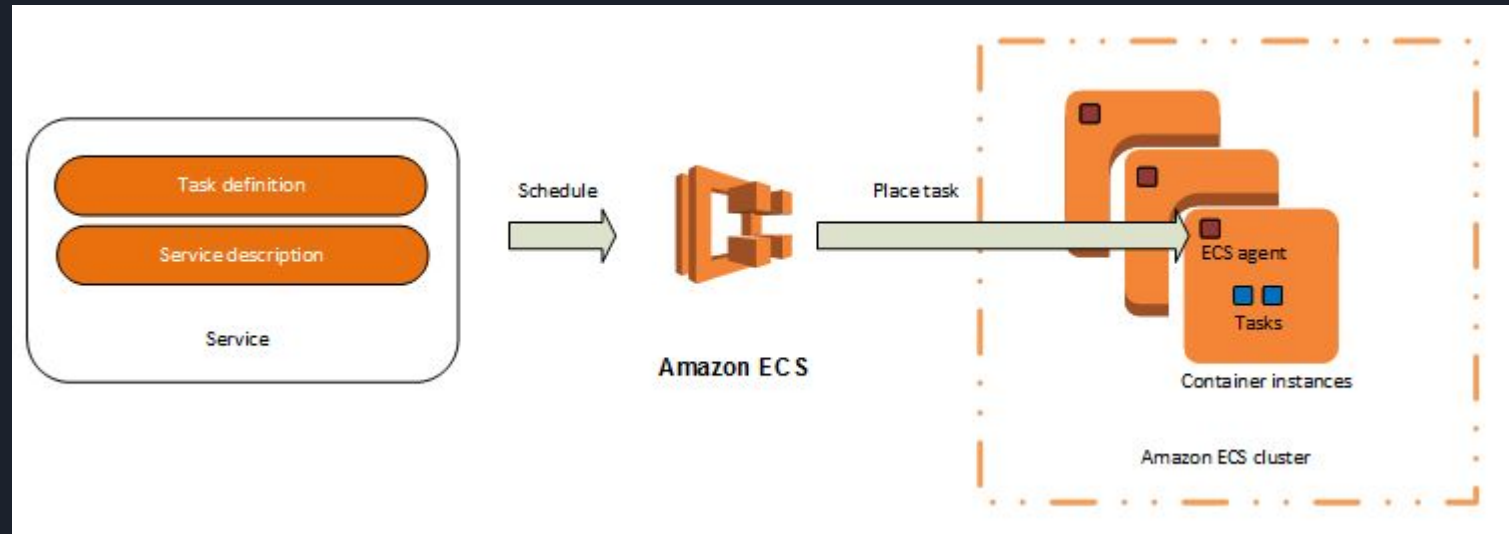
- 1 for Deployments
- 1 for Logging
- 1 for Storage (media, etc.)

User `s3fs-fuse` to mount the S3 bucket(s) in your EC2:

<https://github.com/s3fs-fuse/s3fs-fuse.git>

Re-mount at reboot with crontab (-e)

ECS - visually

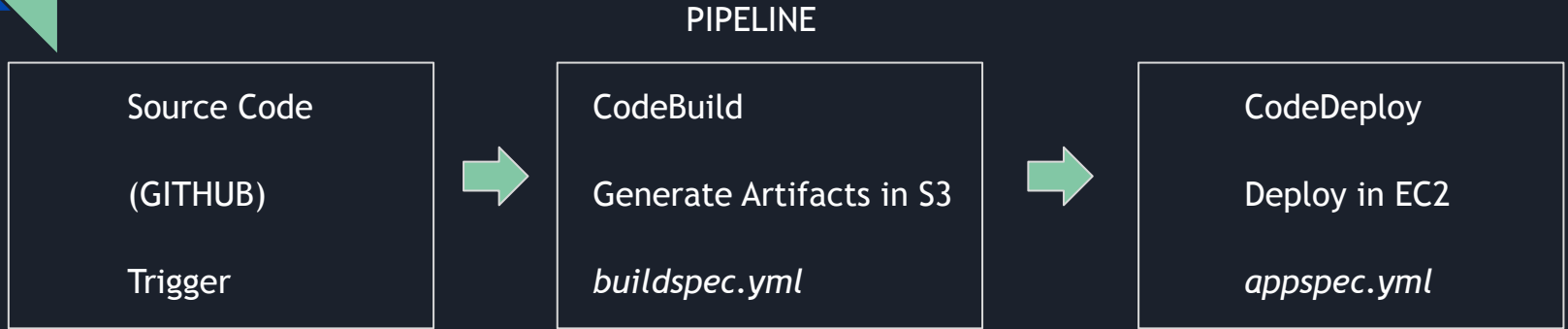




ECS - Elastic Container Service

- Task is a mapping of your Docker file
 - Create the shared mounts (your folders in EC2)
 - Set-up the min (and max) for memory and CPU unit
 - Network 'bridge'
 - Set-up services as non-essential (DB essential)
 - Add networking links so services can 'see' each other
 - Port mapping
- Create a Cluster and a Service associated with the Task.
- Install the ECS Agent in your EC2:
<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-agent-install.html>
- In etc/ecs/ecs.config set-up your ECS service name
ECS_CLUSTER=staging

CodePipeline



- Define IAM Roles
- Install the Agent:
<https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install.html>
- Link:<https://hackernoon.com/continuous-deployment-with-aws-codedeploy-github-d1eb97550b82>
- Hooks:<https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html#appspec-hooks-ecs>



Ok, but where is my website visible

Your EC2 instance has a public and private IP address. But that's not good (not secure)

- Register or transfer your domain with Route53
- Set-up an Elastic IP address (public) for your staging instance
- Point staging.myapp.com to the Elastic IP
- Allow only HTTP/HTTPS to the public IP (no SSH, only private)
- Add your SSL certificate with AWS certificate manager



What about Production?

- Create a snapshot of Staging
- Create a new EC2 instance from it
 - Change the attribute name to production
 - Download the new PEM certificate
 - Log-in inside and change the data source (DB?), env files, and ECS cluster name
- Shut it down - do not terminate (it will destroy)
- Create Launch Template from it

And then and then and then...

- Load balancer
- Target Group
- Auto-scaling + Launch Template
- Assign Elastic IP to load balancer
- Route53 to assign production to that IP (www and without)

Step 4: Register Targets

Configure Security Groups

The security groups for your instances must allow traffic from the VPC CIDR on the health check port.

Register targets with your target group. If you register a target in an enabled Availability Zone, the load balancer starts routing requests to the targets as soon as the registration process completes and the target passes the initial health checks.

Registered targets

To deregister instances, select one or more registered instances and then click Remove.

Remove

<input type="checkbox"/>	Instance	Name	Port	State	Security groups	Zone
<input type="checkbox"/>		app-production	80	● running	Amazon ECS-Optimized Amazon Linux 2 AMI-2-0-20190709-A...	

Instances

To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.

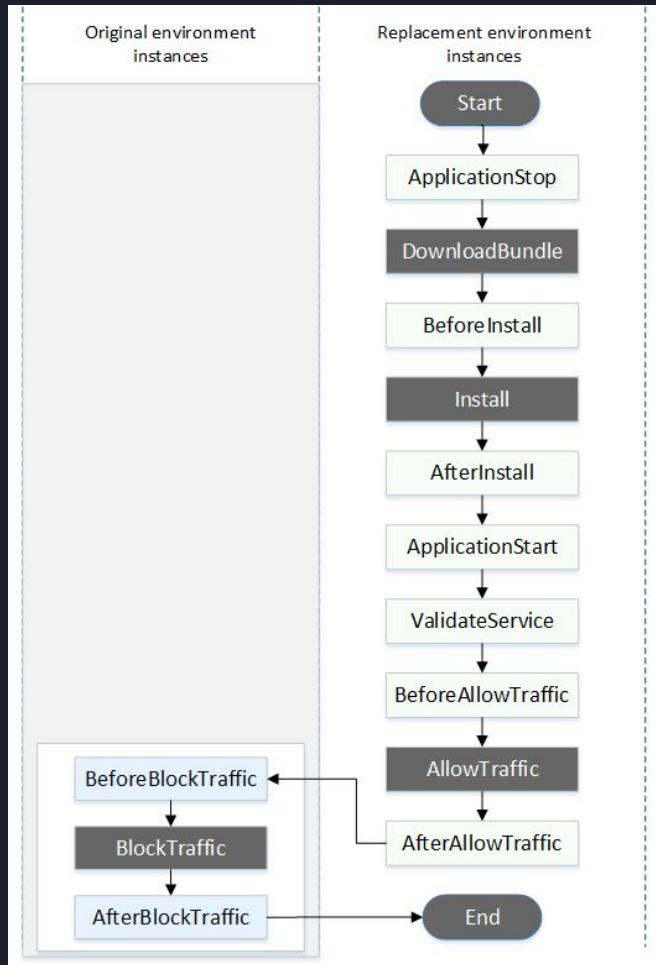
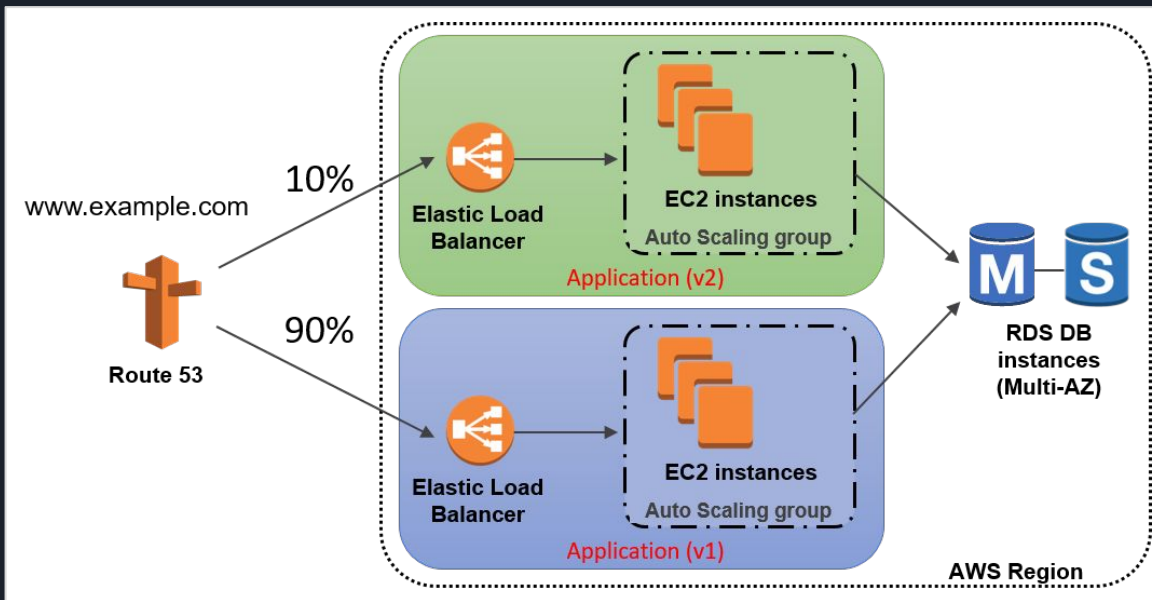
Add to registered on port:

Search Instances

<input type="checkbox"/>	Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR
<input checked="" type="checkbox"/>		app-production	● running	Amazon ECS-Optimized...	eu-central-	subnet-	
<input type="checkbox"/>		app-staging	● running	Amazon ECS-Optimized...	eu-central-	subnet-	

CodePipeline

- Create a new ECS task + service
- Create a Blue/Green deployment
- Take care of the appspec hooks (now with routing traffic become more important)





Todo:

- Monitoring
- TravisCI
- Kibana

Thanks everybody :)

@ciacchi / ciacchi.it