**Content**

[1. Introduction 2](#_Toc68524601)

[2. ECS 3](#_Toc68524602)

[2.1.1. ECS – Task definition 3](#_Toc68524603)

[2.1.2. ECS – Service definition 3](#_Toc68524604)

[3. Fargate 5](#_Toc68524605)

[4. Elastic Beanstalk 6](#_Toc68524606)

[4.1.1. Dockerrun.aws.json 6](#_Toc68524607)

[5. EKS 9](#_Toc68524608)

1. Introduction

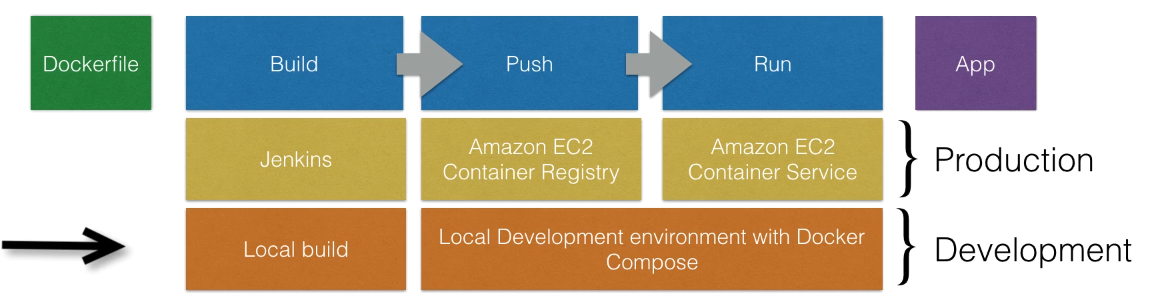
To manage containers we need a container management platform:

* ECS – Amazon’s own platform
* Fargate: Amazon’s own serverless platform
* EKS – Amazons’ managed Kuernetes (open source)

Just like **packer builds AMIs,** you can use docker to build docker images.

Those images can then be run on any Linux host with Docker Engine.

**Docker Engine installed in the Vagrant DevOps box?**



Docker build ca be executed manually or by Jenkins.

You can run the docker app by executing “docker run” locally. Instead we can push it to Amazon and run this app on AWS.

* Push this locally build image to Amazon ECR – where docker images can be stored in.
* Set up a docker cluster (ECS) to run our Docker applications.
* [Optional] the creation of the registry can be done using terraform.

(terraform to create aws resource, build, login to the ecr, push using the url , so easy )

1. ECS

*The cluster is just a group of EC2 instances with this agent installed on.*

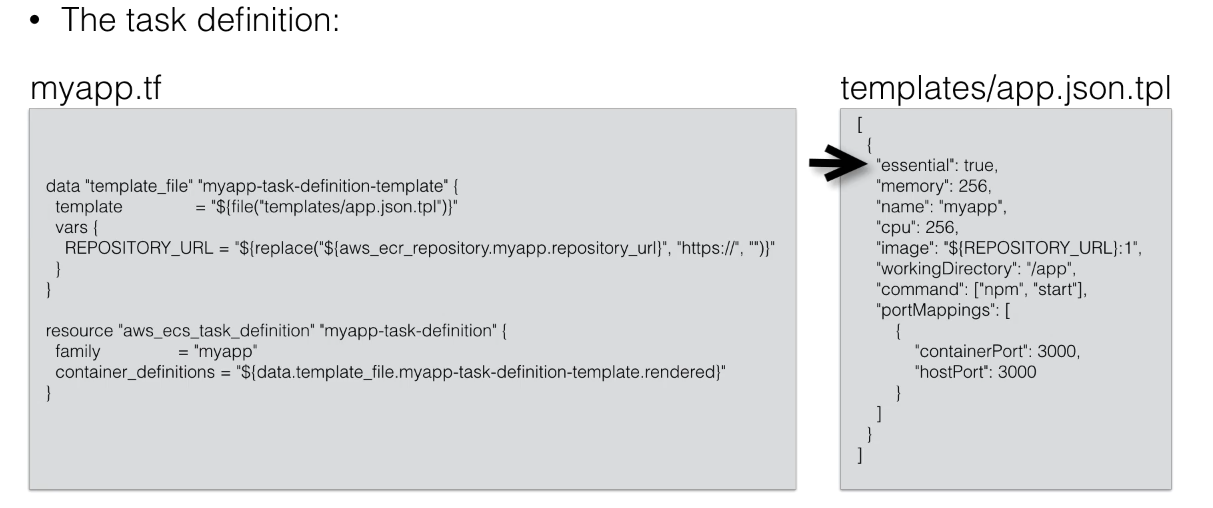
You need to start an autoscaling group with a custom AMI. The custom AMI contains the ECS agent. Once the ECS cluster is online, tasks and services can be started on the cluster.

* + 1. ECS – Task definition

Before docker app can be launched, a task definition needs to be provided. The task definition describes what docker container to be run on the cluster. JSON form, to tell EC2 how to run the container (it does not runs it itself).

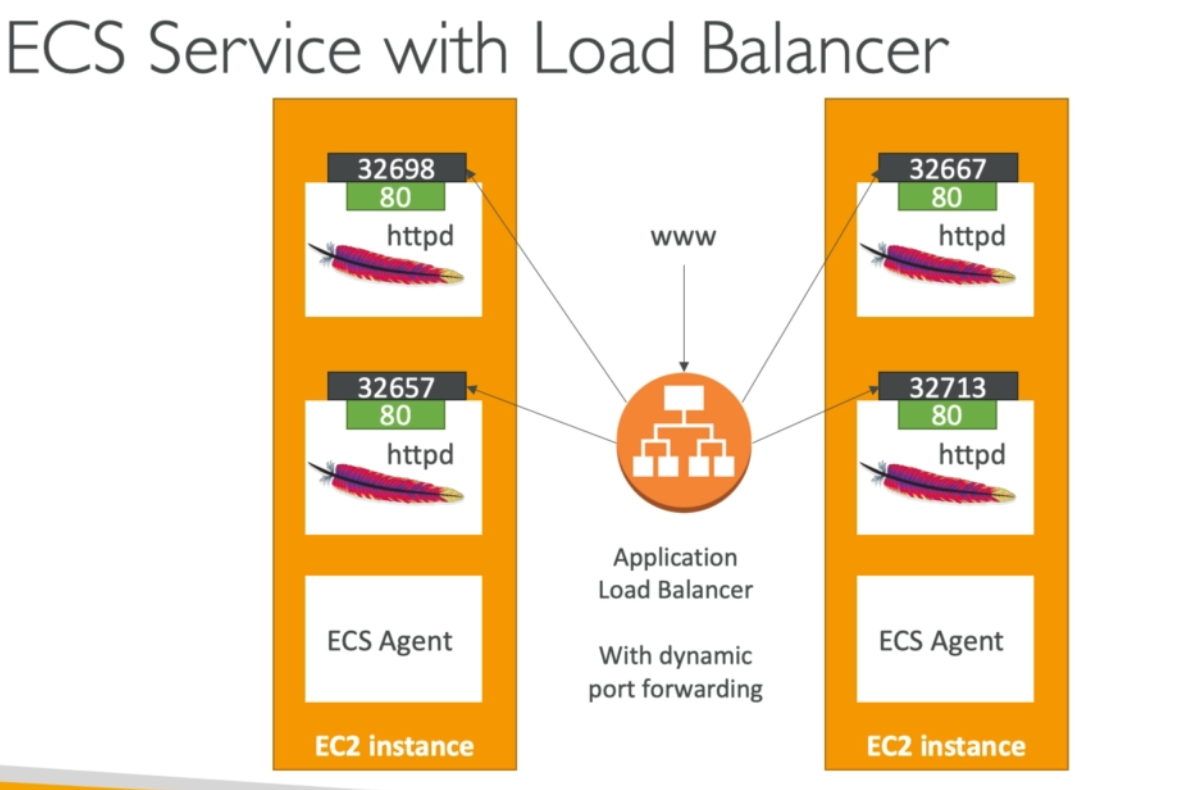
* Specifies Docker image (the docker image in ECR)
* Max CPU usage, max memory usage
* Wether containers should be linked (e.g. link app container w DB container)
* Env vars. (credentials)
* Port binding
* Networking information
* Any other container specific info

The task definition needs an IAM role. So if it cannot do something first thing is to check the permissions.

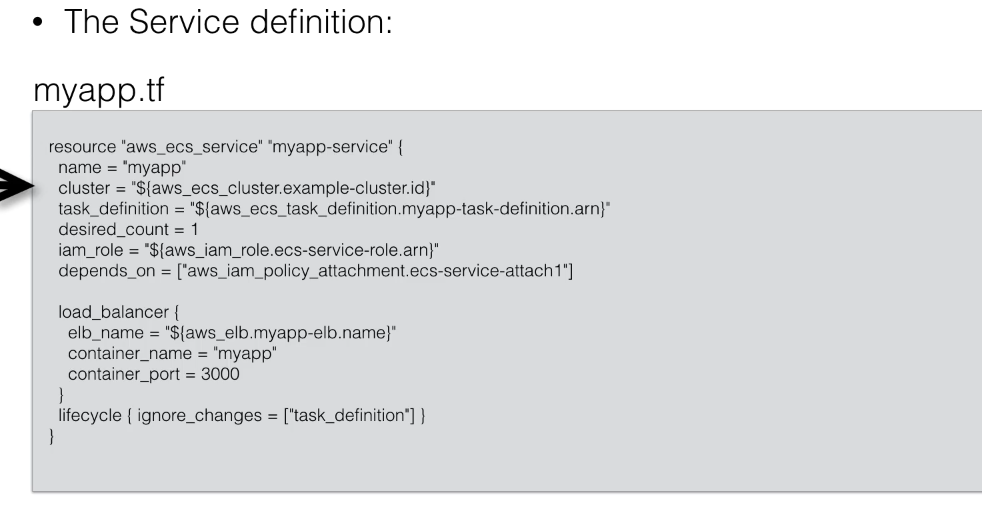


* + 1. ECS – Service definition

It is going to run a specific amount of containers based on the task definition. You can scale. A service is always running, if the container stops it will be restarted, can be scaled, you can run 1 instance of a container or multiple. You can put an ELB in front of a service. Typically run multiple instances of a container, spread over AZs. If one container fails, your LBs stops sending traffic to it. Running multiple instances w an ELB / ALB allows you to have HA.



If we do not set static mapping, aws will generate random port to map the task. Load Balancer is able to figure out these random ports and distribute traffic using dynamic port forwarding.



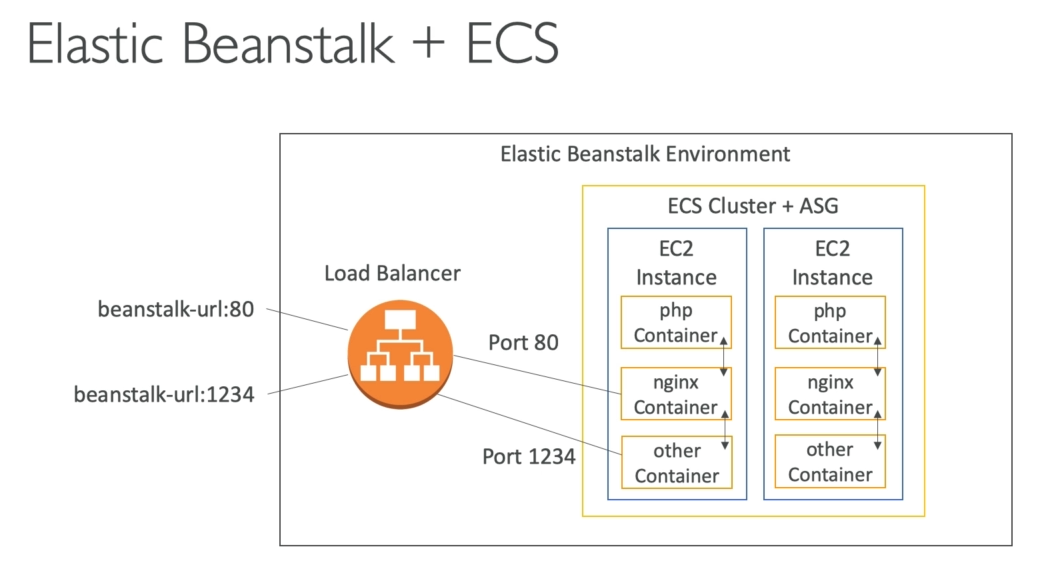
1. Fargate
2. Elastic Beanstalk

Difference between single container deployment and multi-container. If there are e.g. 3 packages and all of them has its own dockerfile then EB will need a little bit of help.

The multi container mode will create for you the:

* ECS Cluster
* EC2 instances, configured to use the ECS Cluster
* Load Balancers (in high availability mode)
* Task definitions and executions

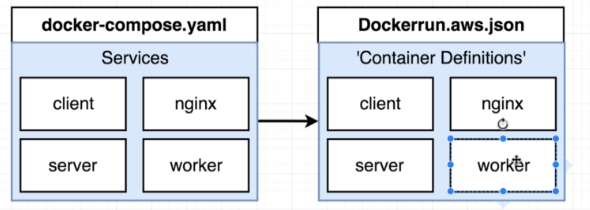
Docker images must be pre-built (stored in ECR for example) and Dockerrun.aws.json must be provided at the root of the source code.



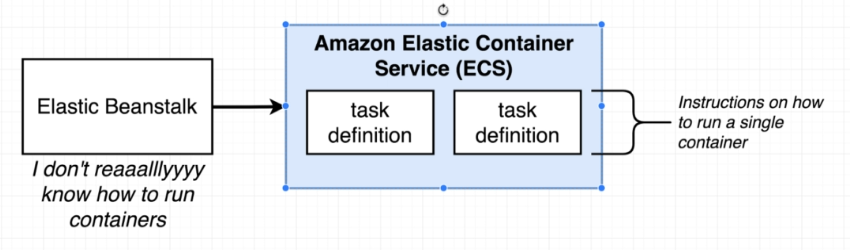
* + 1. Dockerrun.aws.json

That file is going to tell EB exactly where to pull all of our images from, what resources to allocate each one, how to setup some port mappings and some associated information. It will remind you how docker-compose.yml is setup.

Docker-compose.yml is primarly meant to use in development, environments, and you can kind of think of it as a single file that encodes a lot of directions that would normally be passed to docker run.



The biggest difference that in docker-compose.yml we define how to build the image but in Dockerrun.aws.json we already have the image, just need to pull it from DockerHub using an ID.







Tip: use json validator (jsonlint.com? )s

1. EKS