XPEPPERS

\ Docker on Amazon Web Services

*From **Development** to **Production** with ECS*

Paolo Latella
@LatellaPaolo

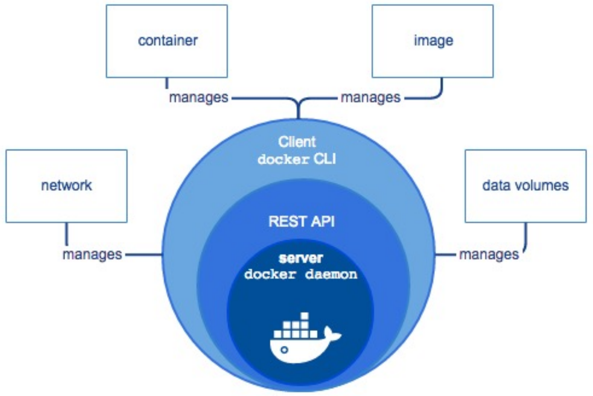
Genesis - the right tools

http://www.simpsoncrazy.com/

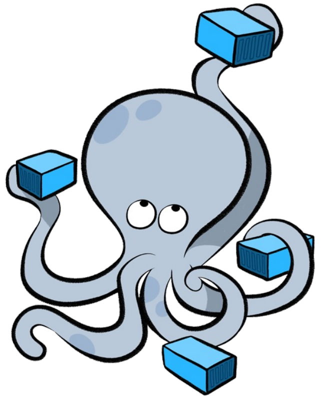
\ Genesis - the right tools

http://www.simpsoncrazy.com/

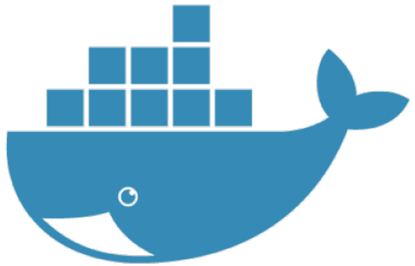
\ Development - Docker



Docker Engine



Docker Compose



Docker Hub

\ Development - Docker compose

Using Compose is basically a three-step process:

1. Define your app’s environment with a Dockerfile so it can be reproduced anywhere.
2. Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
3. Run **docker-compose up** and Compose will start and run your entire app.

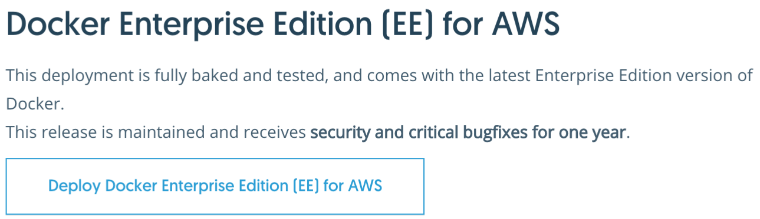
\ Development - Docker compose example

```
version: '2'
services:
  web:
    image: platella/python-yarw-1:blue
    build: .
    ports:
     - "8080"
    volumes:
     - python-microservice-one/application:/code
    cpu_shares: 128
    mem_limit: 134217728
    links:
      - redis
  redis:
    image: "redis:alpine"
    cpu_shares: 128
    mem_limit: 134217728
    ports:
     - "6379"
```

Production - Docker on AWS

- Docker on Amazon Elastic Beanstalk
- **Docker on Amazon EC2 Container Service (ECS)**
  - Elastic Container Registry
  - Task & Services
- Docker Swarm on Elastic Compute Cloud (EC2)

AWS CloudFormation



Docker Enterprise Edition (EE) for AWS

This deployment is fully baked and tested, and comes with the latest Enterprise Edition version of Docker.

This release is maintained and receives **security and critical bugfixes for one year**.

\ Production - Docker on Amazon EB

- Single Container Docker Environments
  - Run one container per instance.
  - Use a `Dockerfile` or `Dockerrun.aws.json` file
- Multi Container Docker Environments
  - Use Elastic Container Services inside a Elastic Beanstalk Environment
  - Set of containers defined in a `Dockerrun.aws.json` file

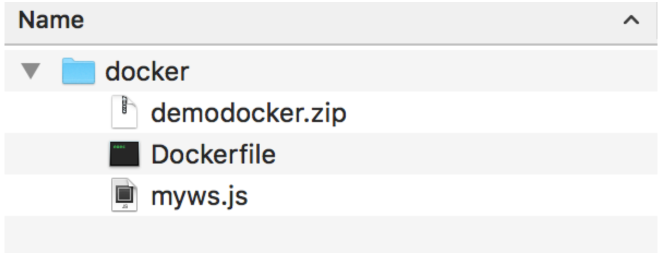
\ Docker on Amazon EB - Single Container

We can deploy from a Docker to Elastic Beanstalk by doing (OR)

- Create a **Dockerfile** to customize an image and to deploy a Docker container to Elastic Beanstalk.
- Create a **Dockerrun.aws.json** file to deploy a Docker container from an existing Docker image to Elastic Beanstalk.
- Create a **.zip** file containing your application files, any application file dependencies, the Dockerfile, and the Dockerrun.aws.json file.

Docker on Amazon EB - Single Container - Dockerfile

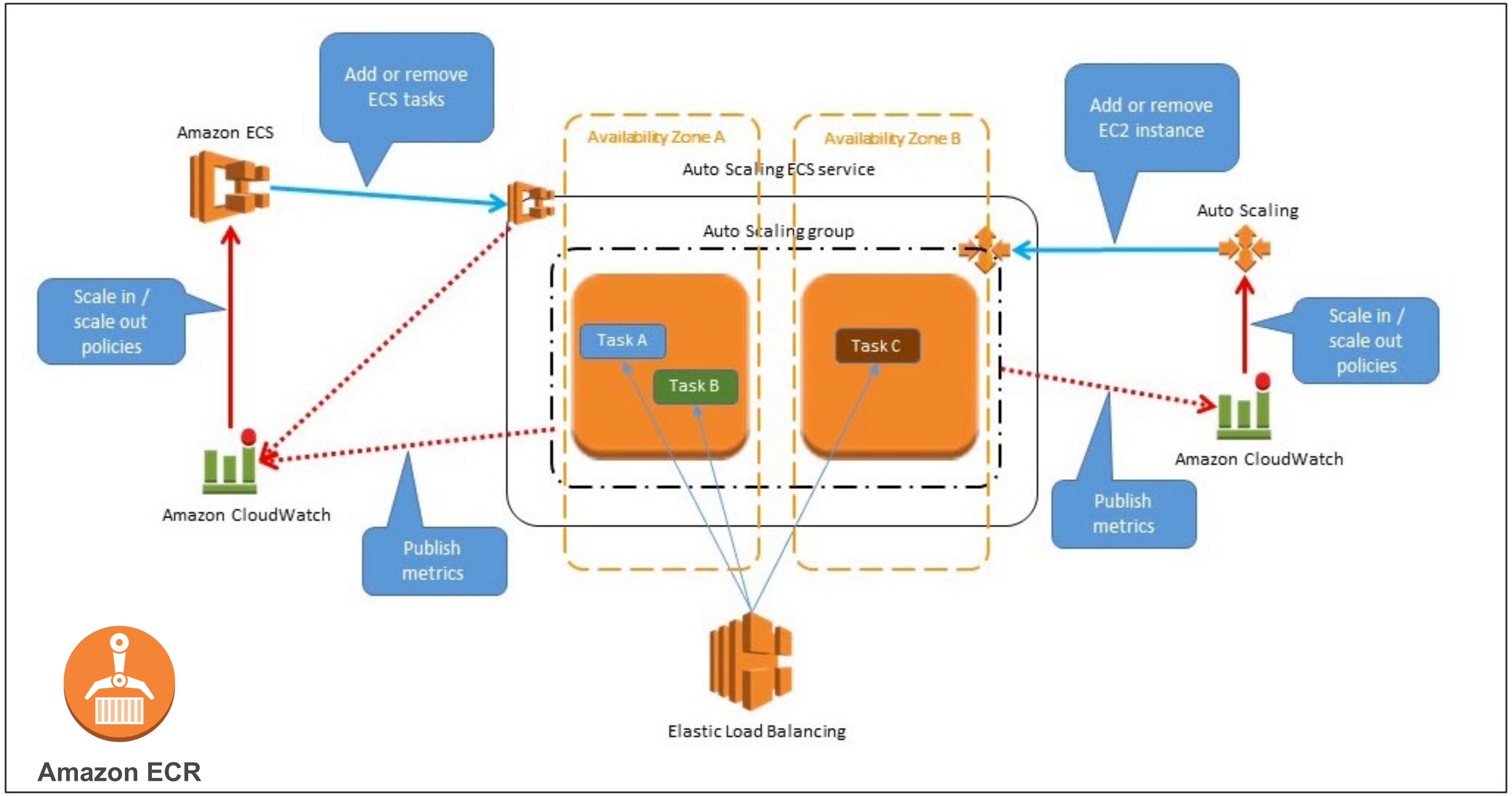
```
FROM ubuntu:14.04
# Ubuntu and nodeJS for ElasticBeanstalk
# VERSION                  0.0.1

FROM ubuntu:14.04
MAINTAINER Paolo Latella <paolo.latella@xpeppers.com>

#Port mapping
EXPOSE 8080

#Update and install nodejs
RUN apt-get update && apt-get install -y nodejs

#Copy files for nodejs application
RUN mkdir /var/www/
ADD myws.js /var/www/

#Start application
CMD /usr/bin/nodejs /var/www/myws.js
```



Docker on Amazon EB - Single Container - Dockerrun

```
{
  "AWSEBDockerrunVersion": "1",
  "Image": {
    "Name": "janedoe/image",
    "Update": "true"
  },
  "Ports": [
    {
      "ContainerPort": "1234"
    }
  ],
  "Volumes": [
    {
      "HostDirectory": "/var/app/mydb",
      "ContainerDirectory": "/etc/mysql"
    }
  ],
  "Logging": "/var/log/nginx"
}.
```

Production - Amazon EC2 Container Service



Amazon Elastic Container Service - *ecs-cli*

ECS CLI Command Line

```
ecs-cli configure --region eu-west-1 --cluster Demo-ECSCluster

ecs-cli up --keypair key --capability-iam --size 4 --instance-type c4.large

ecs-cli compose -f python-microservice1/docker-compose.yml service create

ecs-cli compose -f python-microservice1/docker-compose.yml service start

ecs-cli compose -f python-microservice1/docker-compose.yml service scale 2
```

ECS CLI supports Docker compose file syntax versions 1 and 2

http://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_CLI_reference.html

Amazon Elastic Container Service - *aws ecs*

AWS ECS Command Line

```
aws ecs create-cluster --region eu-west-1 --cluster-name “Demo-ECSCluster"

aws ecs register-task-definition --cli-input-json file://./python-yarw-1.json

aws ecs create-service --service-name python-yarw-1 --task-definition python-
yarw-1 --desired-count 2


aws ecs update-service --service python-yarw-1 --cluster Demo-ECSCluster --task-
definition python-yarw-1 --desired-count 4 --deployment-configuration
"maximumPercent=200,minimumHealthyPercent=100"
```

http://docs.aws.amazon.com/cli/latest/reference/ecs/index.html#cli-aws-ecs

\ Amazon Elastic Container Service - Task definition (1/2)

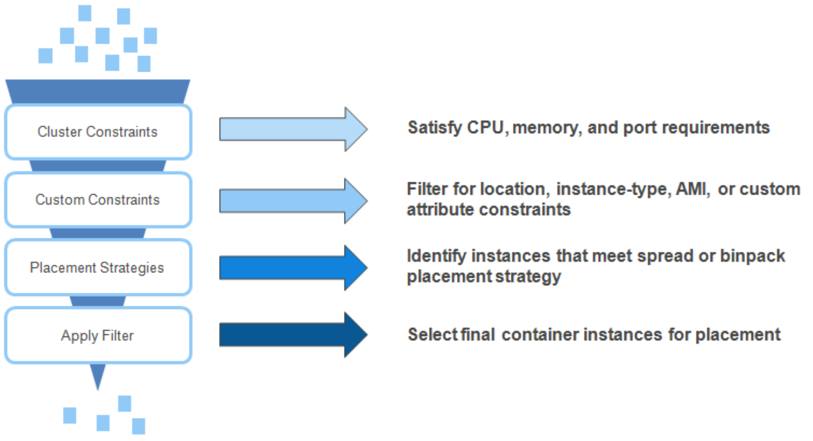
```
{
  "containerDefinitions": [
    {
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 0,
          "containerPort": 6379,
          "protocol": "tcp"
        }
      ],
      "name": "redis",
      "image": "redis:alpine",
      "cpu": 128,
    },
```

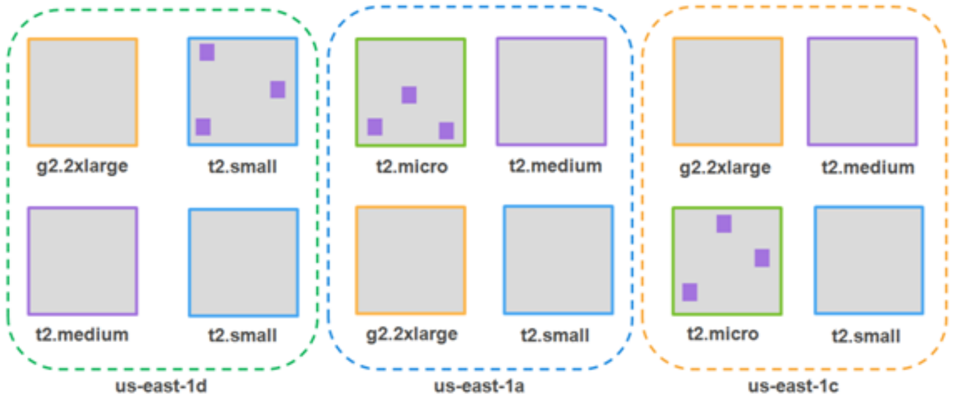
Amazon Elastic Container Service - Task definition (2/2)

```
    {
        "memory": 128,
        "portMappings": [
          {
            "hostPort": 0,
            "containerPort": 8080,
            "protocol": "tcp"
          }
        ],
        "name": "web",
        "links": [
          "redis"
        ],
        "cpu": 128,
      }
    ],
    "family": "ecscompose-python-microservice-one"
  }
```

Amazon Elastic Container Service - Placement

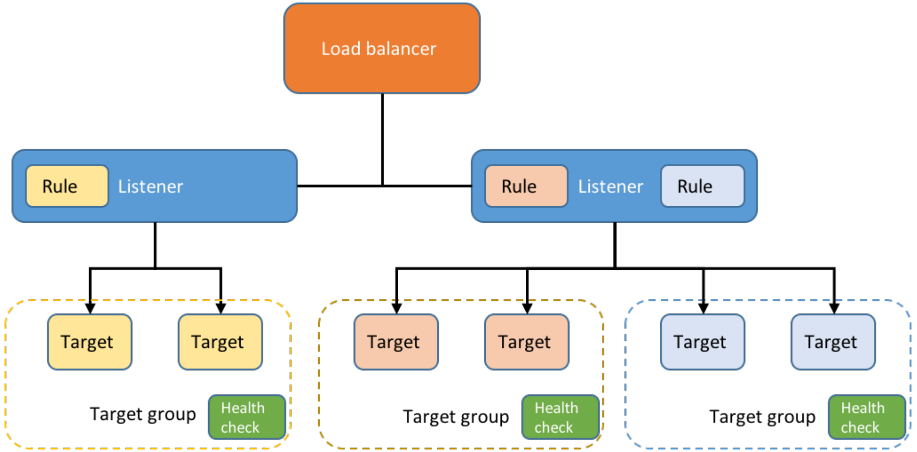
```
aws ecs create-service --service-name python-yarw-1 --task-definition python-
yarw-1 --desired-count 2  --placement-strategy
type="spread",field="attribute:ecs.availability-zone"
type="binpack",field="memory"
```


Cluster Constraints
Custom Constraints
Placement Strategies
Apply Filter
Satisfy CPU, memory, and port requirements
Filter for location, instance-type, AMI, or custom attribute constraints
Identify instances that meet spread or binpack placement strategy
Select final container instances for placement



Amazon Elastic Container Service - Load Balancer

```
aws ecs create-service --service-name python-yarw-1 --task-definition python-
yarw-1 --desired-count 2 --load-balancers
"targetGroupArn=arn:aws:elasticloadbalancing:eu-west-
1:831650818513:targetgroup/Microservices-
one/cca50273455ba775,containerName=web,containerPort=8080" --desired-count 2 --
deployment-configuration "maximumPercent=200,minimumHealthyPercent=50" --role
ECS-TestRole
```



Amazon Elastic Container Service - Blue/Green Deploy

DNS Swap

1. Create new task definition
2. Create new service
3. Create new ALB
4. Attach new service to ALB
5. Update Route53
6. CleanUP Blue Environment

Canary

Service Swap

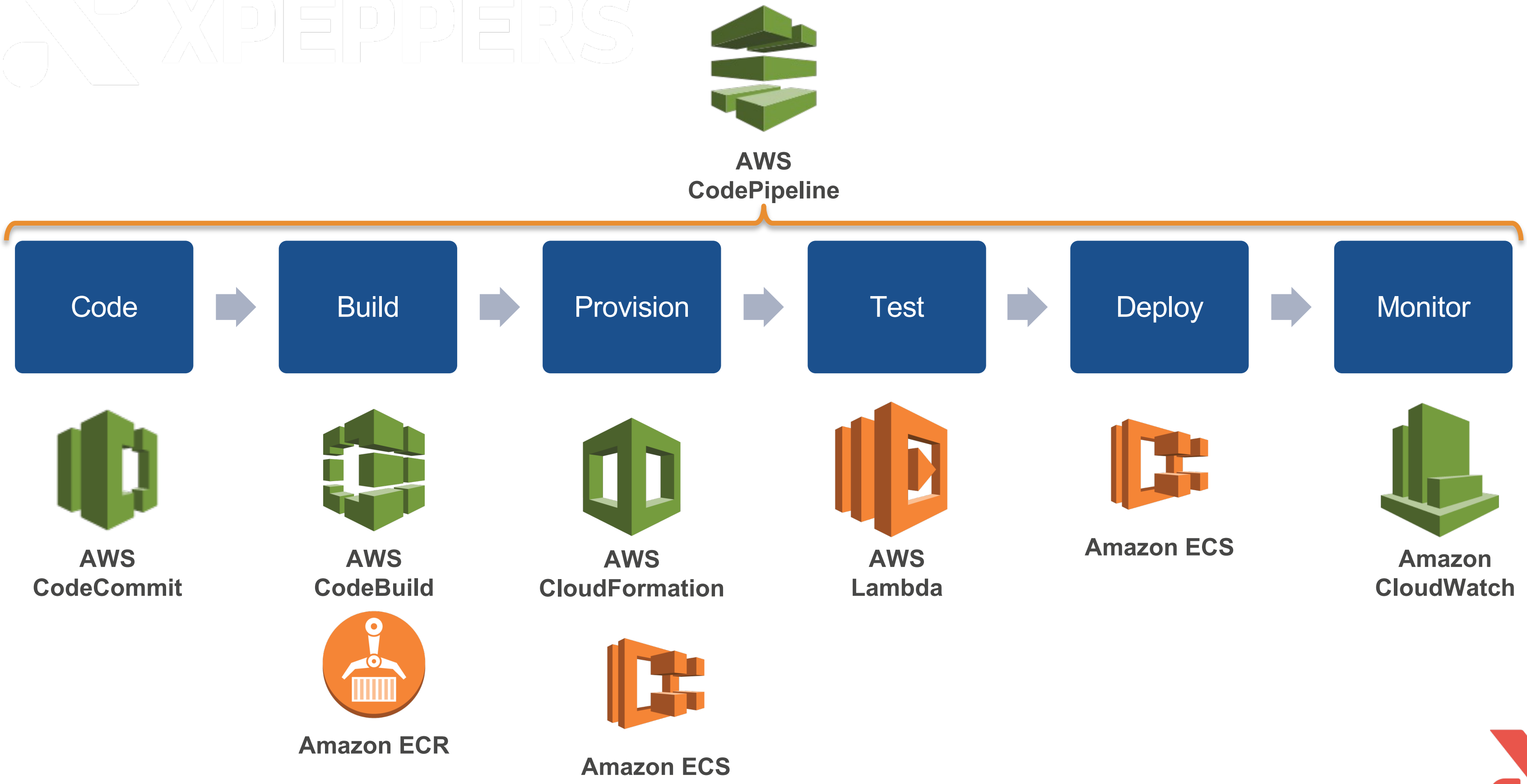
1. Create new task definition
2. - Create new service
3. - Attach new service to ALB
4. - Scale Up Green Service
5. - Scale Down Blue Service

Canary

Service Update

1. Create new task definition
2. Update the service

Production - CI/CD with ECS and CodePipeline



\ Demo - Steps

1. Create ECS Cluster and related resources
2. Creates an ECS task definition from your compose file
3. Create Service from task definition and attach to ALB
4. Scale task associated to services
5. Scale Cluster instances
6. Simulate Blue/Green deployment

\ Links

- https://martinfowler.com/bliki/MonolithFirst.html
- https://docs.docker.com/docker-for-aws/
- http://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_CLI_reference.html
- http://docs.aws.amazon.com/cli/latest/reference/ecs/index.html#cli-aws-ecs
- https://github.com/ExpediaDotCom/c3vis

XPEPPERS

www.xpeppers.com

@xpeppers /xpepperssrl

@LatellaPaolo