AWS Fargate

Never stand in row again!

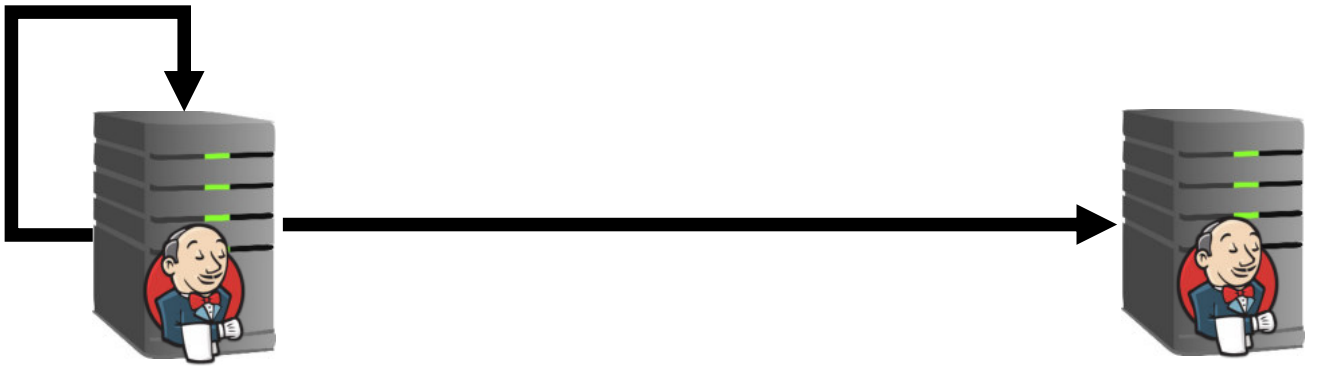
With Jenkins on AWS ECS and Fargate

Name: Philipp Koch
Company: Senacor Technologies AG
Position: Technical Leader
Email: Philipp.Koch@senacor.com
Twitter: phienor1

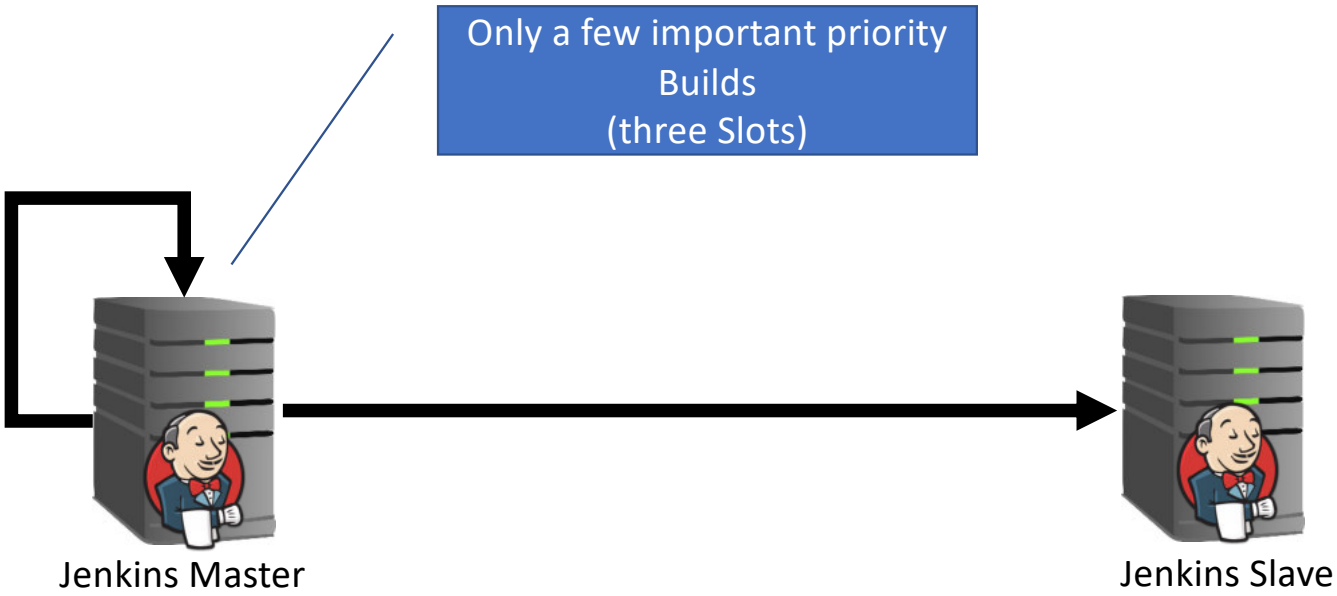
Interests:

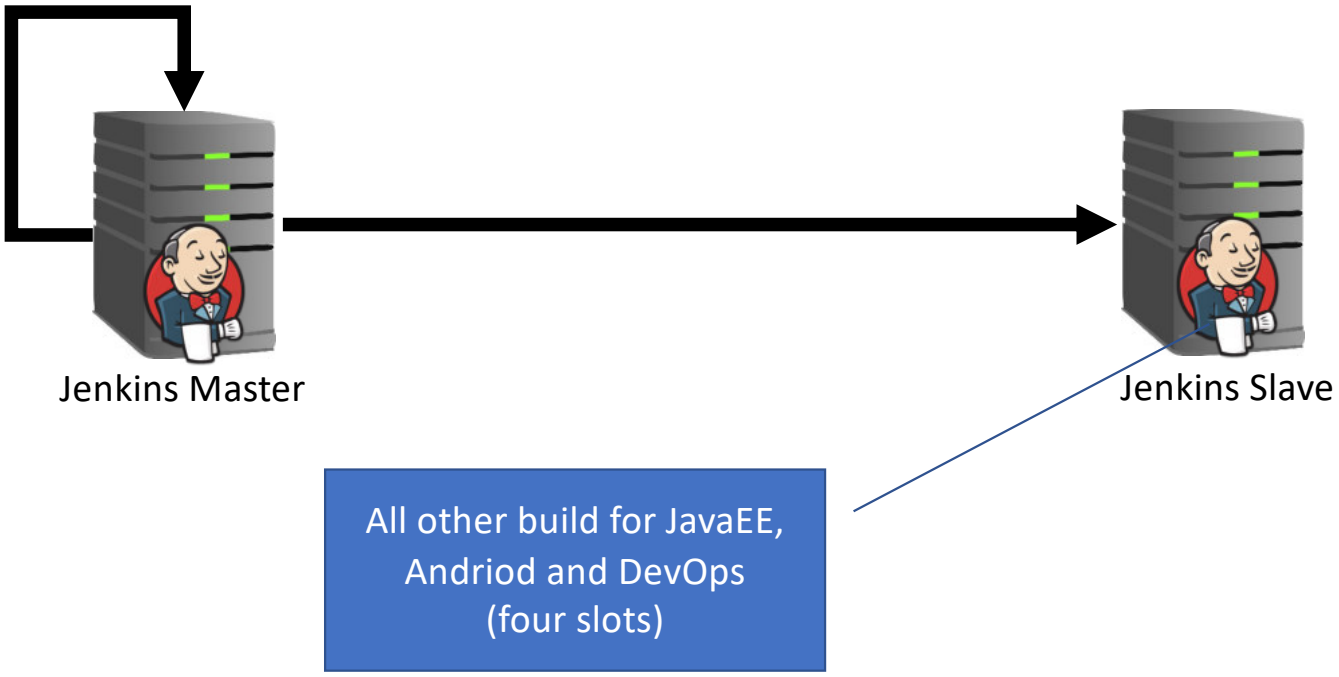
- Java
- Agile development
- DevOps
- Cloud technologies


Me waiting for my build to complete



the original setup


Only a few important priority Builds
(three Slots)
Jenkins Master
Jenkins Slave


Jenkins Master
Jenkins Slave
All other build for JavaEE,
Andriod and DevOps
(four slots)

Pull Request PR-142

Full project name: datamodel-project-build/parcelos-data-model-scripts/PR-142

Recent Changes

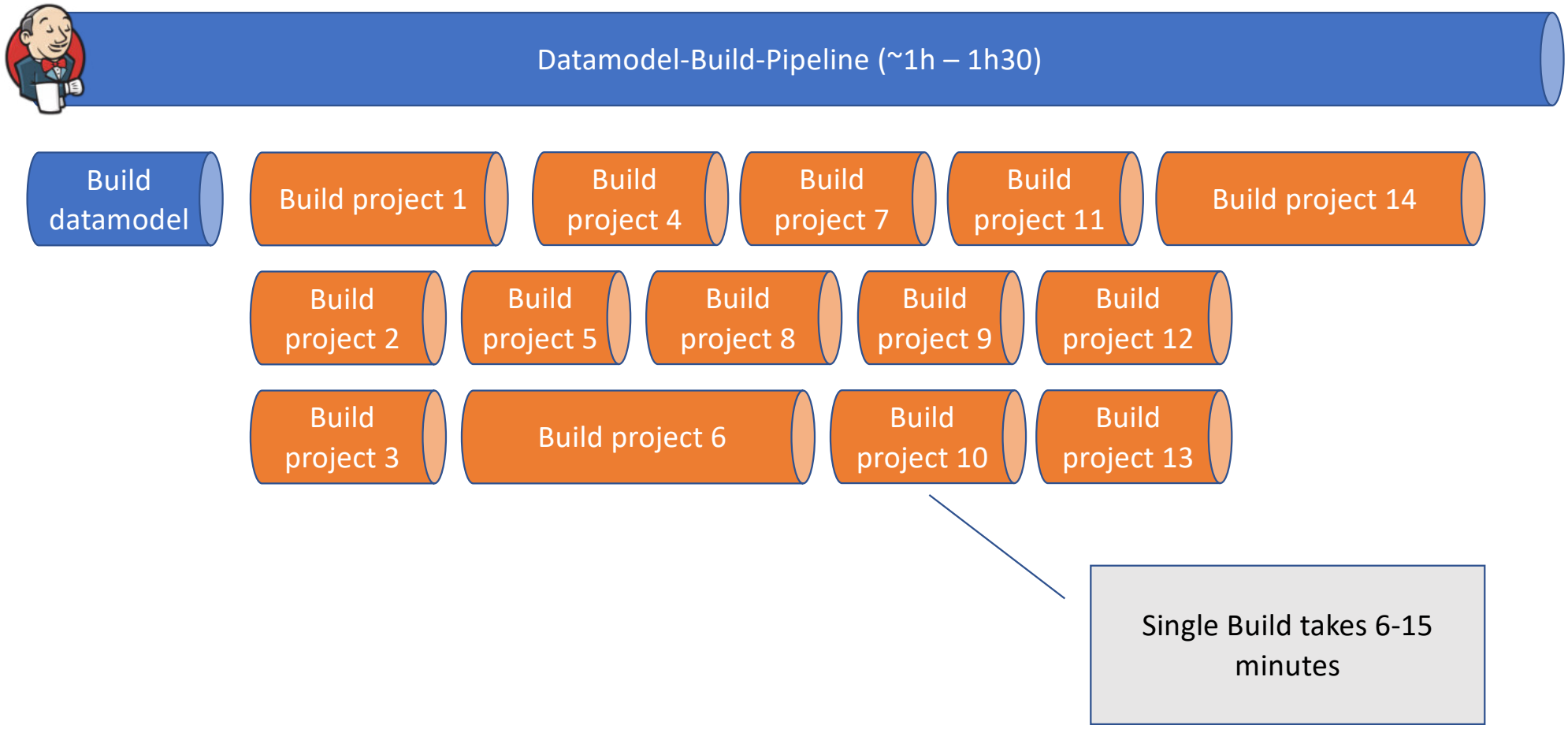
Stage View

	prepare environment	checkout dependencies project	build parcelos-data-model-scripts	subsequent build parcelos-subsystem-seedmasterdata	subsequent build parcelos-fwk	subsequent build parcelos-fwk-configuration	subsequent build parcelos-subsystem-partner	subsequent build parcelos-legacy-breach	subsequent build parcelos-subsystem-order	subsequent build parcelos-subsystem-parcel	subsequent build parcelos-subsystem-transport	subsequent build parcelos-parcel-flow	subsequent build parcelos-transport-flow	subsequent build parcelos-order-flow	subsequent build parcelos-workflow-flow	subsequent build parcelos-subsystem-user	subsequent build parcelos-partner-flow
Average stage times:	7s	28s	1min 49s	5min 44s	3min 55s	7min 11s	8min 29s	11min 42s	8min 49s	14min 8s	24min 39s	9min 32s	14min 24s	7min 51s	7min 26s	7min 56s	8min 36s
#2 Nov 19 11:30 5 commits	11s	25s	1min 57s	1min 35s failed	3min 44s	6min 53s	8min 17s	11min 28s failed	8min 11s	13min 24s	23min 52s	8min 19s	13min 39s failed	7min 32s	7min 11s	7min 58s	8min 31s
#1 Nov 19 07:34 No Changes	3s	30s	1min 41s	9min 52s failed	4min 7s failed	7min 29s	8min 41s	11min 56s failed	9min 27s	14min 52s	25min 26s	10min 45s	15min 9s failed	8min 10s	7min 42s	7min 53s	8min 40s

Permalinks

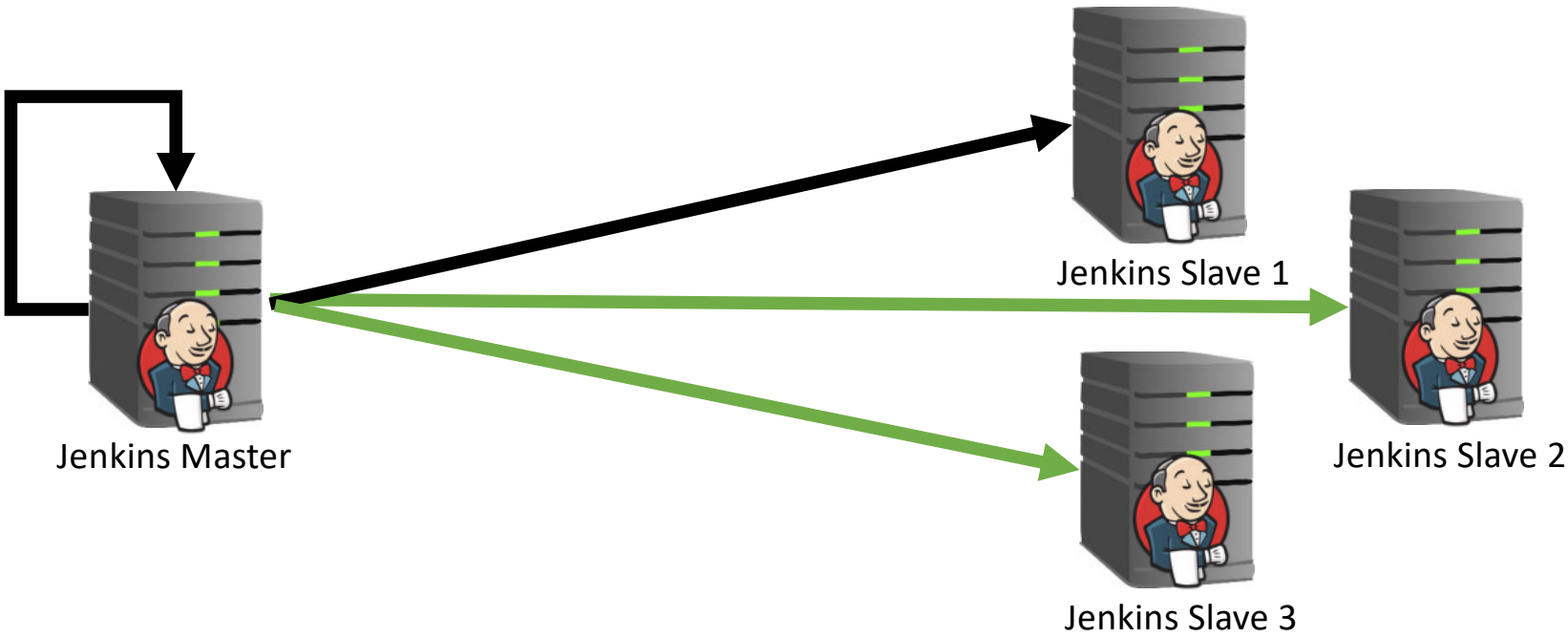
- Last build (#2), 1 day 9 hr ago
- Last failed build (#2), 1 day 9 hr ago
- Last unsuccessful build (#2), 1 day 9 hr ago
- Last completed build (#2), 1 day 9 hr ago

Some special pipelines produces a lot of traffic

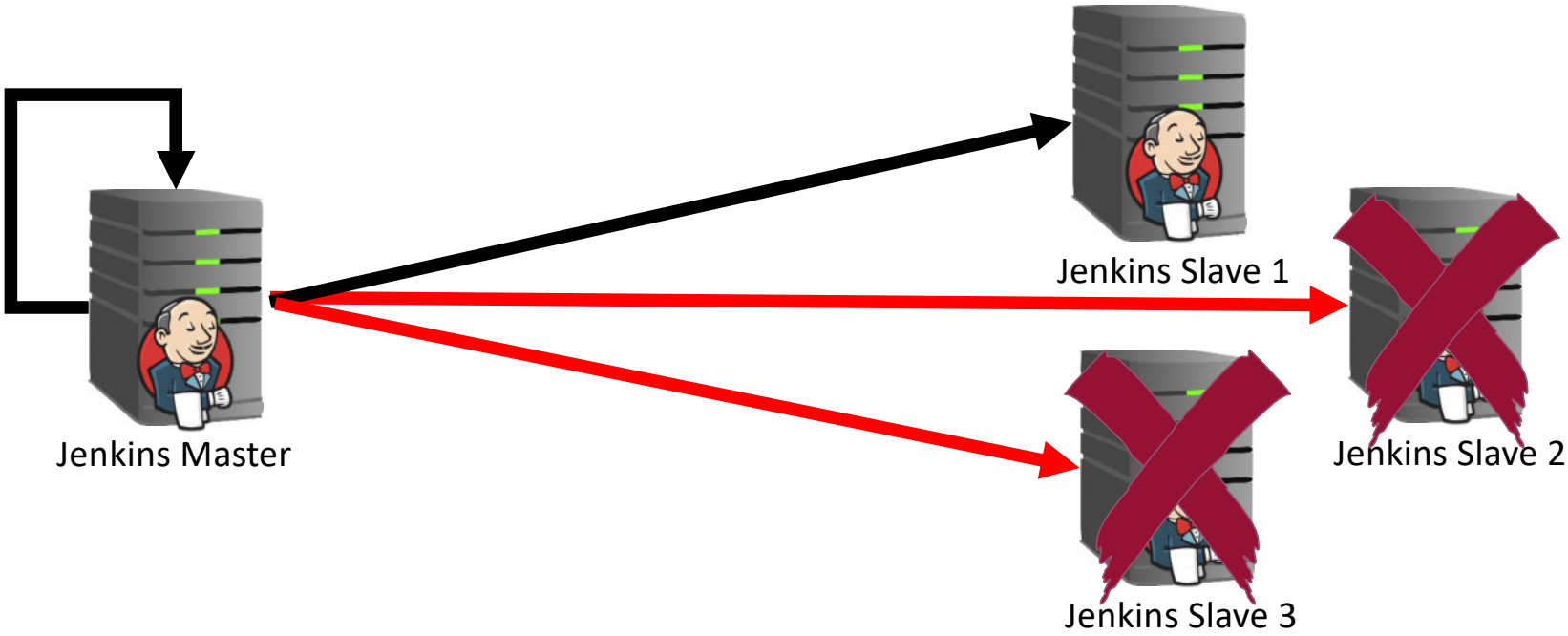

Datamodel-Build-Pipeline (~1h – 1h30)
Build datamodel
Build project 1
Build project 4
Build project 7
Build project 11
Build project 14
Build project 2
Build project 5
Build project 8
Build project 9
Build project 12
Build project 3
Build project 6
Build project 10
Build project 13
Single Build takes 6-15 minutes


Too many builds for too few slots…


... leads to stress and anger at the developer

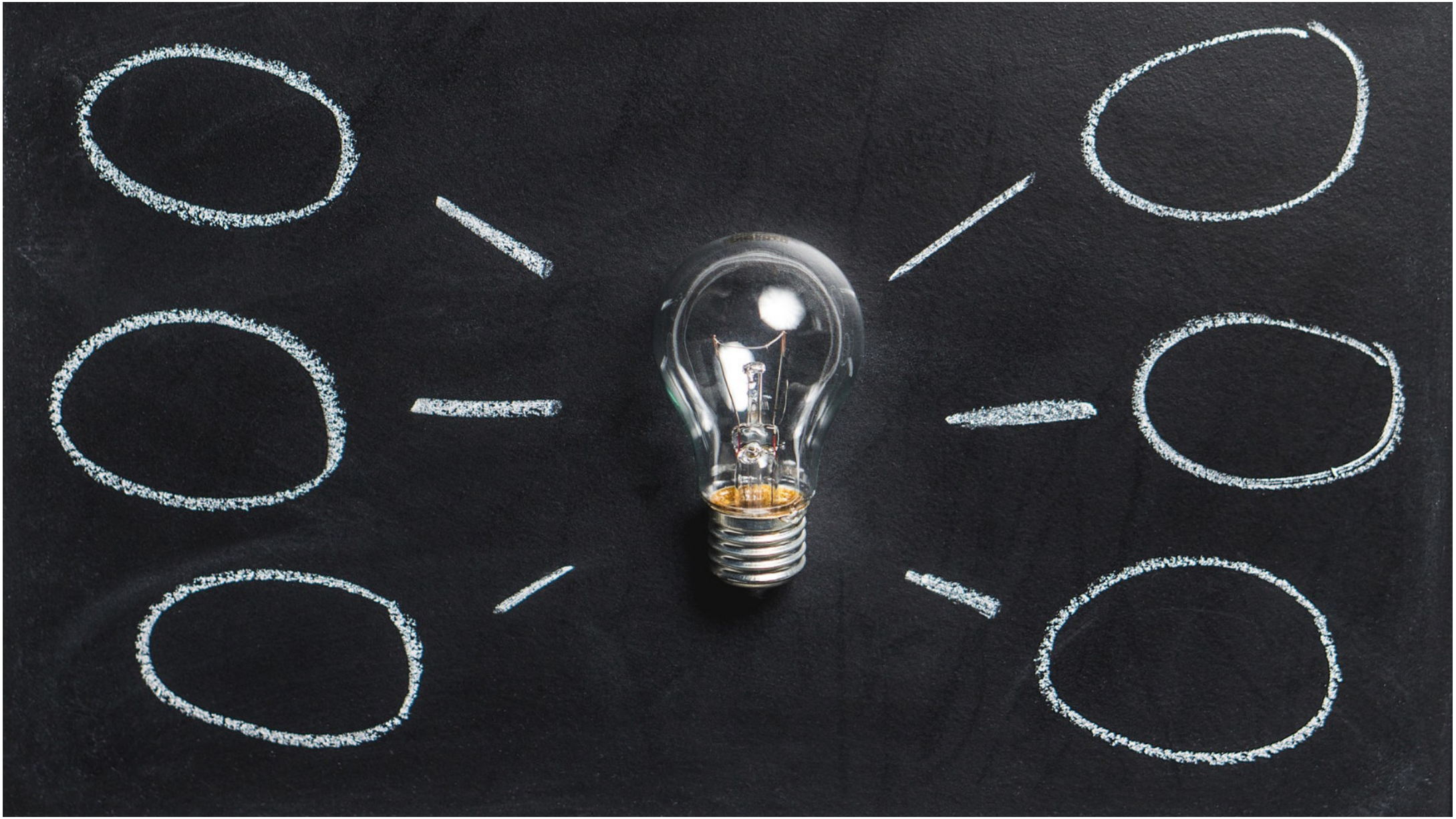


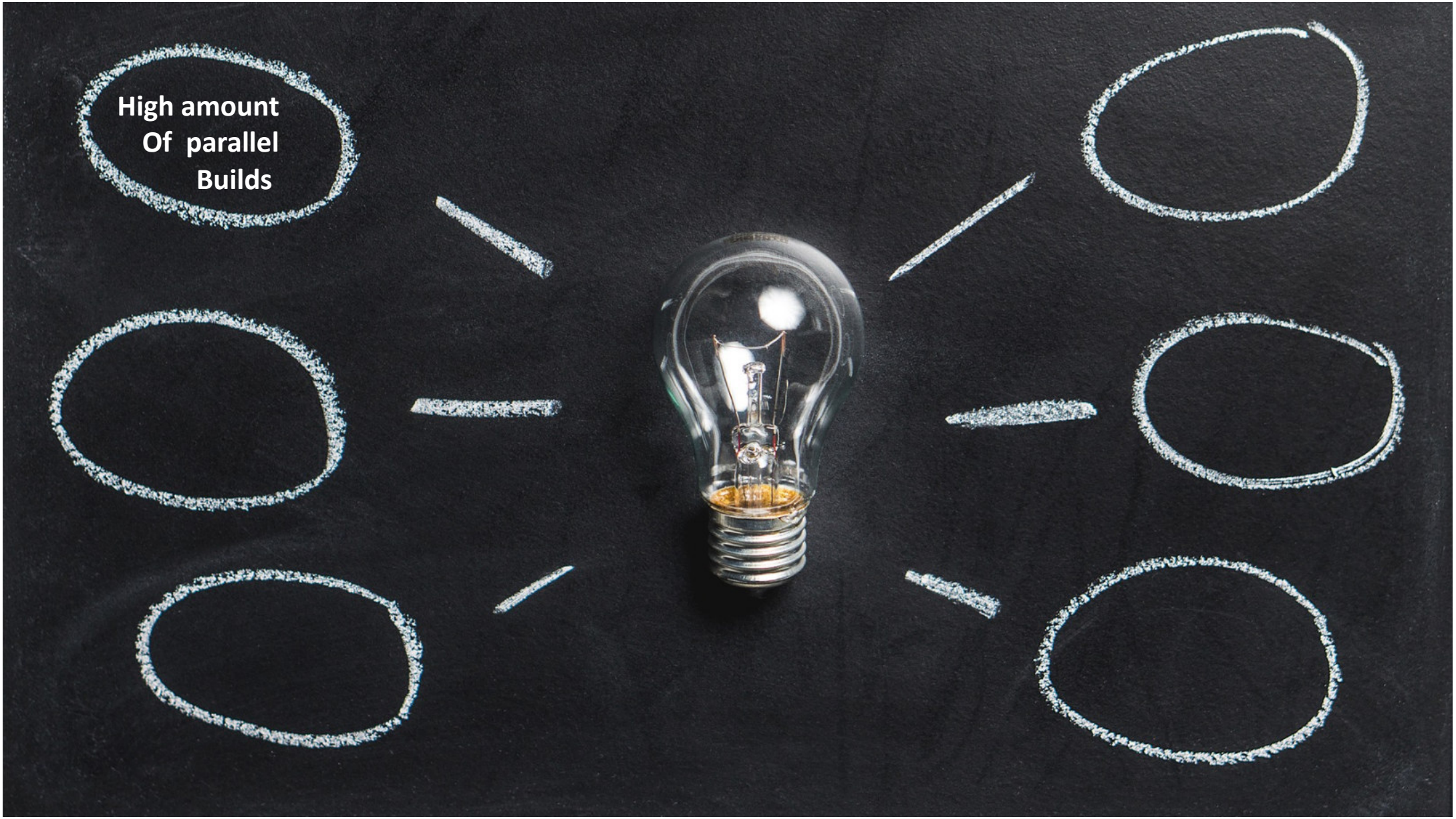
Quick win: more of the same

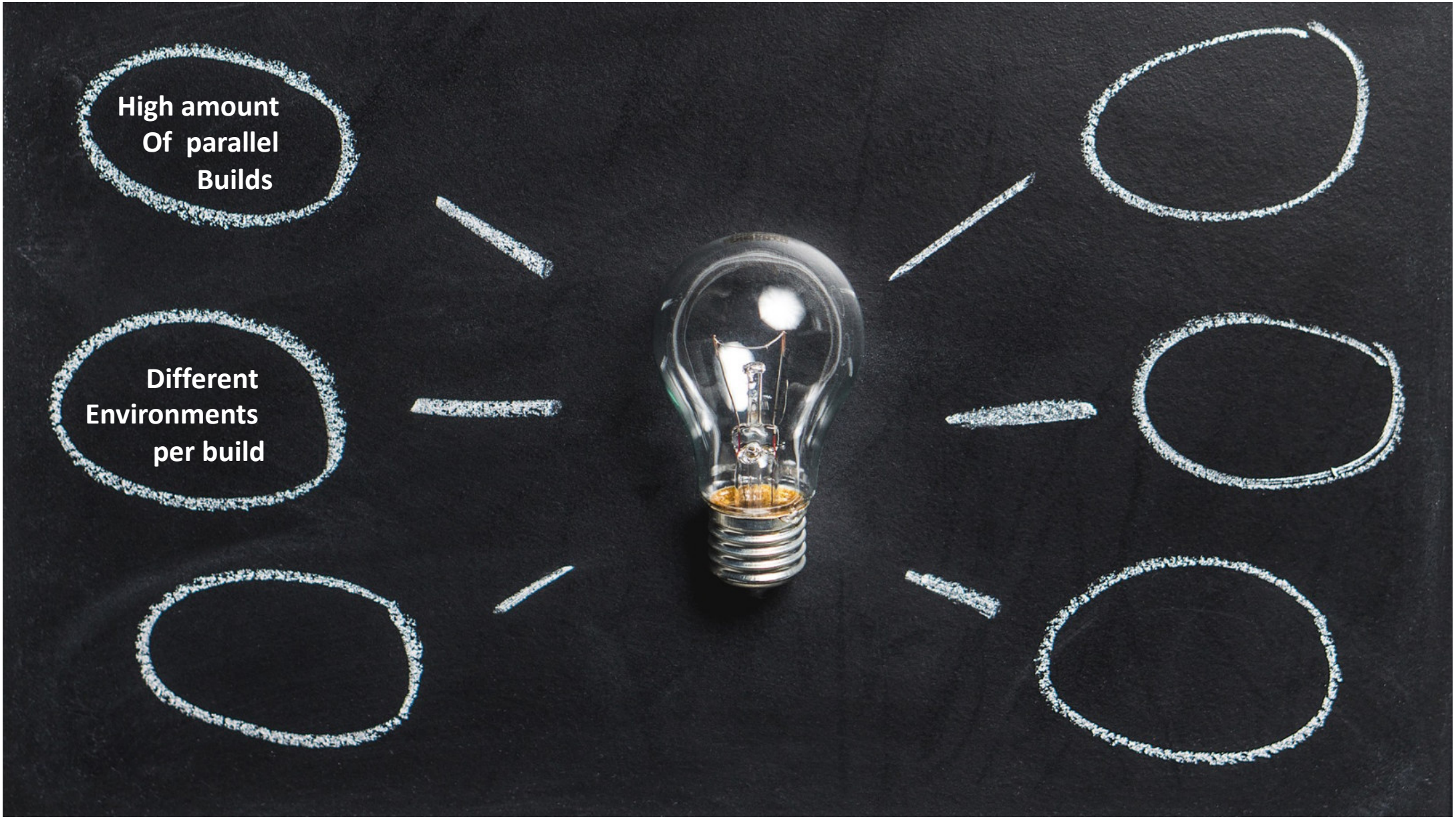


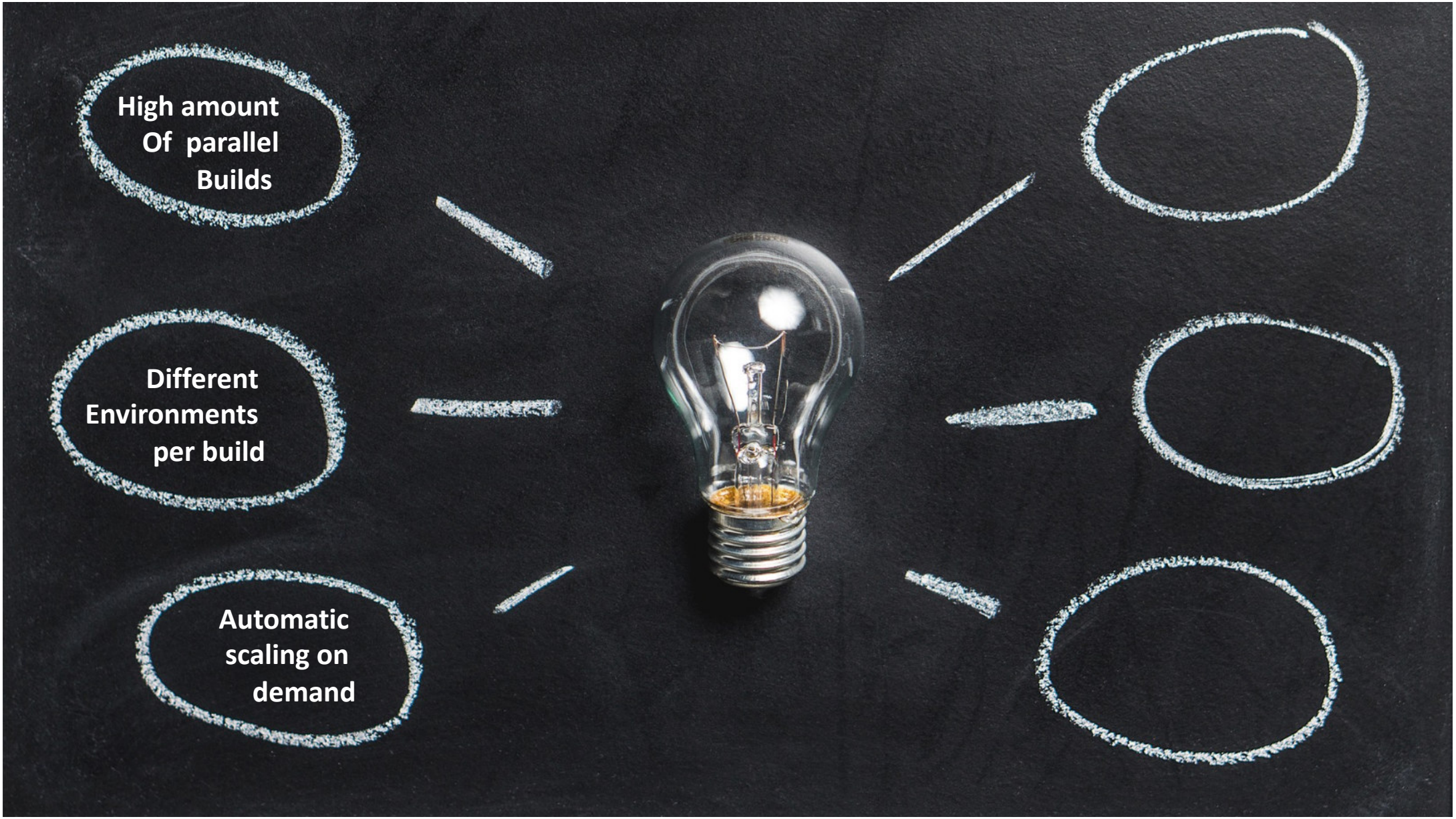
Has not been so easy for internal reasons

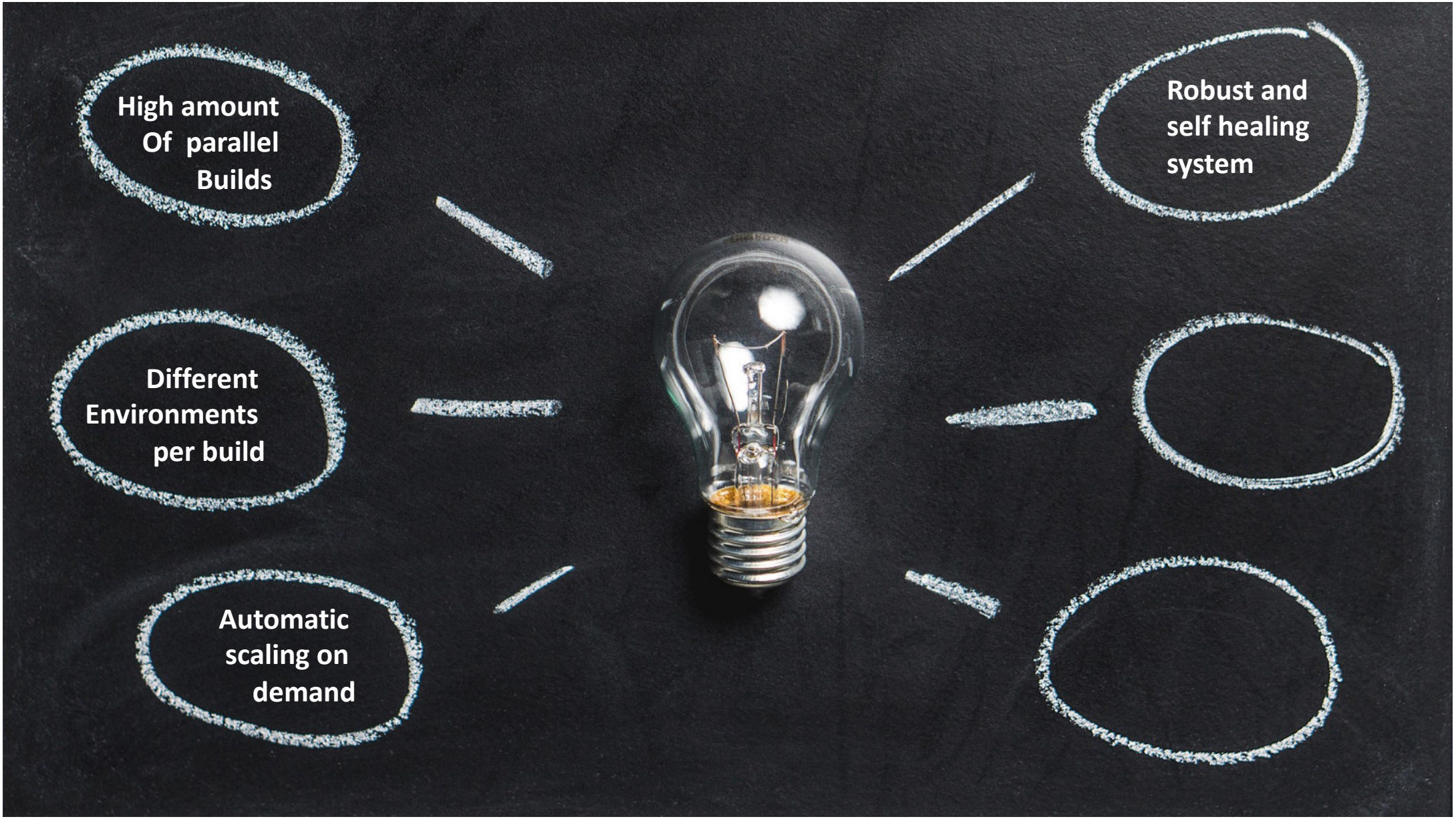

Step back and plan!

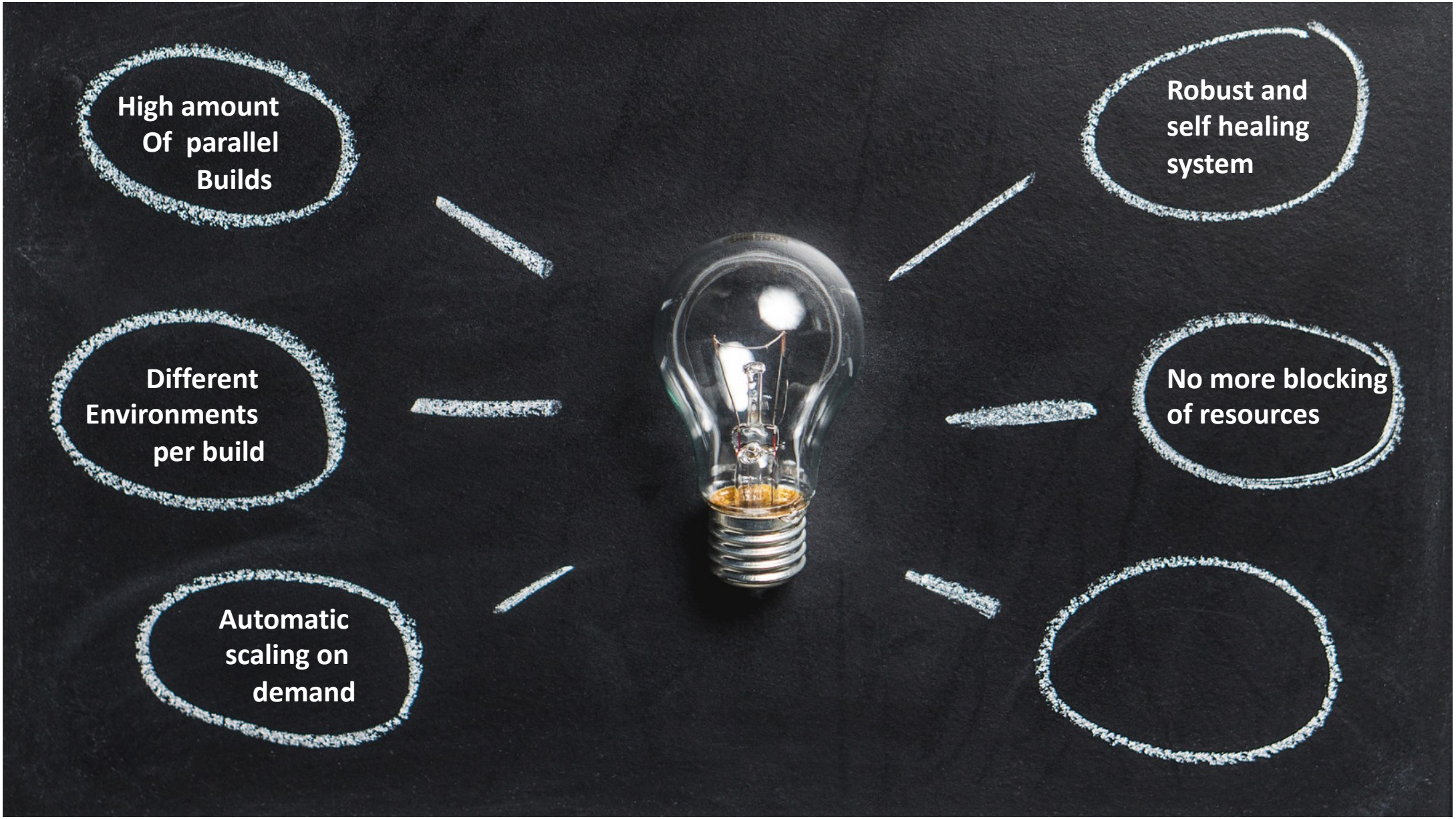


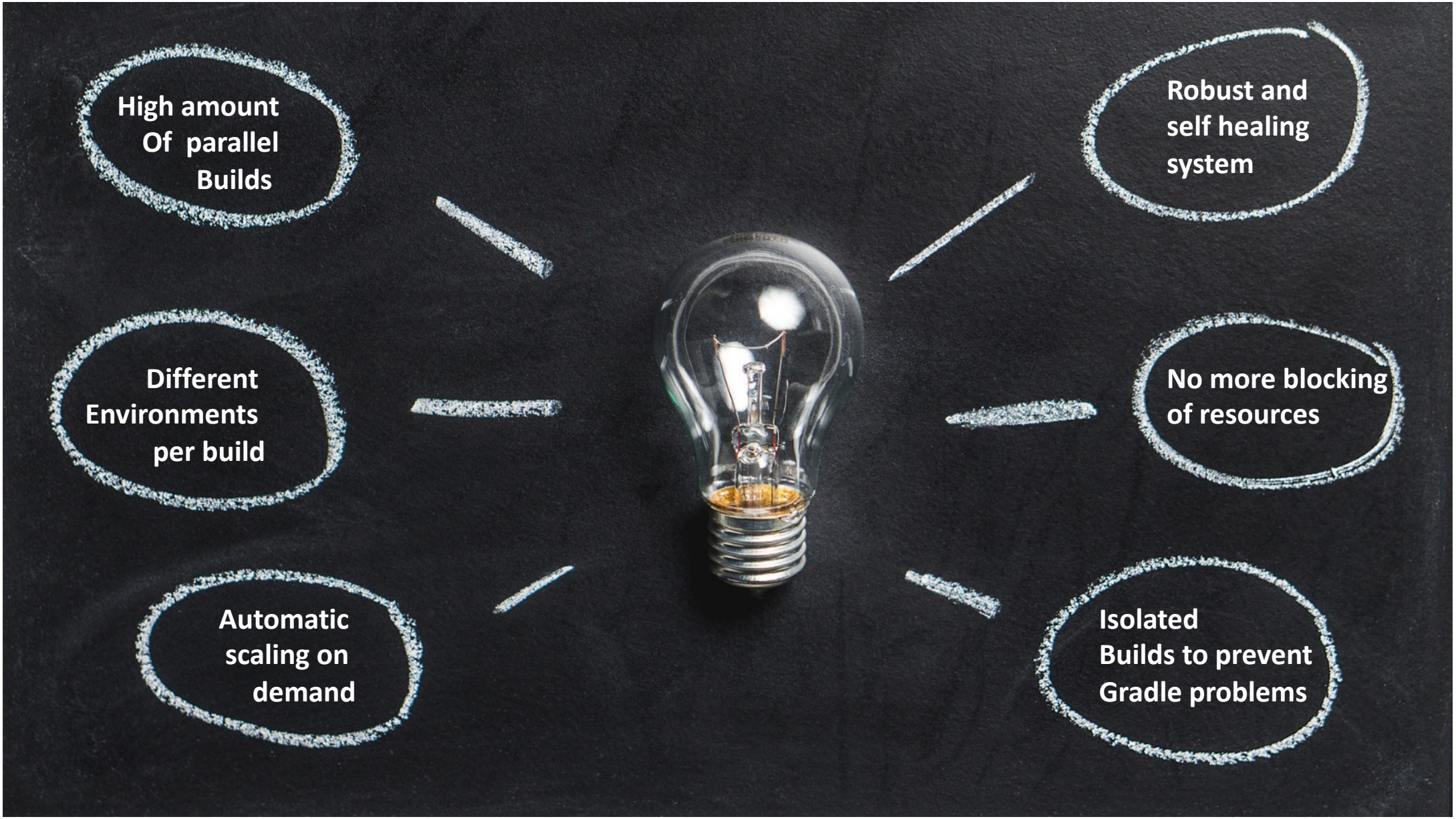

High amount
Of parallel
Builds


High amount
Of parallel
Builds
Different
Environments
per build

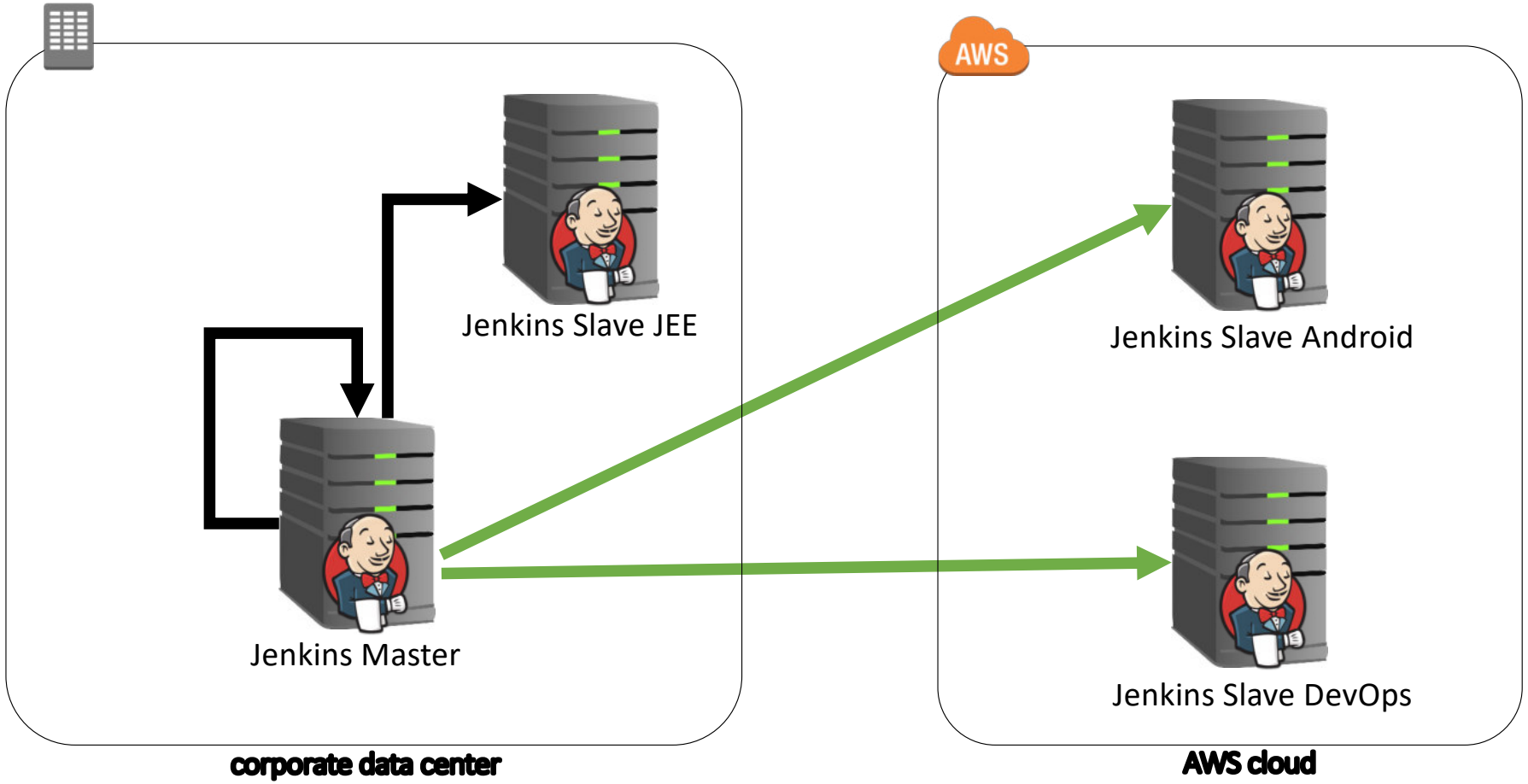

High amount
Of parallel
Builds
Different
Environments
per build
Automatic
scaling on
demand


High amount Of parallel Builds
Robust and self healing system
Different Environments per build
Automatic scaling on demand

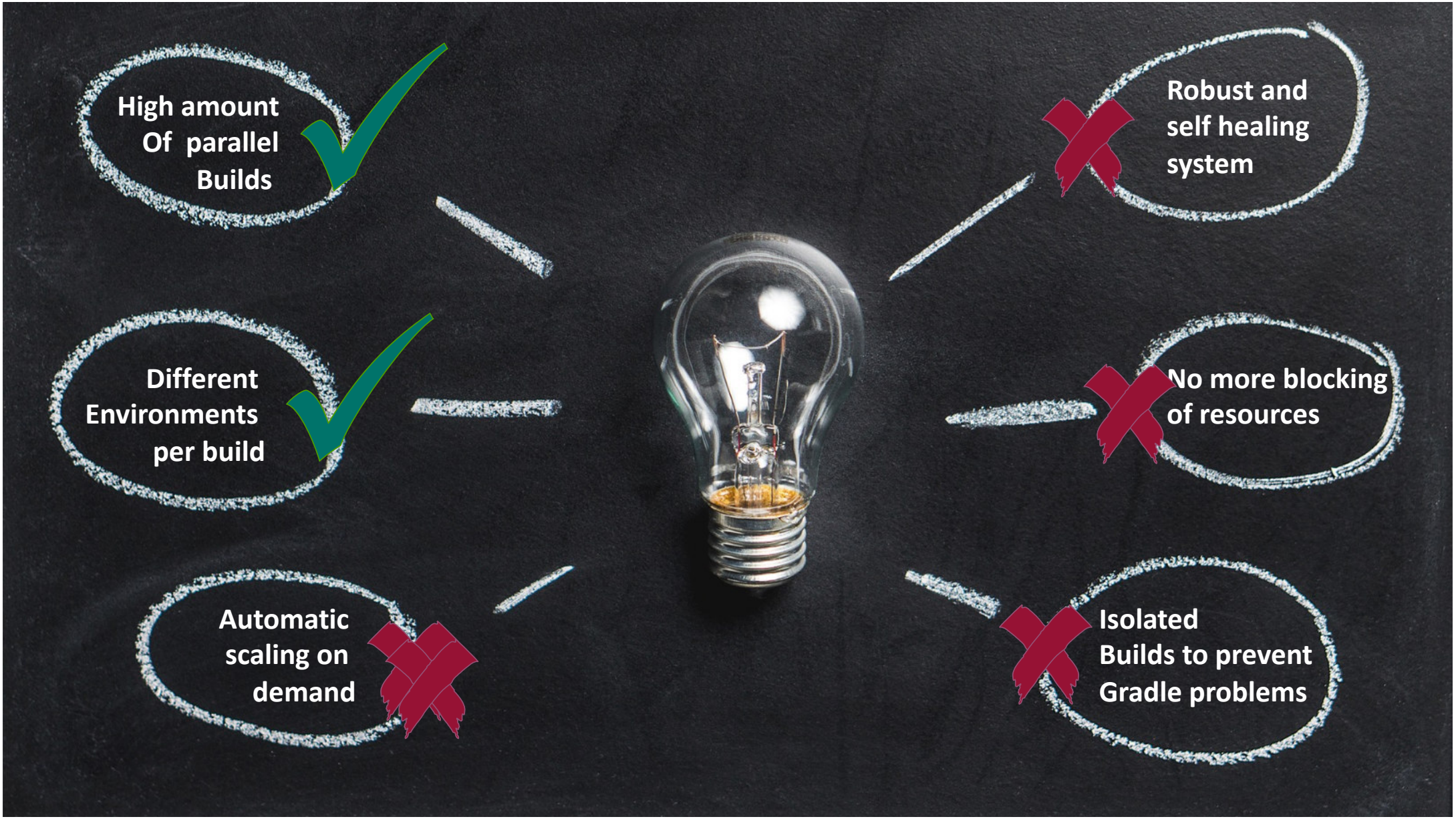

High amount Of parallel Builds
Robust and self healing system
Different Environments per build
No more blocking of resources
Automatic scaling on demand

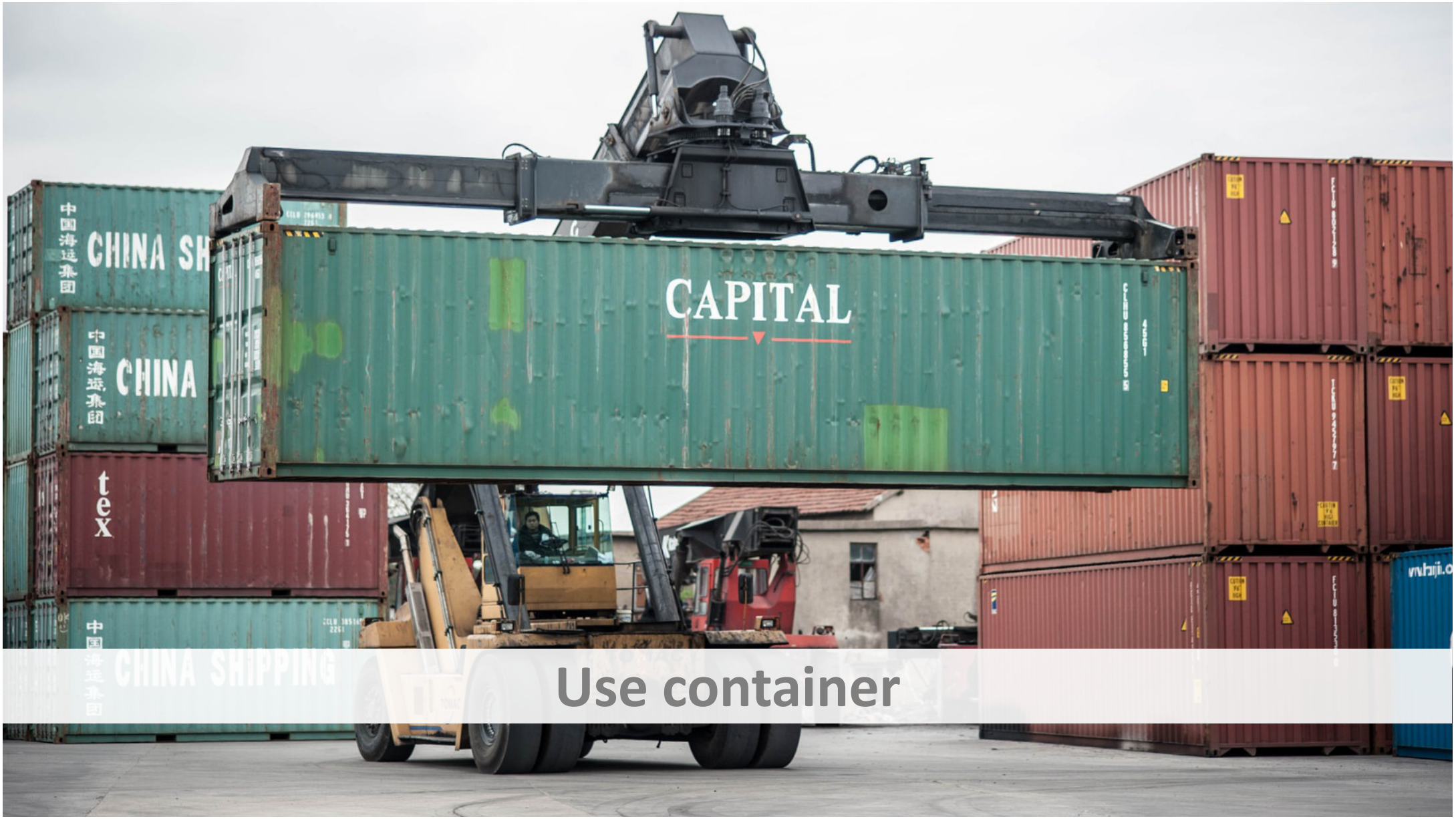

High amount Of parallel Builds
Robust and self healing system
Different Environments per build
No more blocking of resources
Automatic scaling on demand
Isolated Builds to prevent Gradle problems

Why not use our AWS Environment



Quick win(2): more of the same (But in AWS)


High amount
Of parallel
Builds
Different
Environments
per build
Automatic
scaling on
demand
Robust and
self healing
system
No more blocking
of resources
Isolated
Builds to prevent
Gradle problems



Use container

AWS Fargate

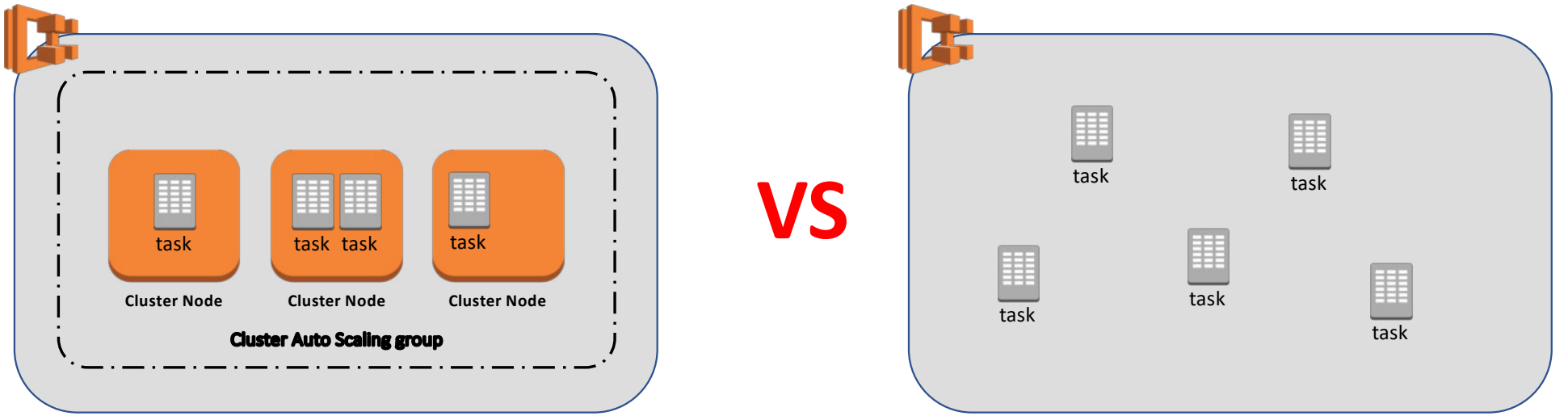
Cluster-as-a-Service

Serverless

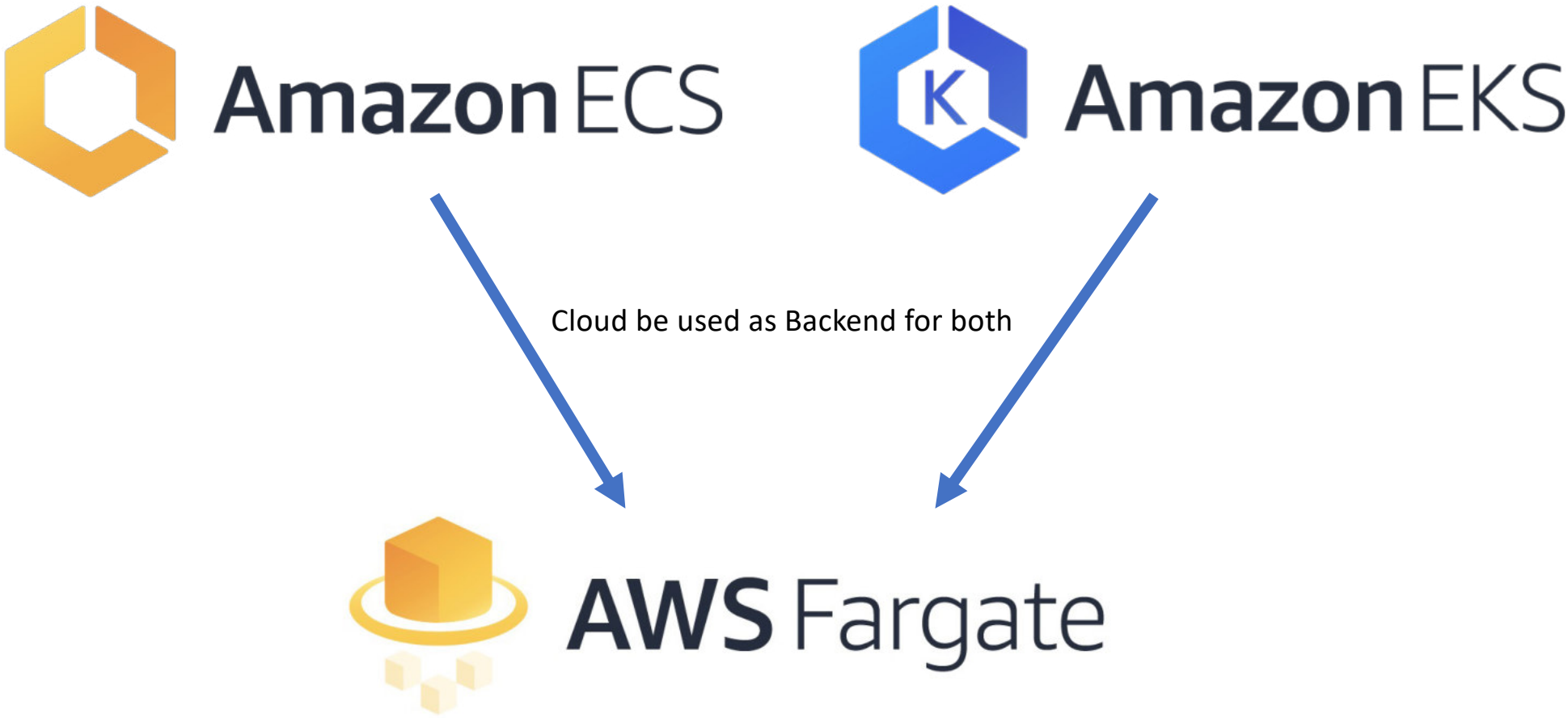
Container Orchestration

AWS Fargate

Cluster without operation

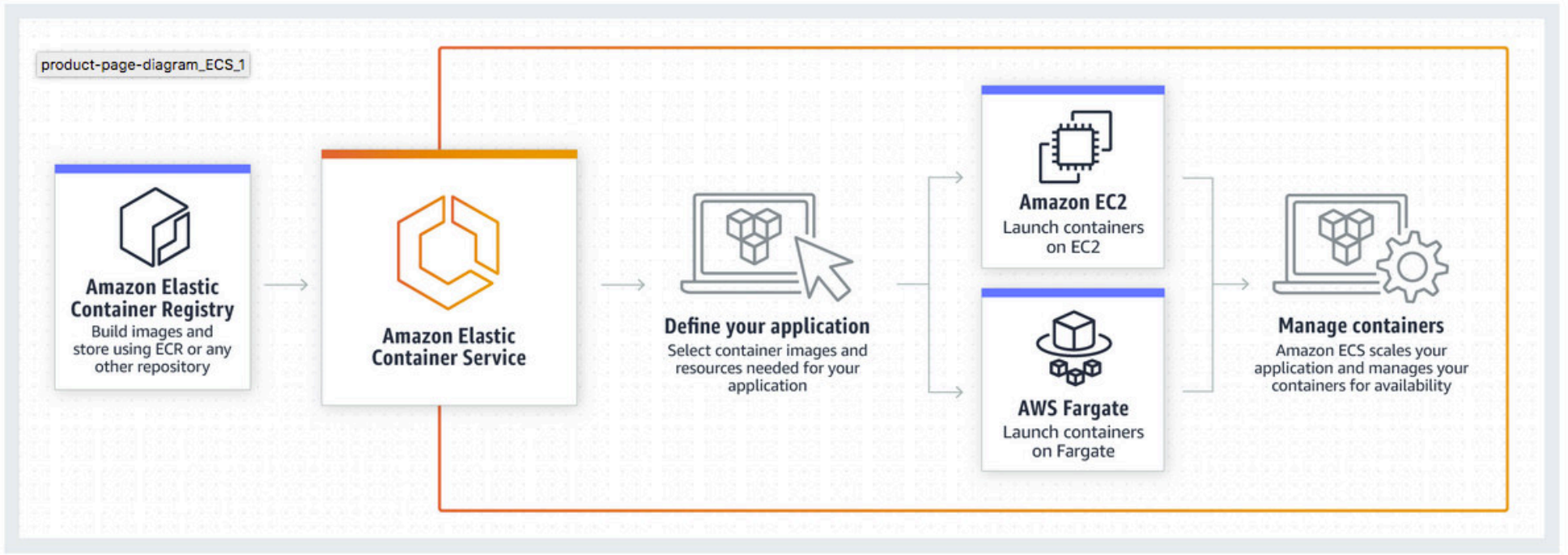


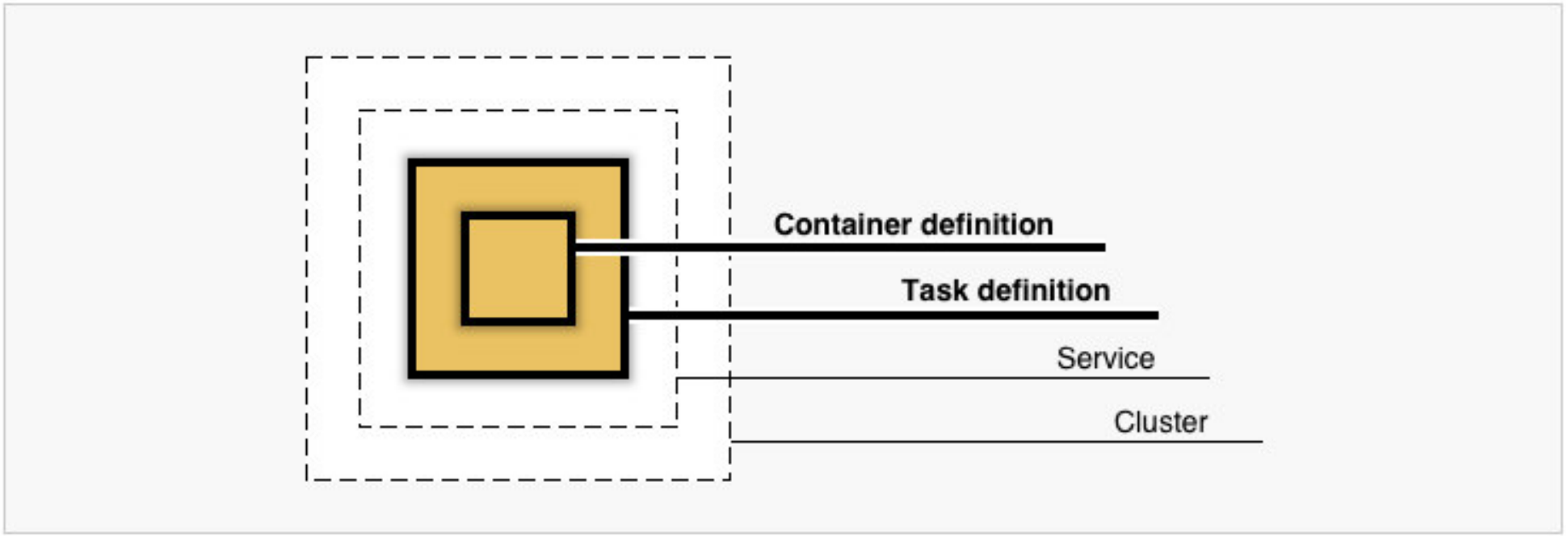
Only pay what you use


Amazon ECS
Amazon EKS
Cloud be used as Backend for both
AWS Fargate

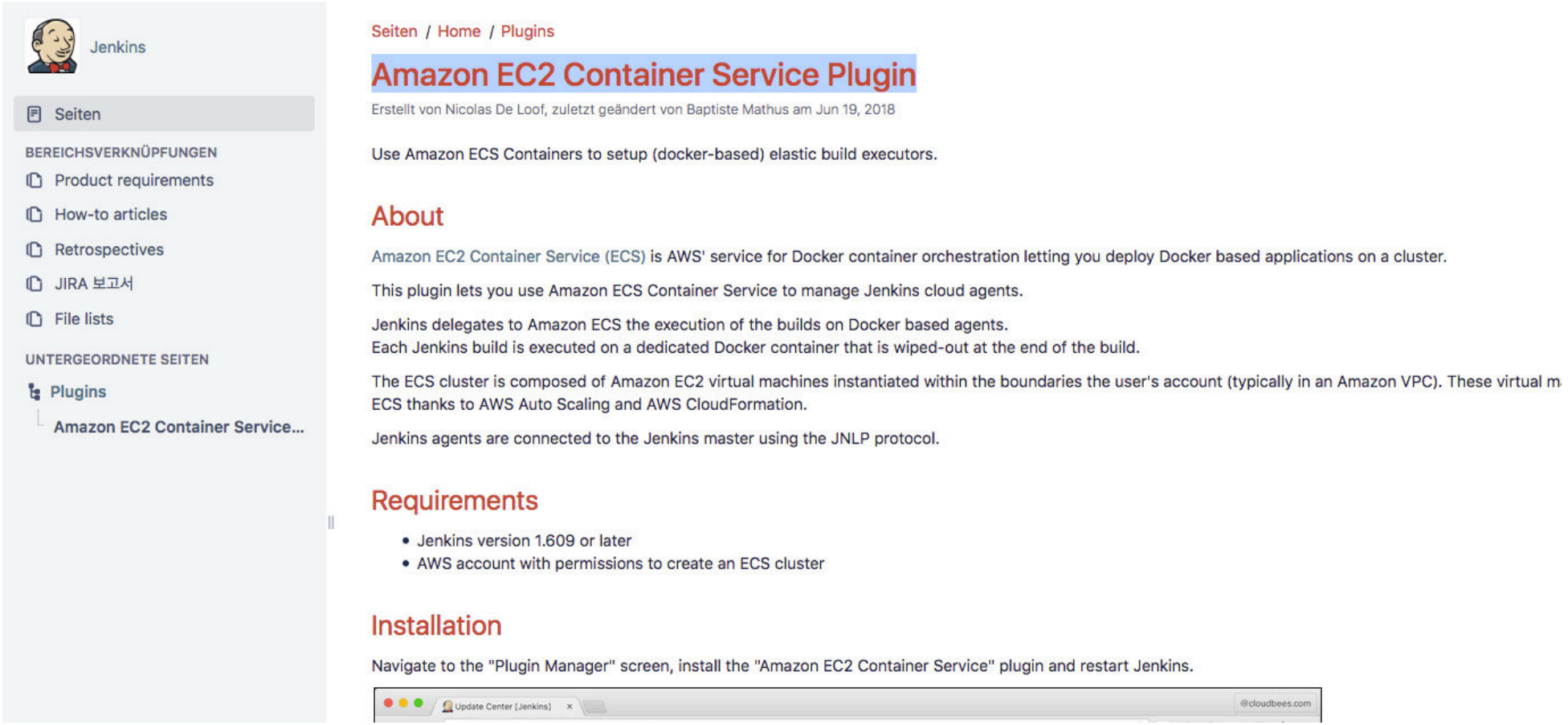

Amazon ECS
AWS Fargate


Amazon ECS


product-page-diagram_ECS_1
Amazon Elastic Container Registry
Build images and store using ECR or any other repository
Amazon Elastic Container Service
Define your application
Select container images and resources needed for your application
Amazon EC2
Launch containers on EC2
AWS Fargate
Launch containers on Fargate
Manage containers
Amazon ECS scales your application and manages your containers for availability

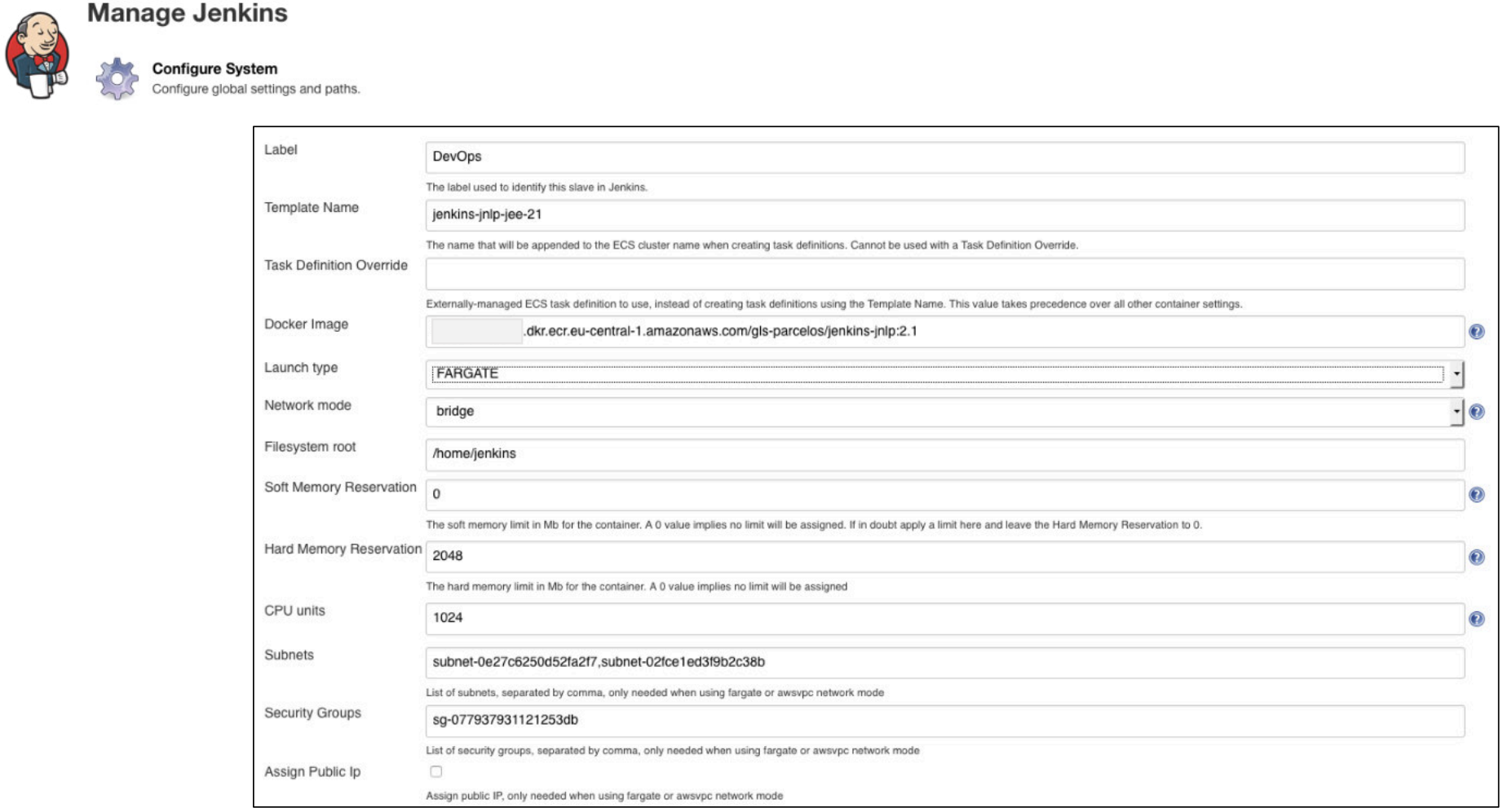

Container definition
Task definition
Service
Cluster


Just another Jenkins plugin

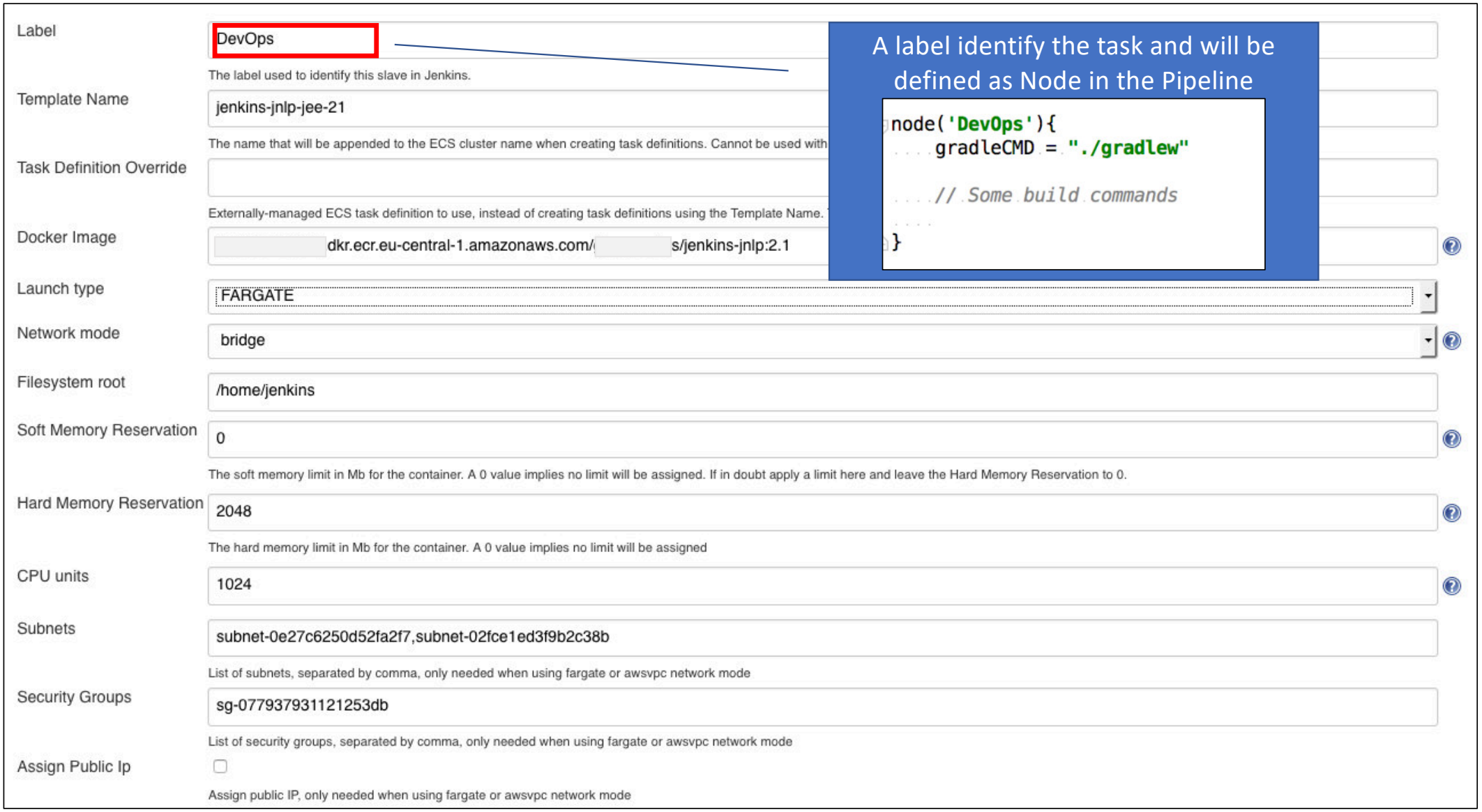


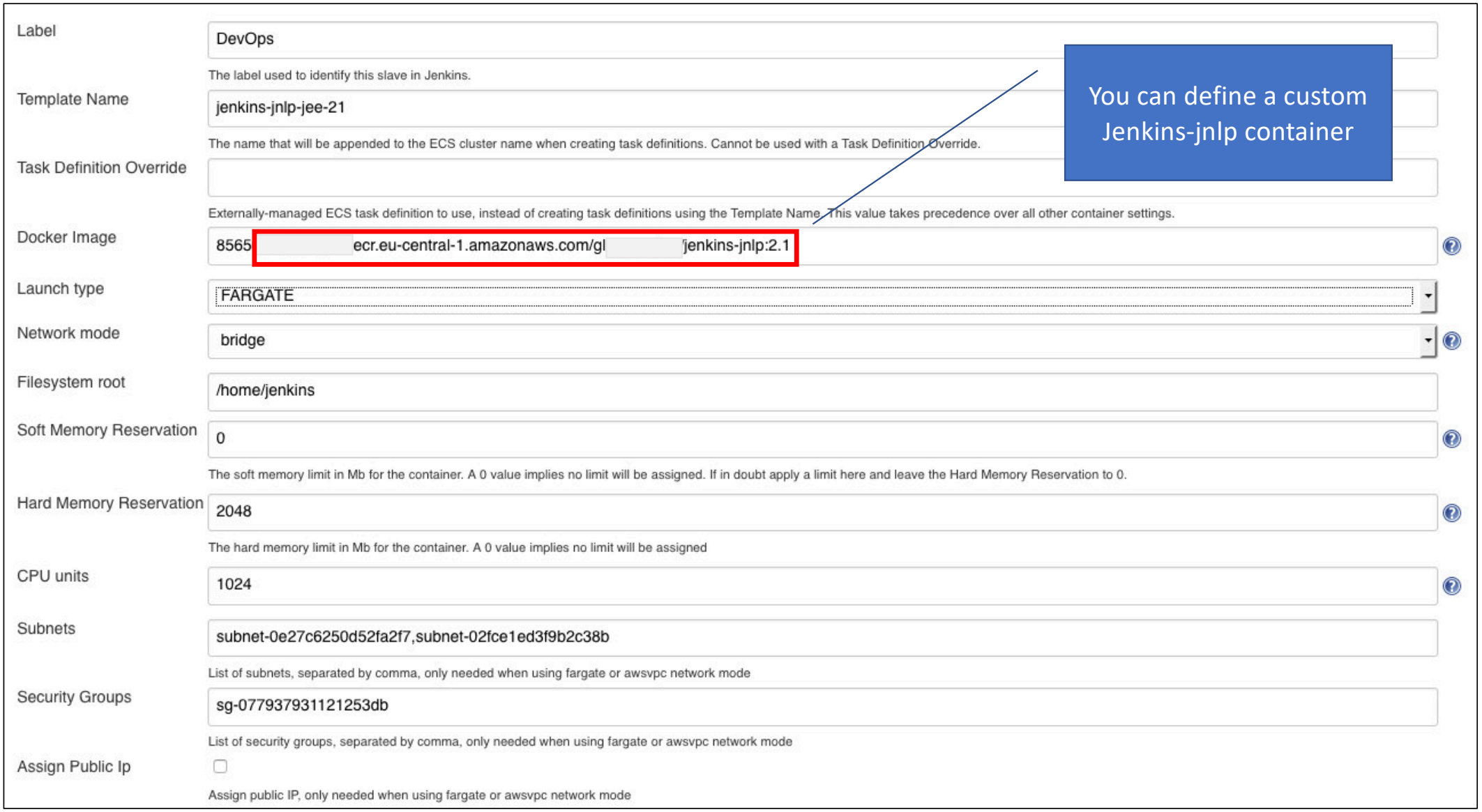
https://wiki.jenkins.io/display/JENKINS/Amazon+EC2+Container+Service+Plugin

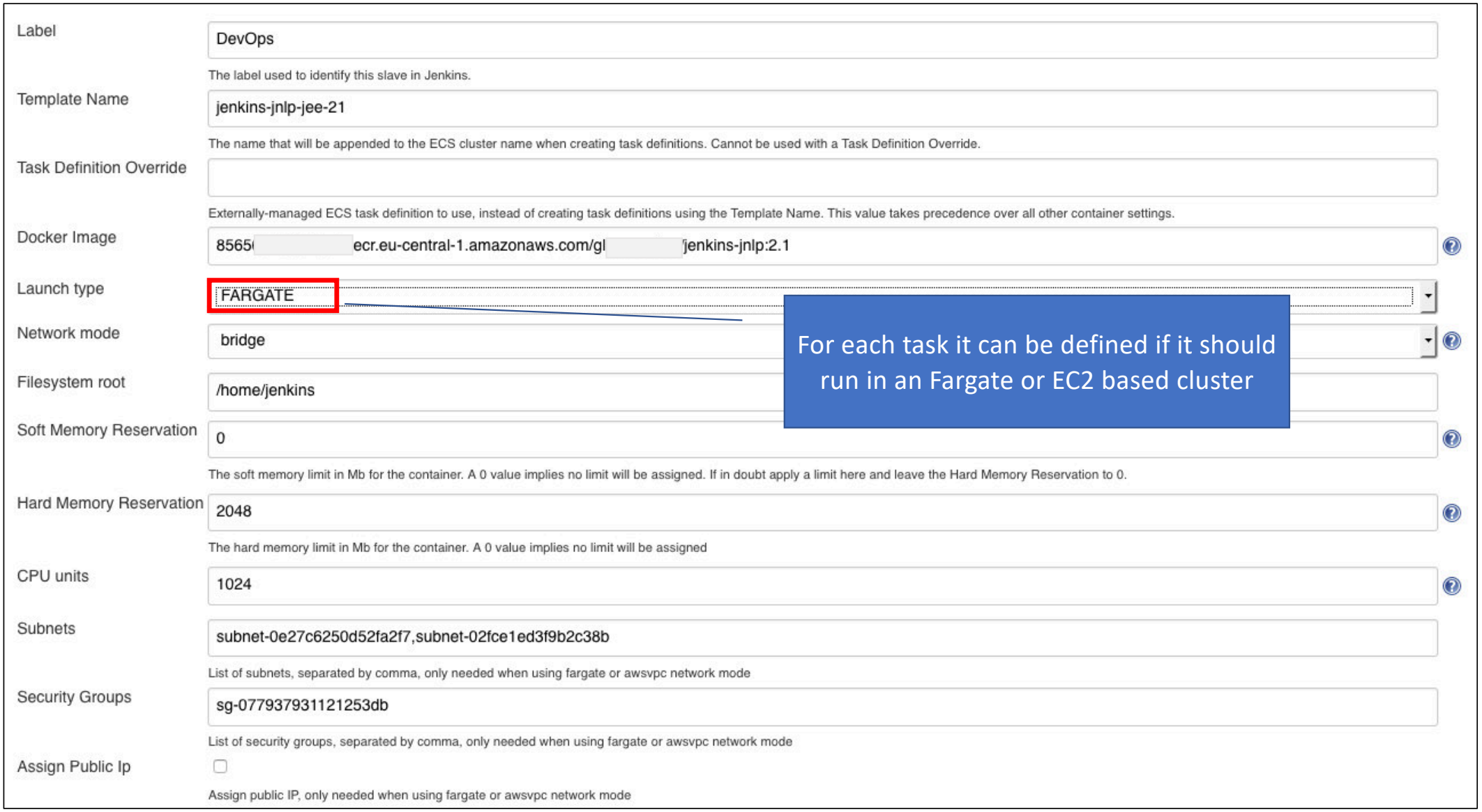
Create a cluster for the slaves

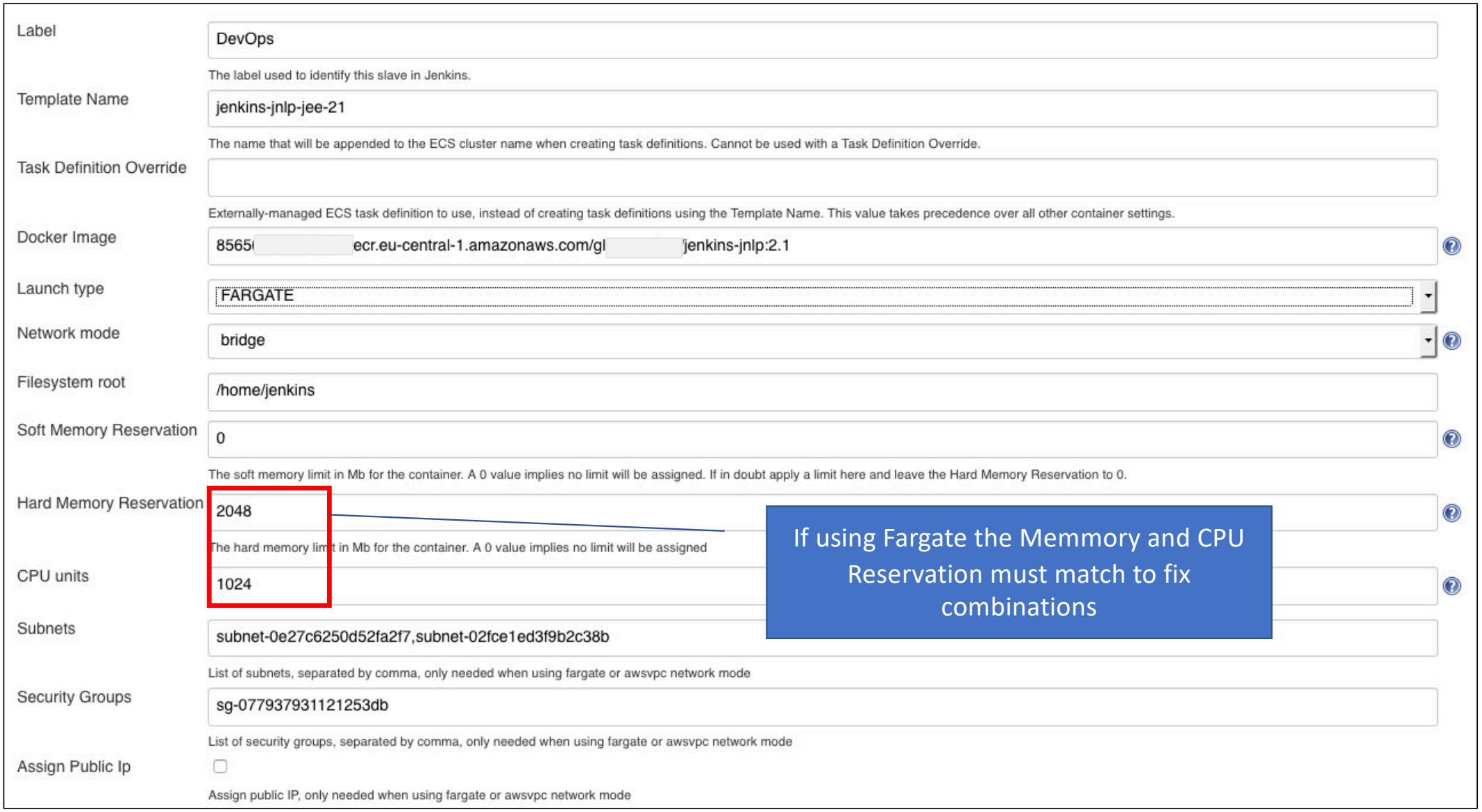


Define a Jenkins slave task

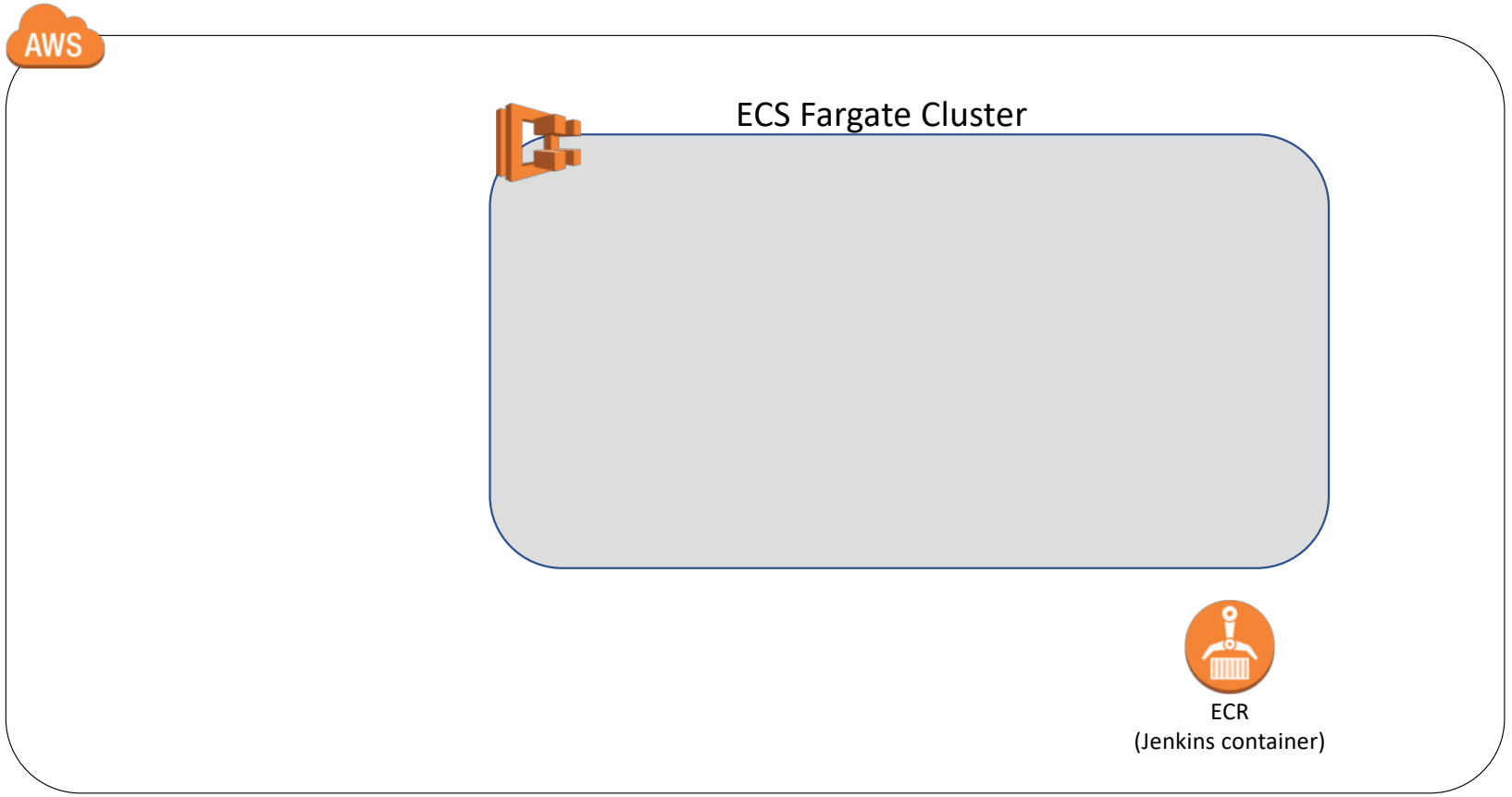

Label
DevOps
The label used to identify this slave in Jenkins.
Template Name
jenkins-jnlp-jee-21
The name that will be appended to the ECS cluster name when creating task definitions. Cannot be used with
Task Definition Override
Externally-managed ECS task definition to use, instead of creating task definitions using the Template Name.
Docker Image
dkr.ecr.eu-central-1.amazonaws.com/ s/jenkins-jnlp:2.1
Launch type
FARGATE
Network mode
bridge
Filesystem root
/home/jenkins
Soft Memory Reservation
0
The soft memory limit in Mb for the container. A 0 value implies no limit will be assigned. If in doubt apply a limit here and leave the Hard Memory Reservation to 0.
Hard Memory Reservation
2048
The hard memory limit in Mb for the container. A 0 value implies no limit will be assigned
CPU units
1024
Subnets
subnet-0e27c6250d52fa2f7,subnet-02fce1ed3f9b2c38b
List of subnets, separated by comma, only needed when using fargate or awsvpc network mode
Security Groups
sg-077937931121253db
List of security groups, separated by comma, only needed when using fargate or awsvpc network mode
Assign Public Ip
Assign public IP, only needed when using fargate or awsvpc network mode
A label identify the task and will be defined as Node in the Pipeline
node('DevOps'){
gradleCMD = "./gradlew"
// Some build commands
}


Label
DevOps
The label used to identify this slave in Jenkins.
Template Name
jenkins-jnlp-jee-21
The name that will be appended to the ECS cluster name when creating task definitions. Cannot be used with a Task Definition Override.
Task Definition Override
Externally-managed ECS task definition to use, instead of creating task definitions using the Template Name. This value takes precedence over all other container settings.
Docker Image
8565 ecr.eu-central-1.amazonaws.com/gl jenkins-jnlp:2.1
Launch type
FARGATE
Network mode
bridge
Filesystem root
/home/jenkins
Soft Memory Reservation
0
The soft memory limit in Mb for the container. A 0 value implies no limit will be assigned. If in doubt apply a limit here and leave the Hard Memory Reservation to 0.
Hard Memory Reservation
2048
The hard memory limit in Mb for the container. A 0 value implies no limit will be assigned
CPU units
1024
Subnets
subnet-0e27c6250d52fa2f7,subnet-02fce1ed3f9b2c38b
List of subnets, separated by comma, only needed when using fargate or awsvpc network mode
Security Groups
sg-077937931121253db
List of security groups, separated by comma, only needed when using fargate or awsvpc network mode
Assign Public Ip
Assign public IP, only needed when using fargate or awsvpc network mode
You can define a custom Jenkins-jnlp container

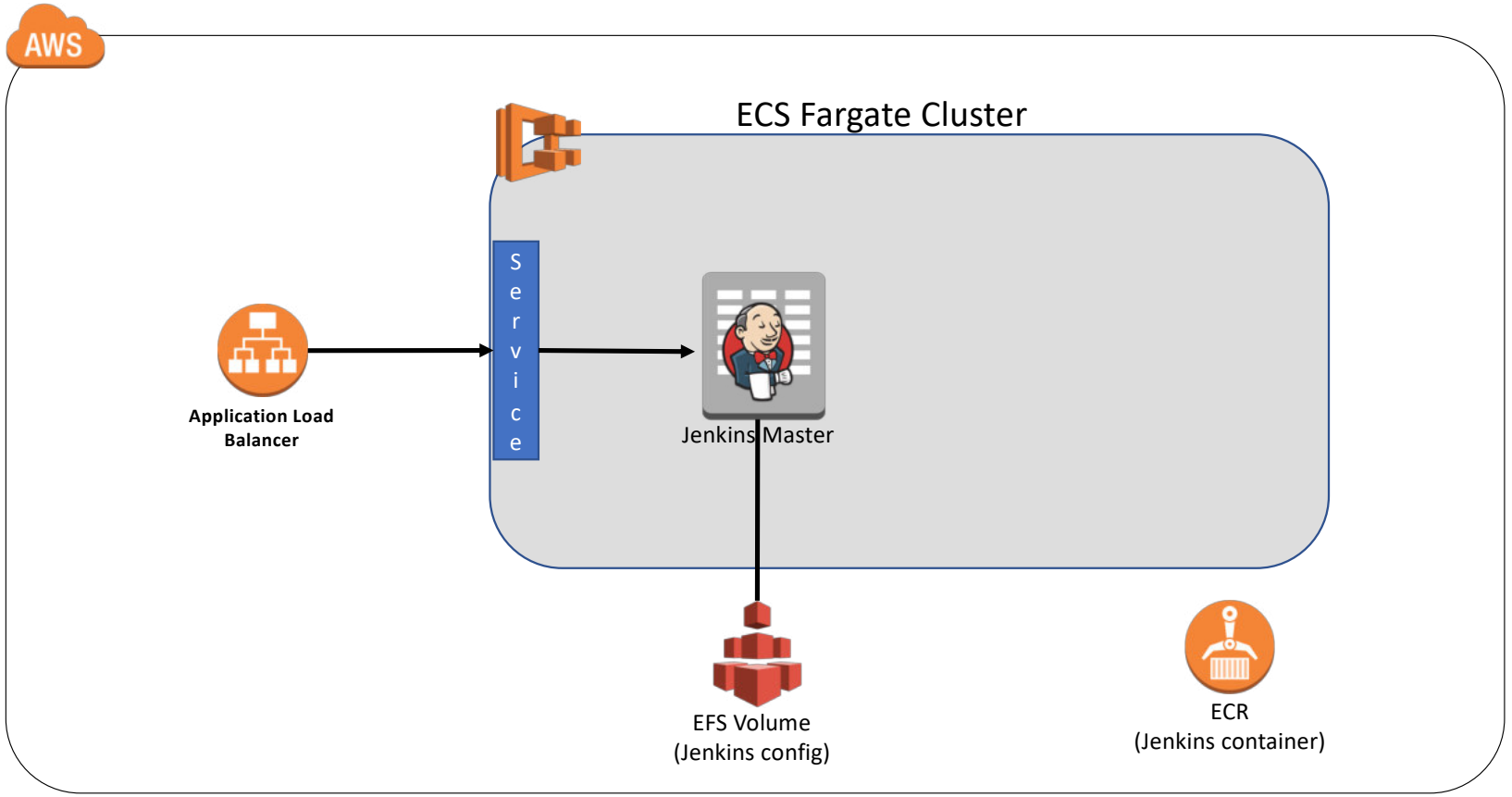

Label
DevOps
The label used to identify this slave in Jenkins.
Template Name
jenkins-jnlp-jee-21
The name that will be appended to the ECS cluster name when creating task definitions. Cannot be used with a Task Definition Override.
Task Definition Override
Externally-managed ECS task definition to use, instead of creating task definitions using the Template Name. This value takes precedence over all other container settings.
Docker Image
8565 ecr.eu-central-1.amazonaws.com/gl jenkins-jnlp:2.1
Launch type
FARGATE
Network mode
bridge
Filesystem root
/home/jenkins
Soft Memory Reservation
0
The soft memory limit in Mb for the container. A 0 value implies no limit will be assigned. If in doubt apply a limit here and leave the Hard Memory Reservation to 0.
Hard Memory Reservation
2048
The hard memory limit in Mb for the container. A 0 value implies no limit will be assigned
CPU units
1024
Subnets
subnet-0e27c6250d52fa2f7,subnet-02fce1ed3f9b2c38b
List of subnets, separated by comma, only needed when using fargate or awsvpc network mode
Security Groups
sg-077937931121253db
List of security groups, separated by comma, only needed when using fargate or awsvpc network mode
Assign Public Ip
Assign public IP, only needed when using fargate or awsvpc network mode
For each task it can be defined if it should run in an Fargate or EC2 based cluster


Label
DevOps
The label used to identify this slave in Jenkins.
Template Name
jenkins-jnlp-jee-21
The name that will be appended to the ECS cluster name when creating task definitions. Cannot be used with a Task Definition Override.
Task Definition Override
Externally-managed ECS task definition to use, instead of creating task definitions using the Template Name. This value takes precedence over all other container settings.
Docker Image
8565 ecr.eu-central-1.amazonaws.com/gl jenkins-jnlp:2.1
Launch type
FARGATE
Network mode
bridge
Filesystem root
/home/jenkins
Soft Memory Reservation
0
The soft memory limit in Mb for the container. A 0 value implies no limit will be assigned. If in doubt apply a limit here and leave the Hard Memory Reservation to 0.
Hard Memory Reservation
2048
The hard memory limit in Mb for the container. A 0 value implies no limit will be assigned
CPU units
1024
Subnets
subnet-0e27c6250d52fa2f7,subnet-02fce1ed3f9b2c38b
List of subnets, separated by comma, only needed when using fargate or awsvpc network mode
Security Groups
sg-077937931121253db
List of security groups, separated by comma, only needed when using fargate or awsvpc network mode
Assign Public Ip
Assign public IP, only needed when using fargate or awsvpc network mode
If using Fargate the Memmory and CPU Reservation must match to fix combinations

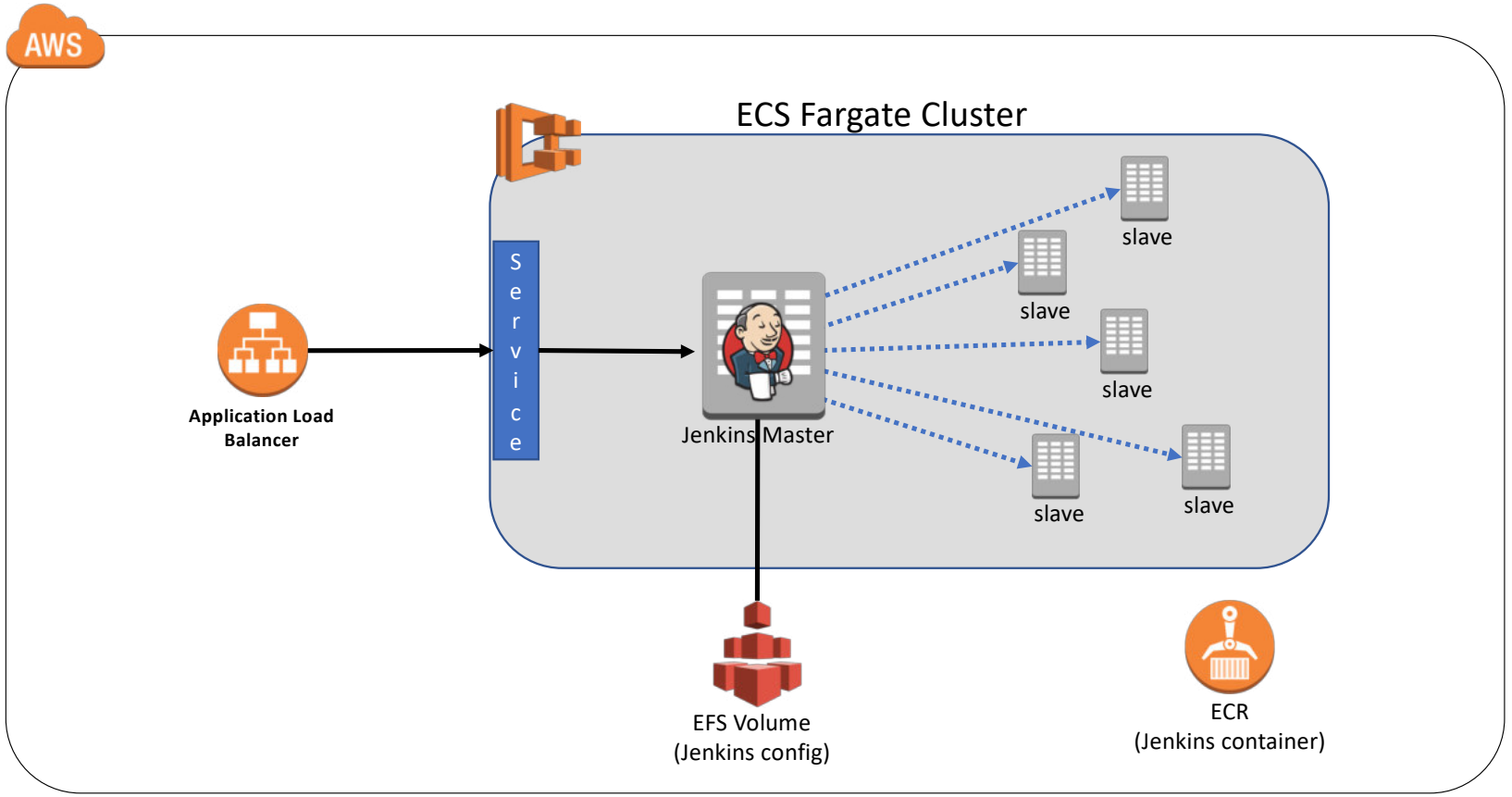

Ok lets design our Jenkins cluster!



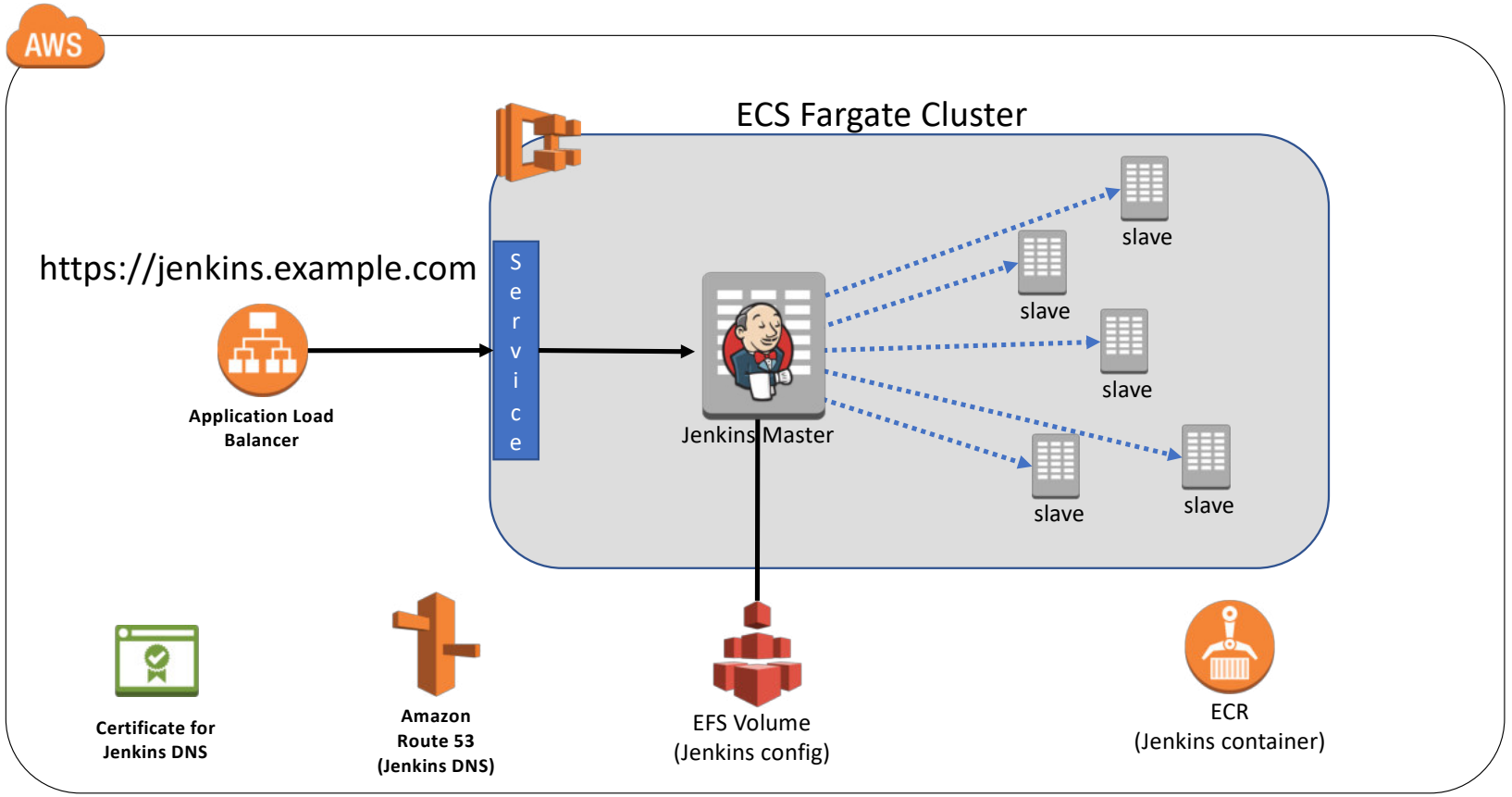
HashiCorp
Terraform



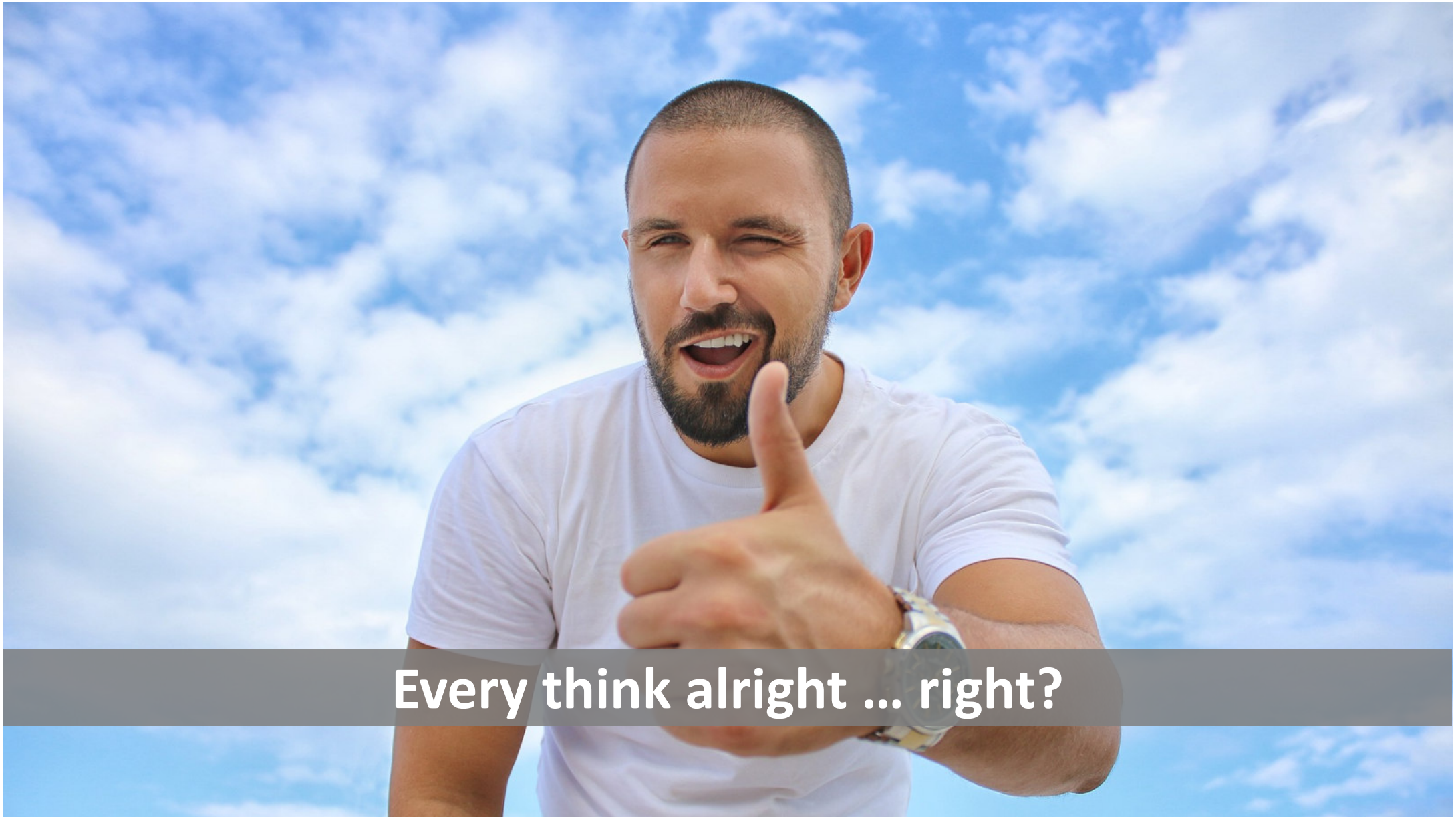
HashiCorp Terraform



HashiCorp Terraform



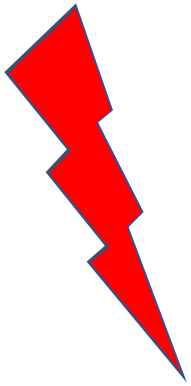
HashiCorp Terraform


Every think alright … right?

It is not that easy

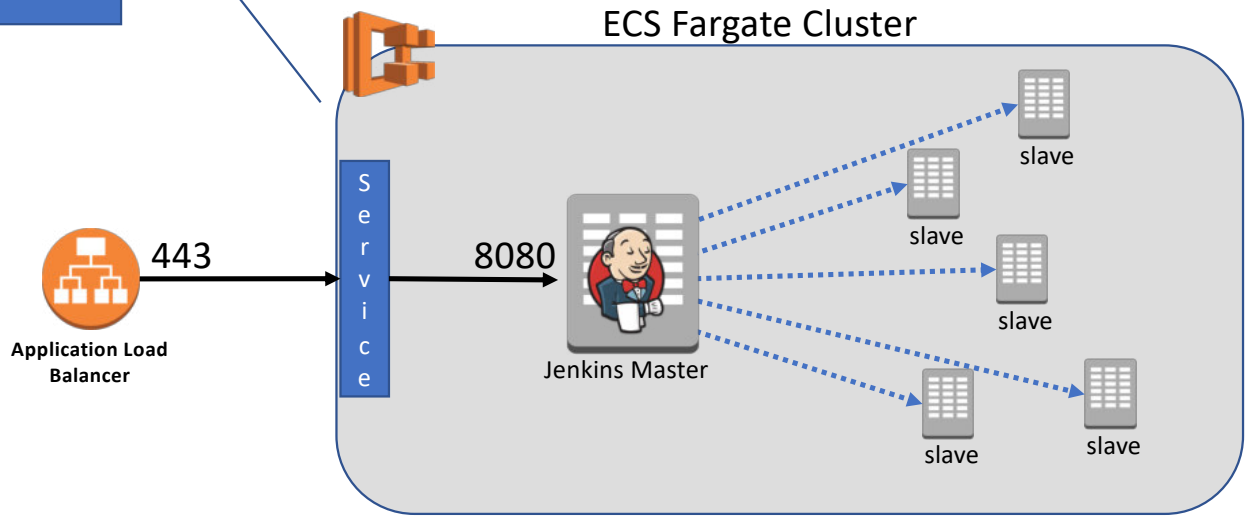
Load balancing in ECS cloud be very difficult

ECS

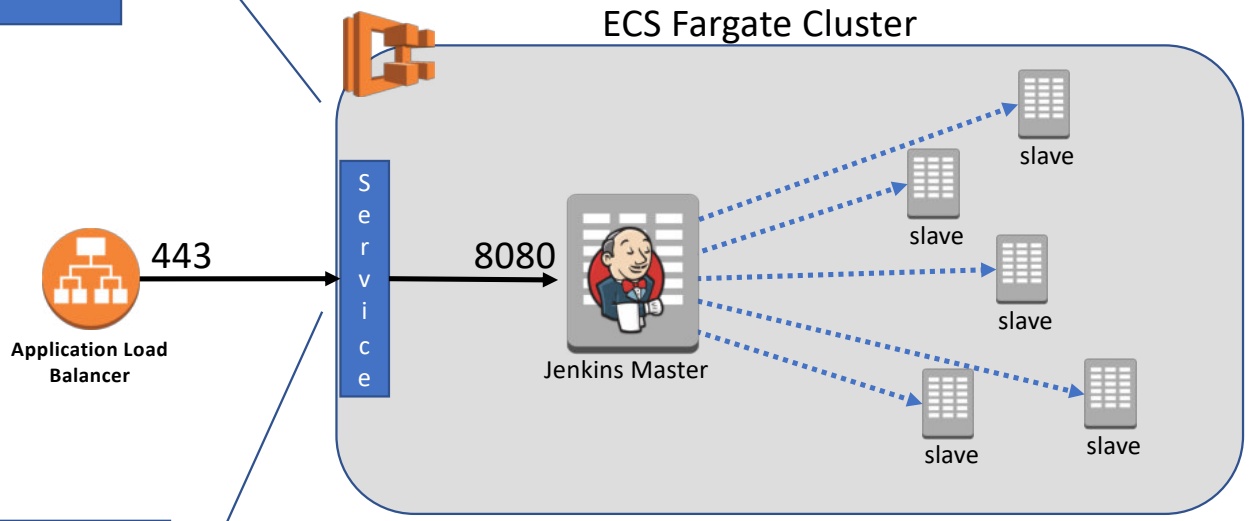


Application Load Balancer

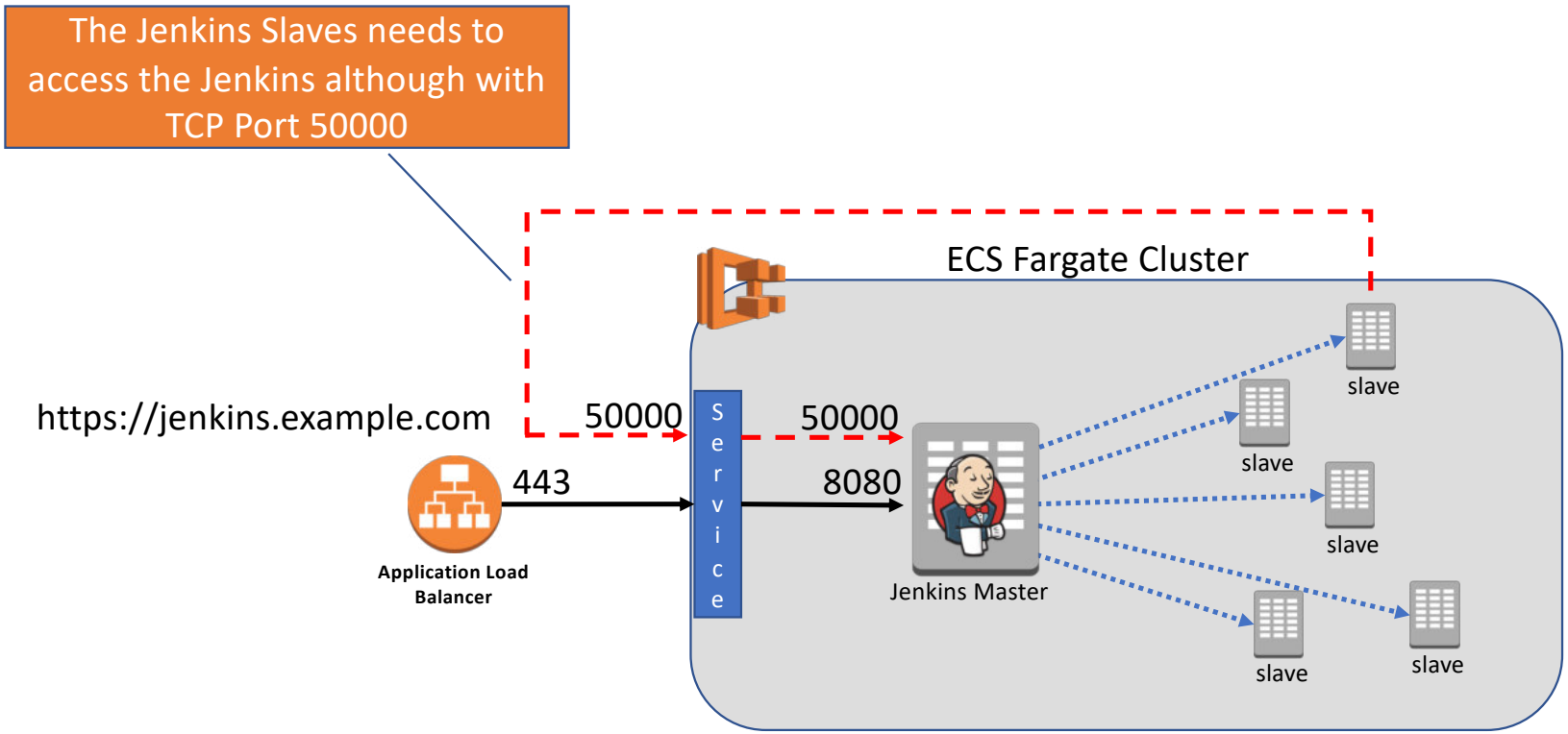
ECS can manage the configuration of your Service to the loadbalancer.



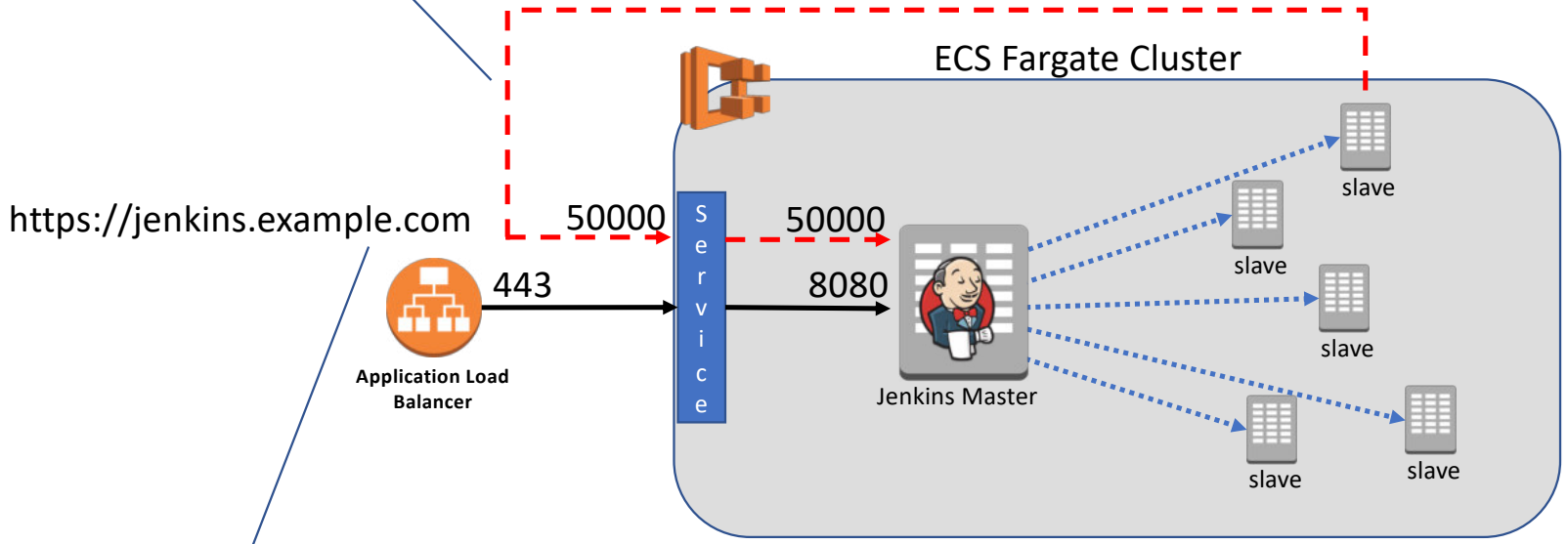
ECS can manage the configuration of your Service to the loadbalancer.



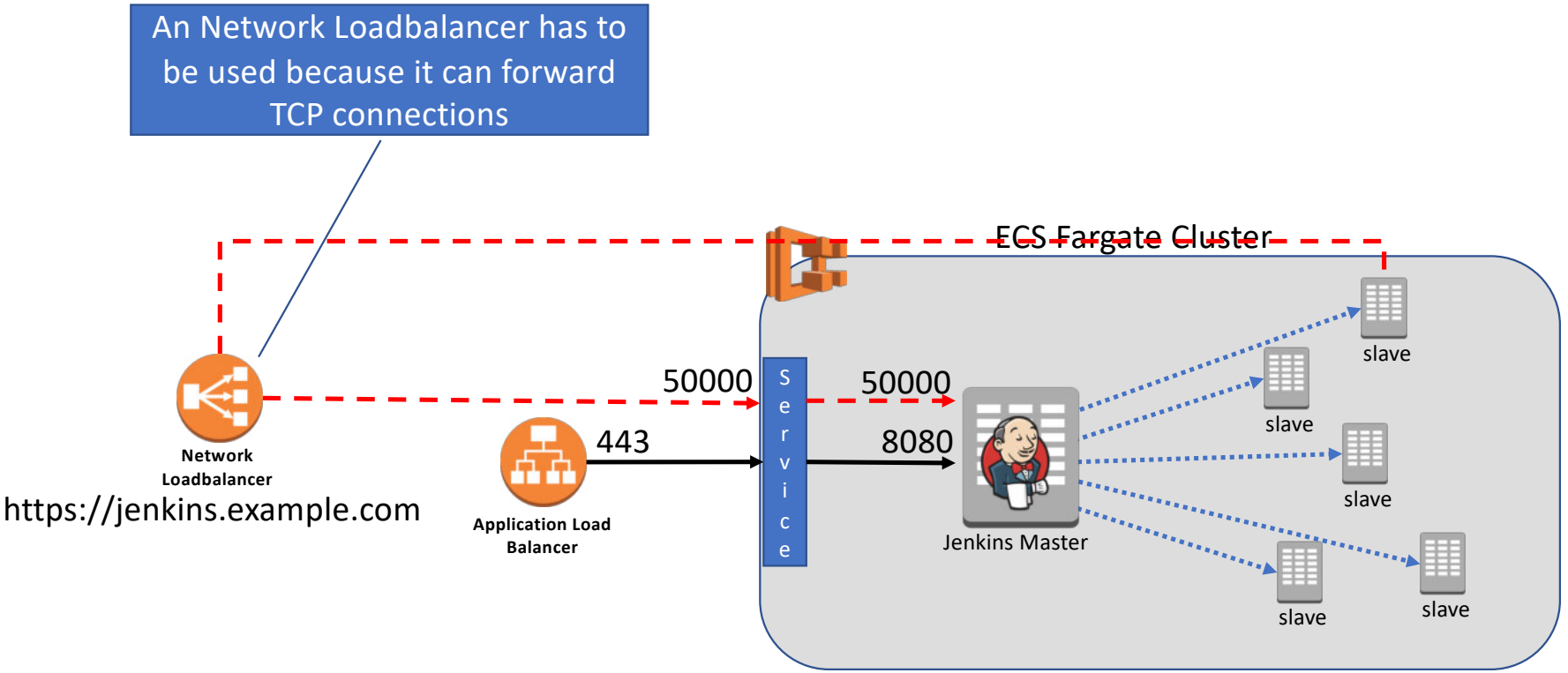
But only for one Port!

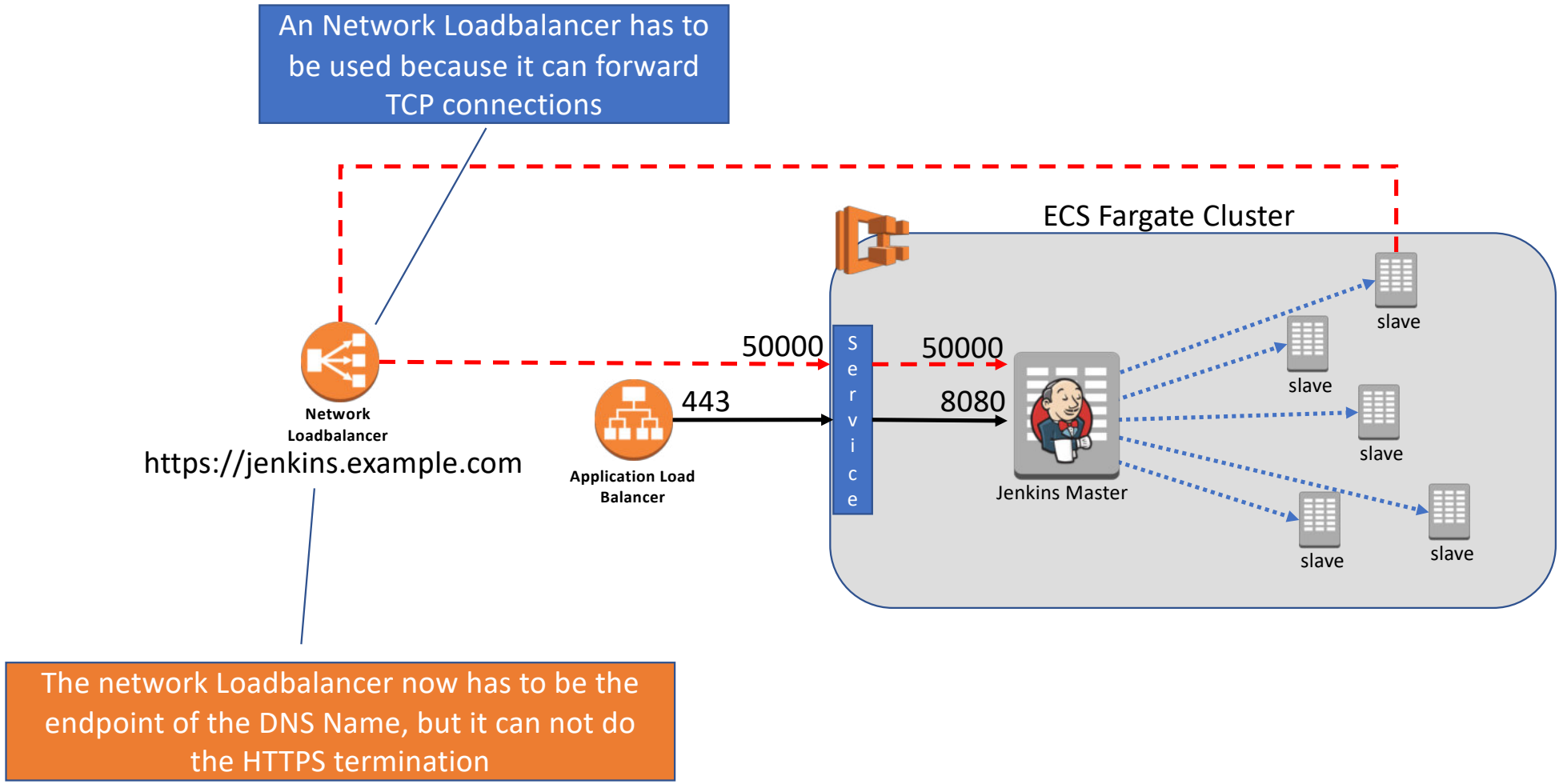

The Jenkins Slaves needs to access the Jenkins although with TCP Port 50000
ECS Fargate Cluster
https://jenkins.example.com
50000
50000
443
8080
Service
Application Load Balancer
Jenkins Master
slave
slave
slave
slave
slave

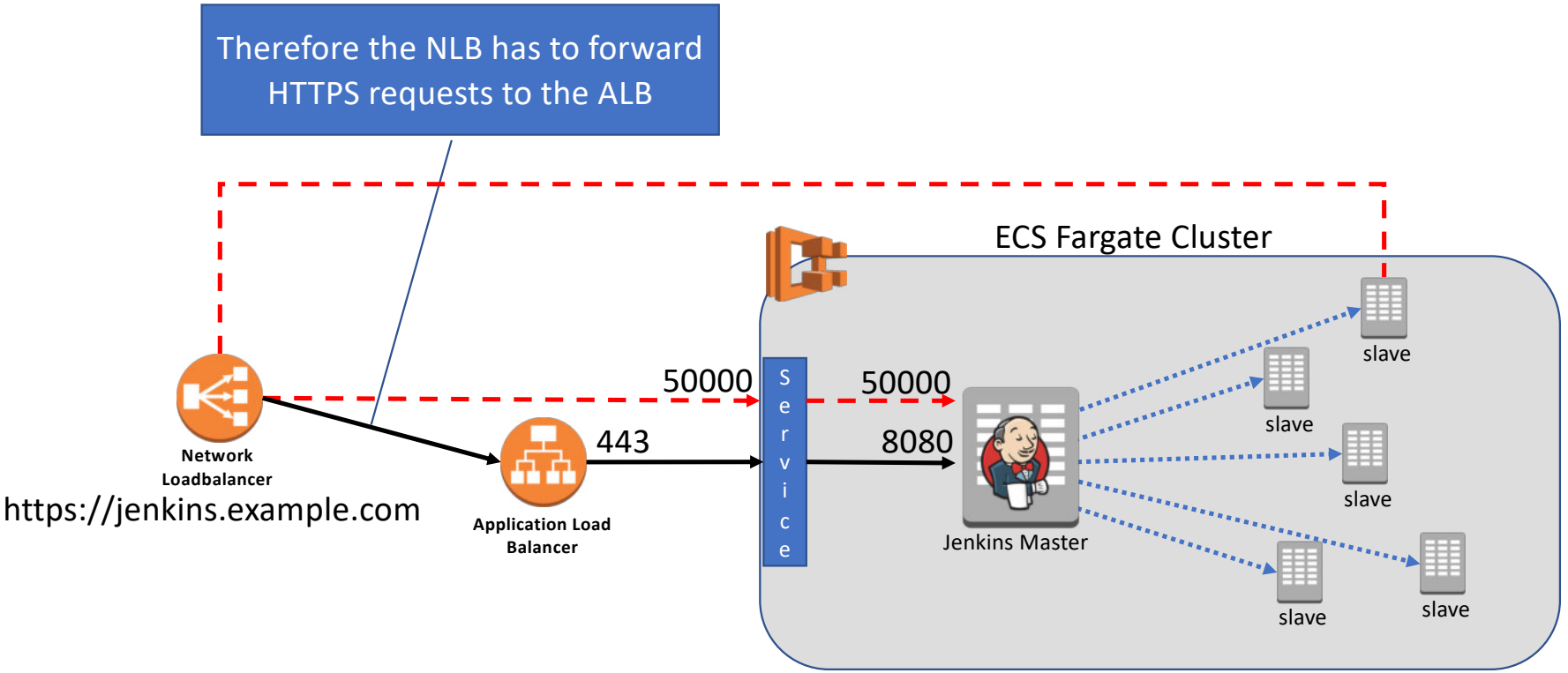
The Jenkins Slaves needs to access the Jenkins although with TCP Port 50000

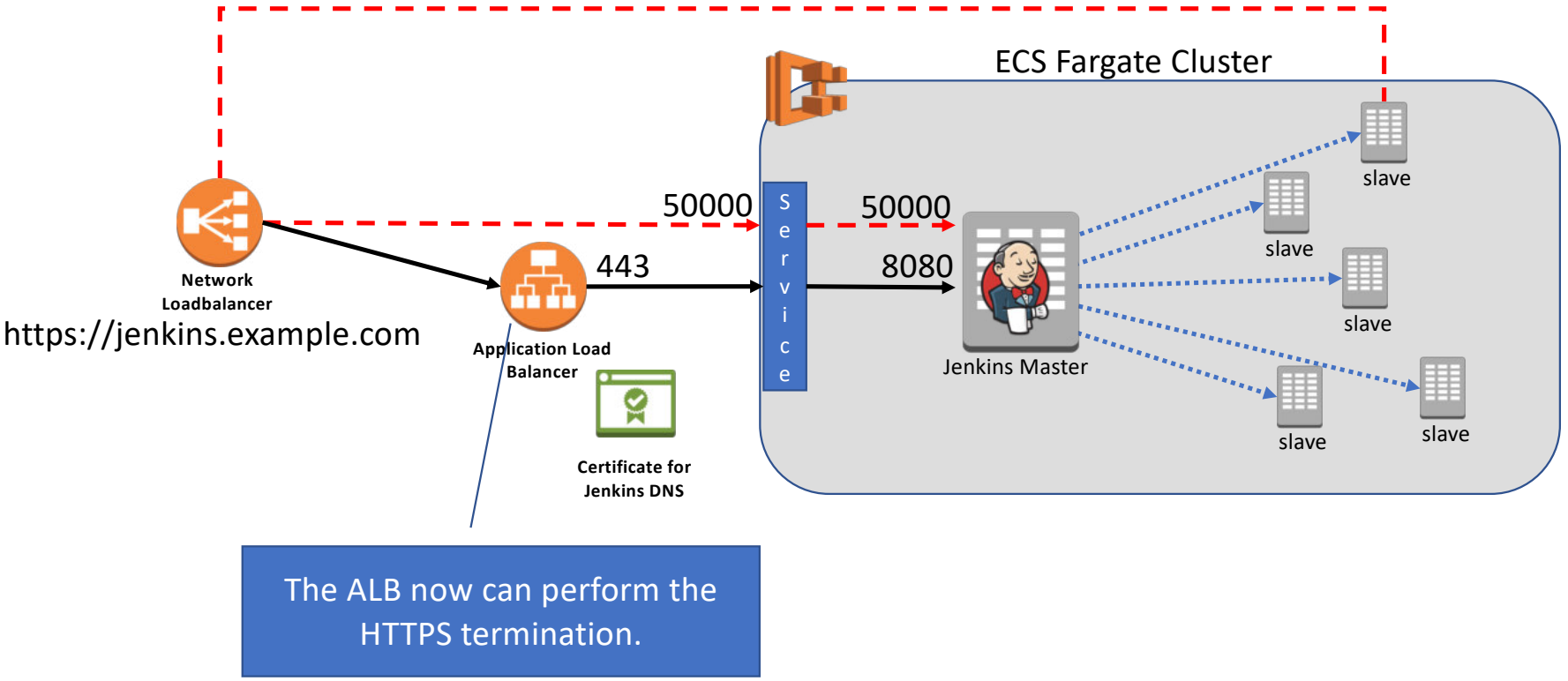


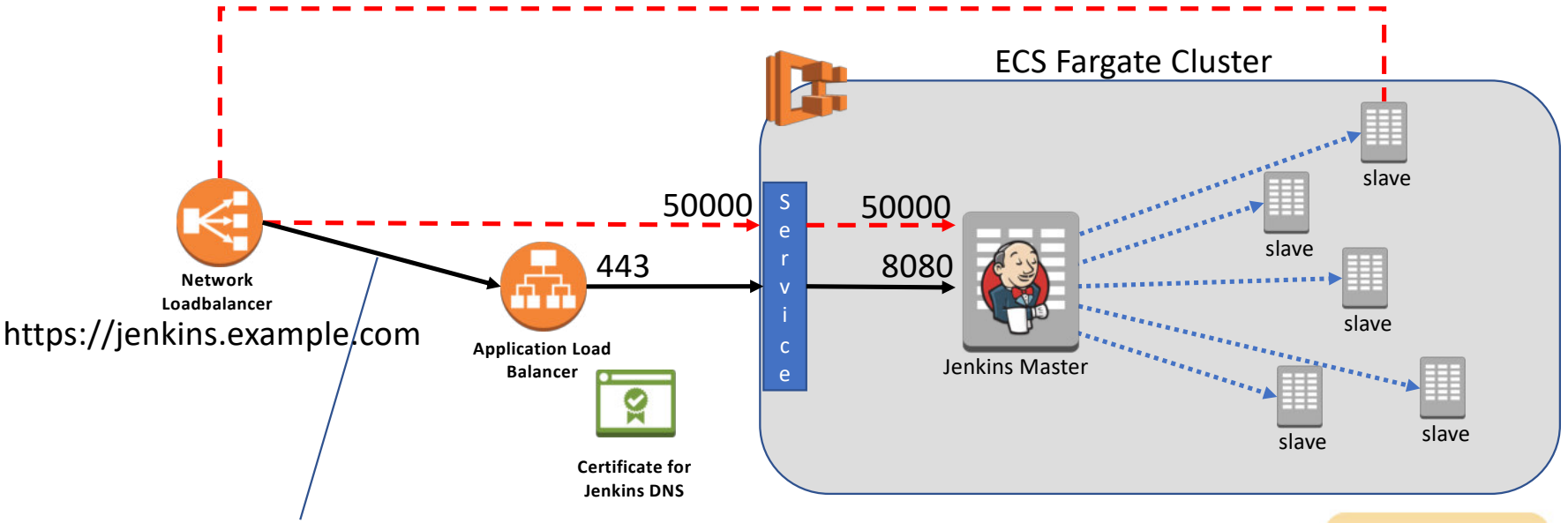
The Plugin need to use the same Jenkins Host Name as you use to configure the Plugin.


An Network Loadbalancer has to be used because it can forward TCP connections
ECS Fargate Cluster
Network Loadbalancer
https://jenkins.example.com
50000
443
Application Load Balancer
Service
50000
8080
Jenkins Master
slave
slave
slave
slave
slave

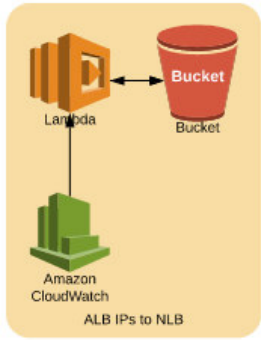

An Network Loadbalancer has to be used because it can forward TCP connections
ECS Fargate Cluster
50000
50000
443
8080
Service
Network Loadbalancer
https://jenkins.example.com
Application Load Balancer
Jenkins Master
slave
slave
slave
slave
slave
The network Loadbalancer now has to be the endpoint of the DNS Name, but it can not do the HTTPS termination


Therefore the NLB has to forward HTTPS requests to the ALB
ECS Fargate Cluster
Network Loadbalancer
https://jenkins.example.com
Application Load Balancer
443
50000
50000
8080
Service
Jenkins Master
slave
slave
slave
slave
slave

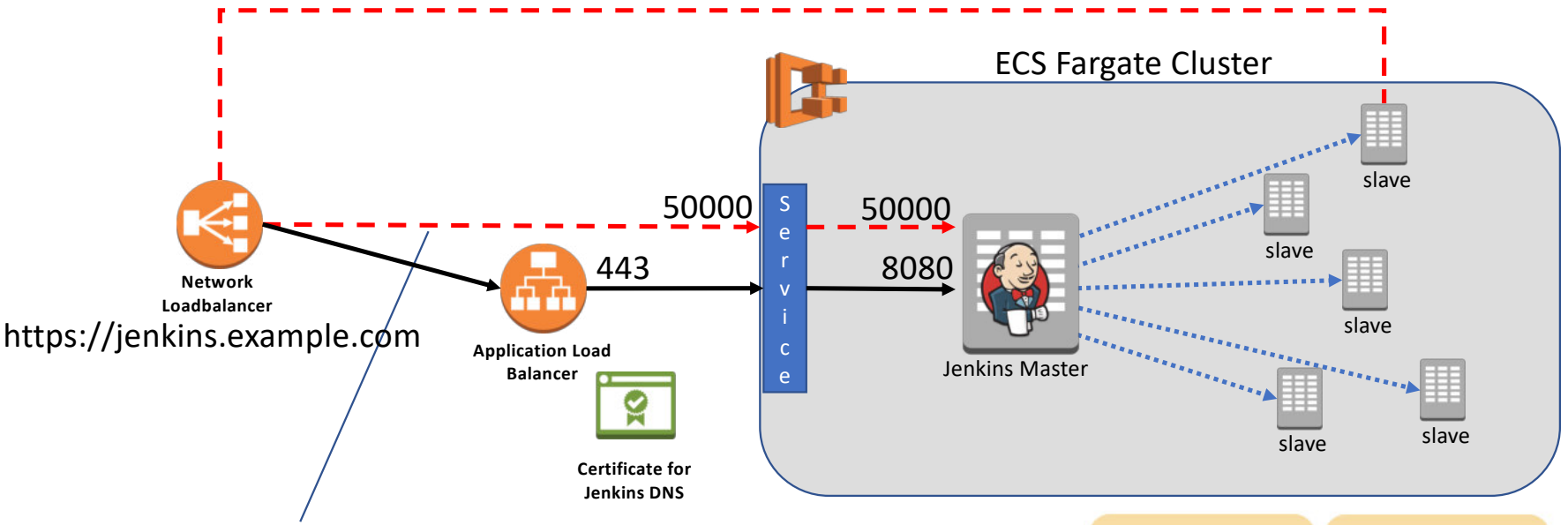

ECS Fargate Cluster
Network
Loadbalancer
https://jenkins.example.com
50000
443
Application Load
Balancer
Certificate for
Jenkins DNS
S
e
r
v
i
c
e
50000
8080
Jenkins Master
slave
slave
slave
slave
slave
The ALB now can perform the
HTTPS termination.



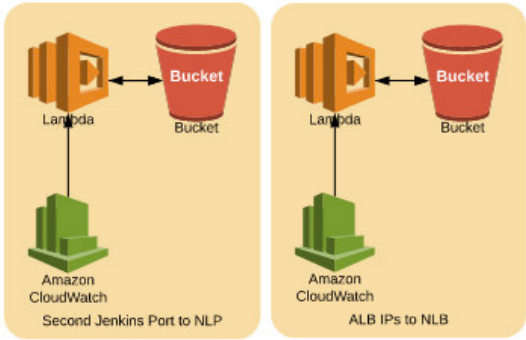
As target groups the NLB has to use the IP Addresses of the ALB. Because they are dynamic, a Lambda has to be used to dynamically update them



https://aws.amazon.com/de/blogs/networking-and-content-delivery/using-static-ip-addresses-for-application-load-balancers/



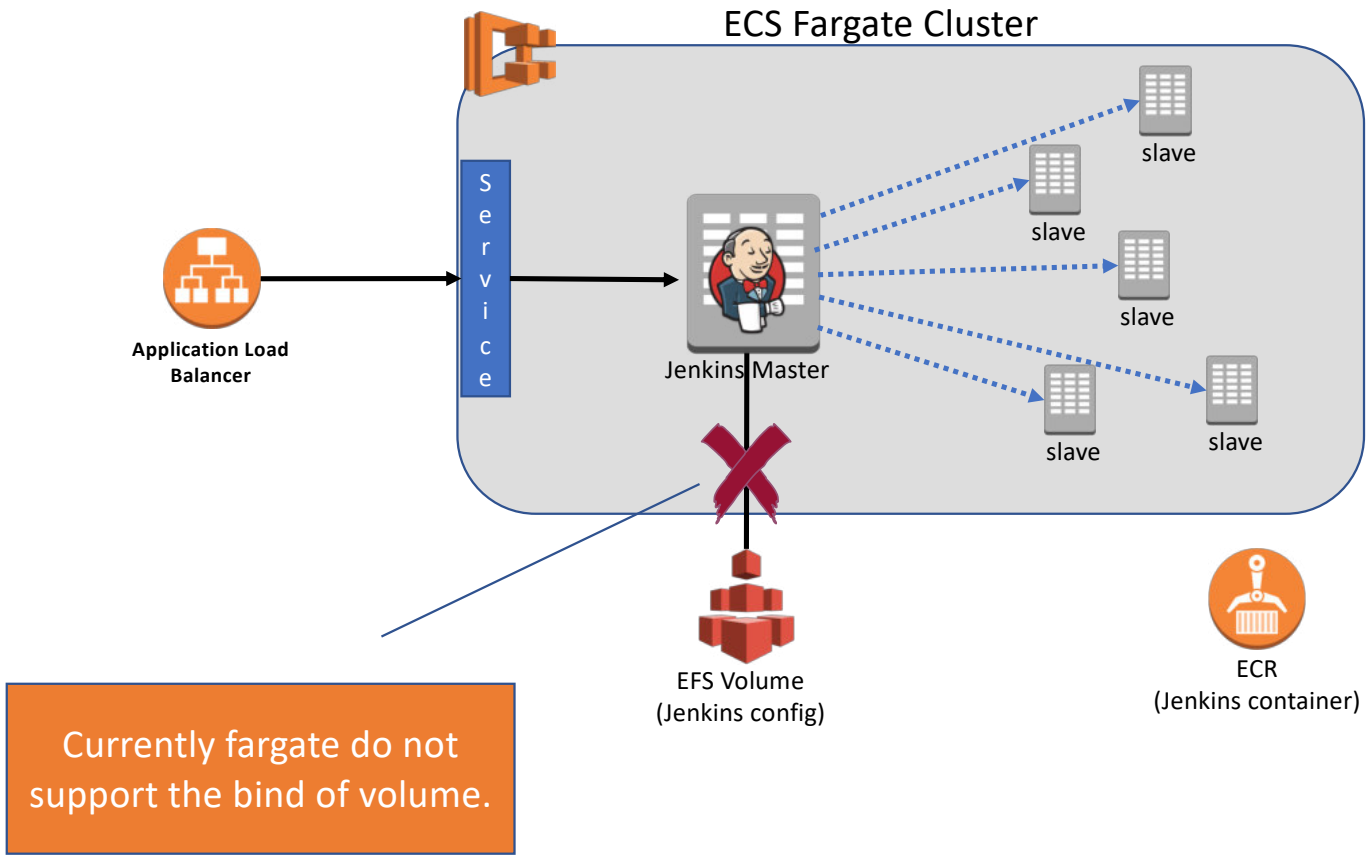
Same with the dynamic IPs of the Jenkins Container for the Second Port in the NLB. Here we use a modified version of the other Lambda.

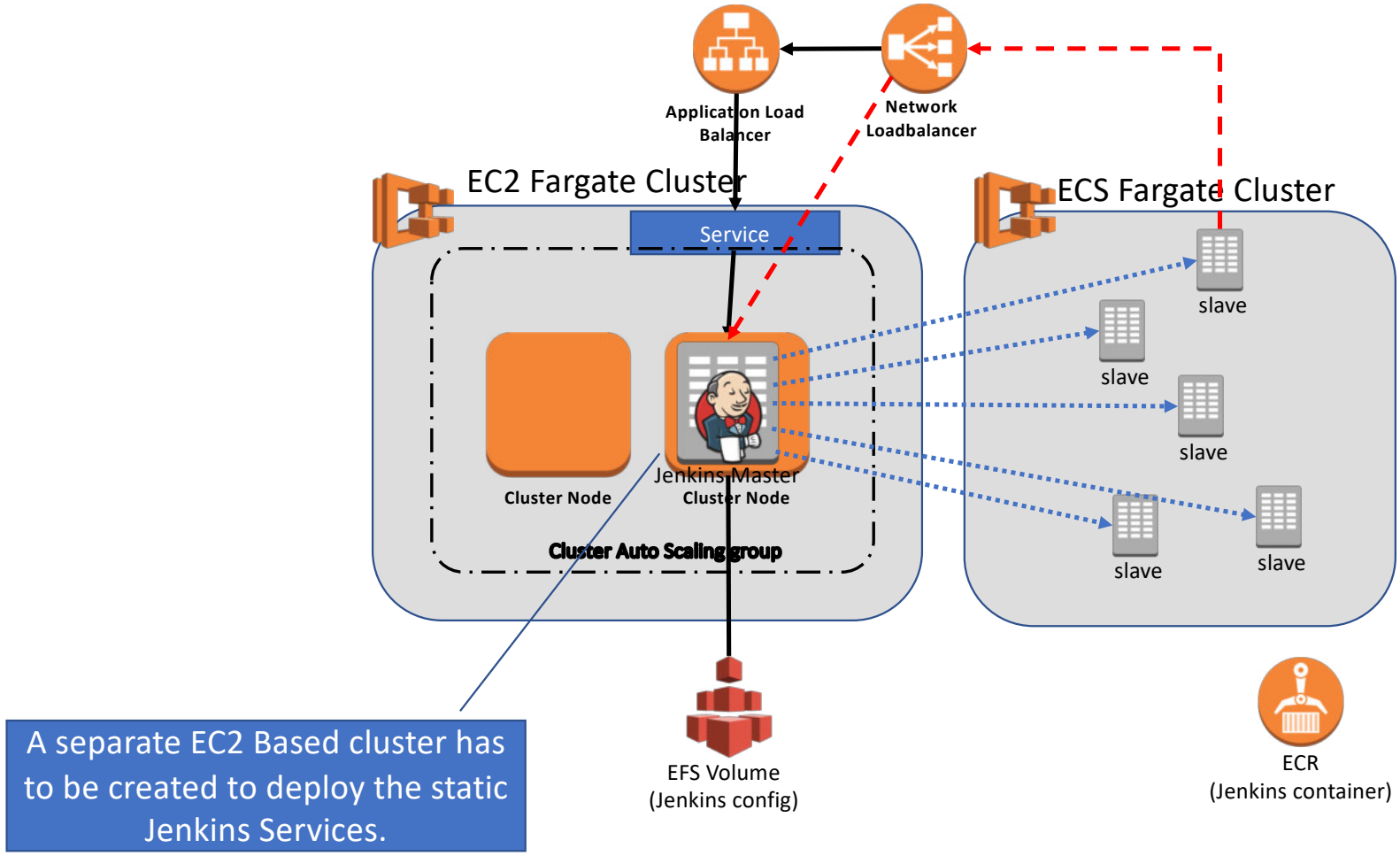


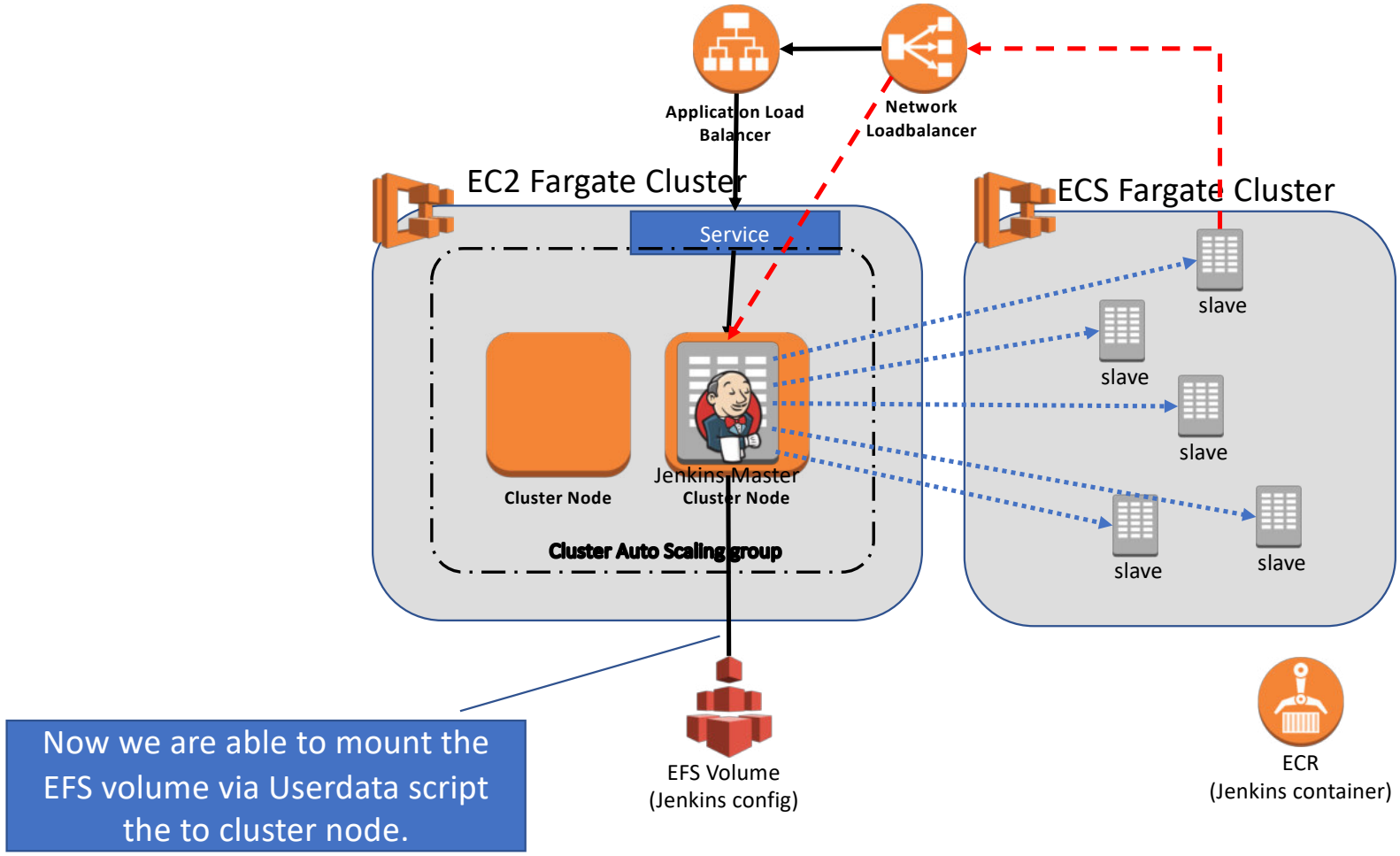
https://aws.amazon.com/de/blogs/networking-and-content-delivery/using-static-ip-addresses-for-application-load-balancers/

Fargate don‘t support Volume mounts

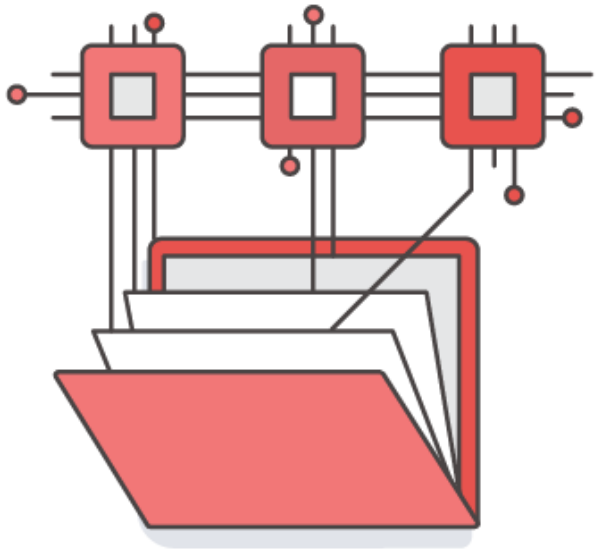
AWS Fargate

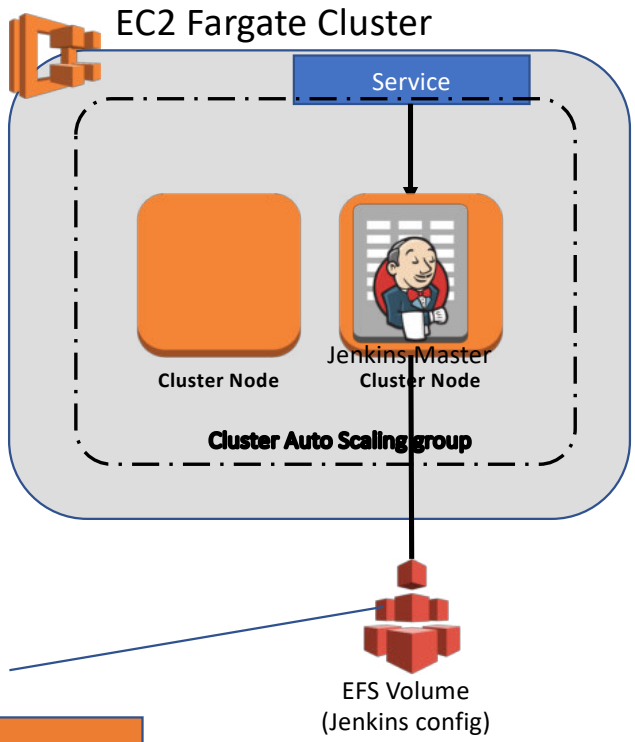

ECS Fargate Cluster
Service
Application Load Balancer
Jenkins Master
slave
slave
slave
slave
slave
EFS Volume
(Jenkins config)
ECR
(Jenkins container)
Currently fargate do not support the bind of volume.


Application Load Balancer
Network Loadbalancer
EC2 Fargate Cluster
ECS Fargate Cluster
Service
Cluster Node
Jenkins Master
Cluster Node
Cluster Auto Scaling group
slave
slave
slave
slave
slave
EFS Volume
(Jenkins config)
ECR
(Jenkins container)
A separate EC2 Based cluster has to be created to deploy the static Jenkins Services.

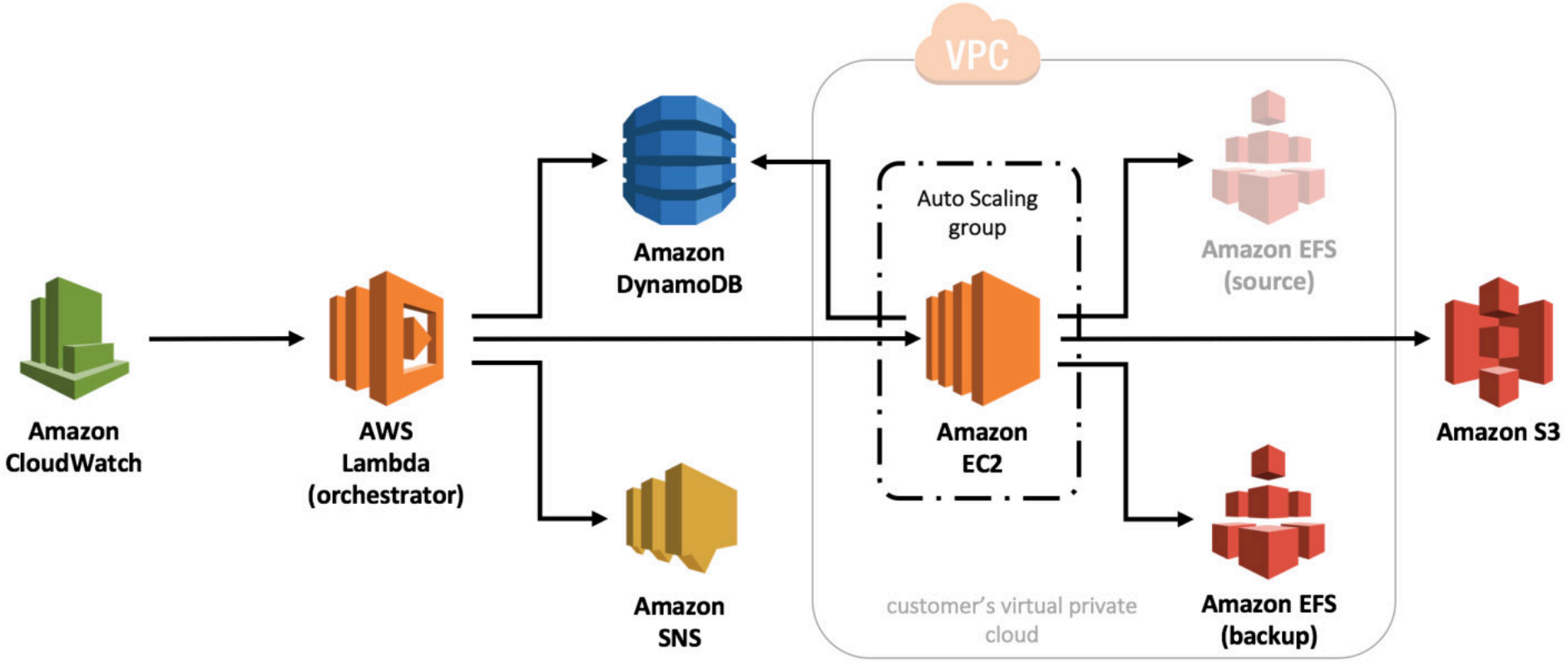

Application Load Balancer
Network Loadbalancer
EC2 Fargate Cluster
ECS Fargate Cluster
Service
Cluster Node
Jenkins Master
Cluster Node
Cluster Auto Scaling group
slave
slave
slave
slave
slave
EFS Volume
(Jenkins config)
ECR
(Jenkins container)
Now we are able to mount the EFS volume via Userdata script the to cluster node.

EFS don’t support snapshotting



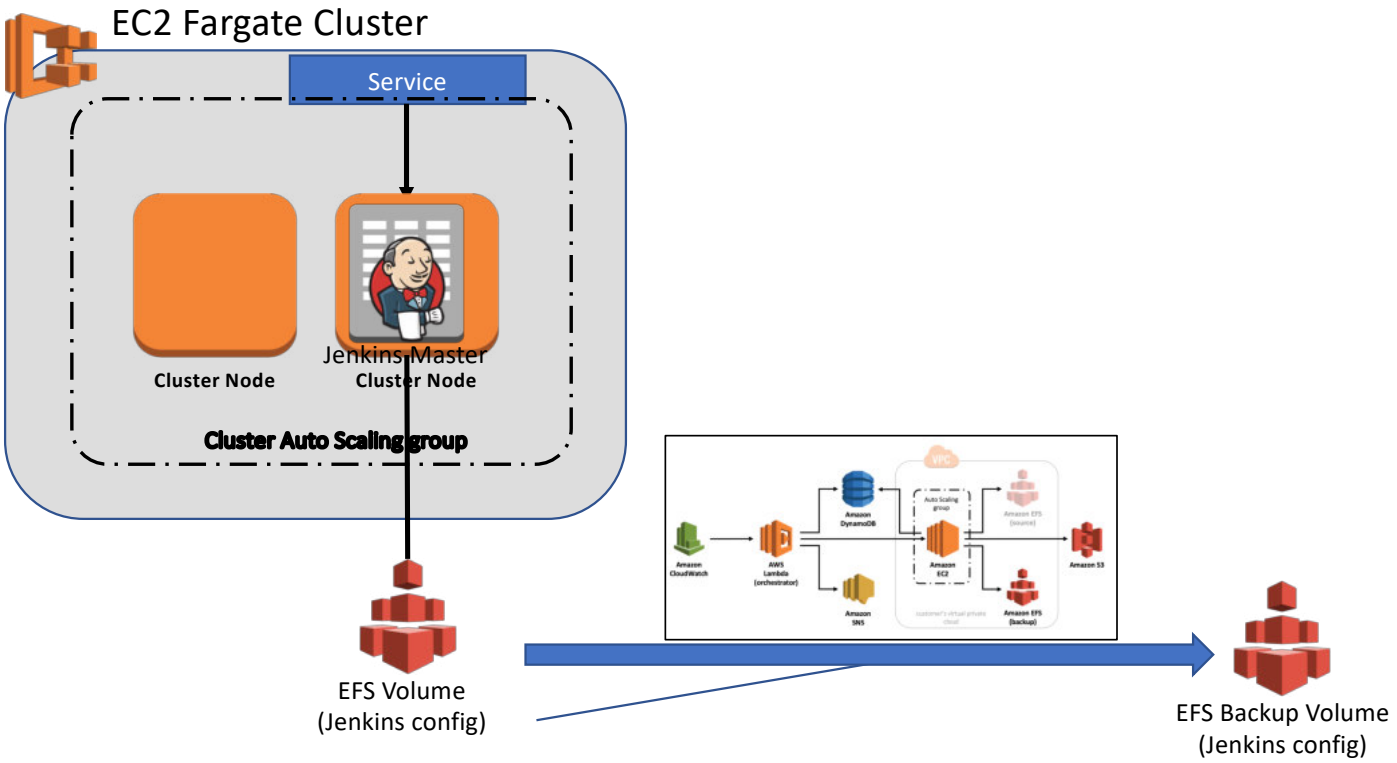


There is no standard backup or snapshot solution for EFS

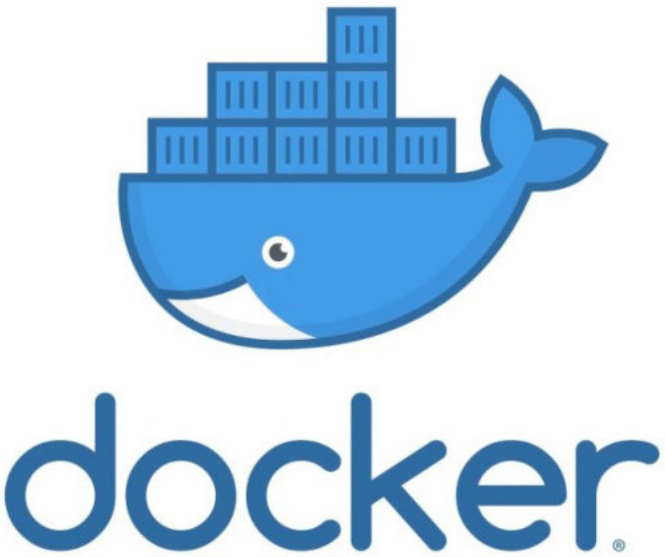


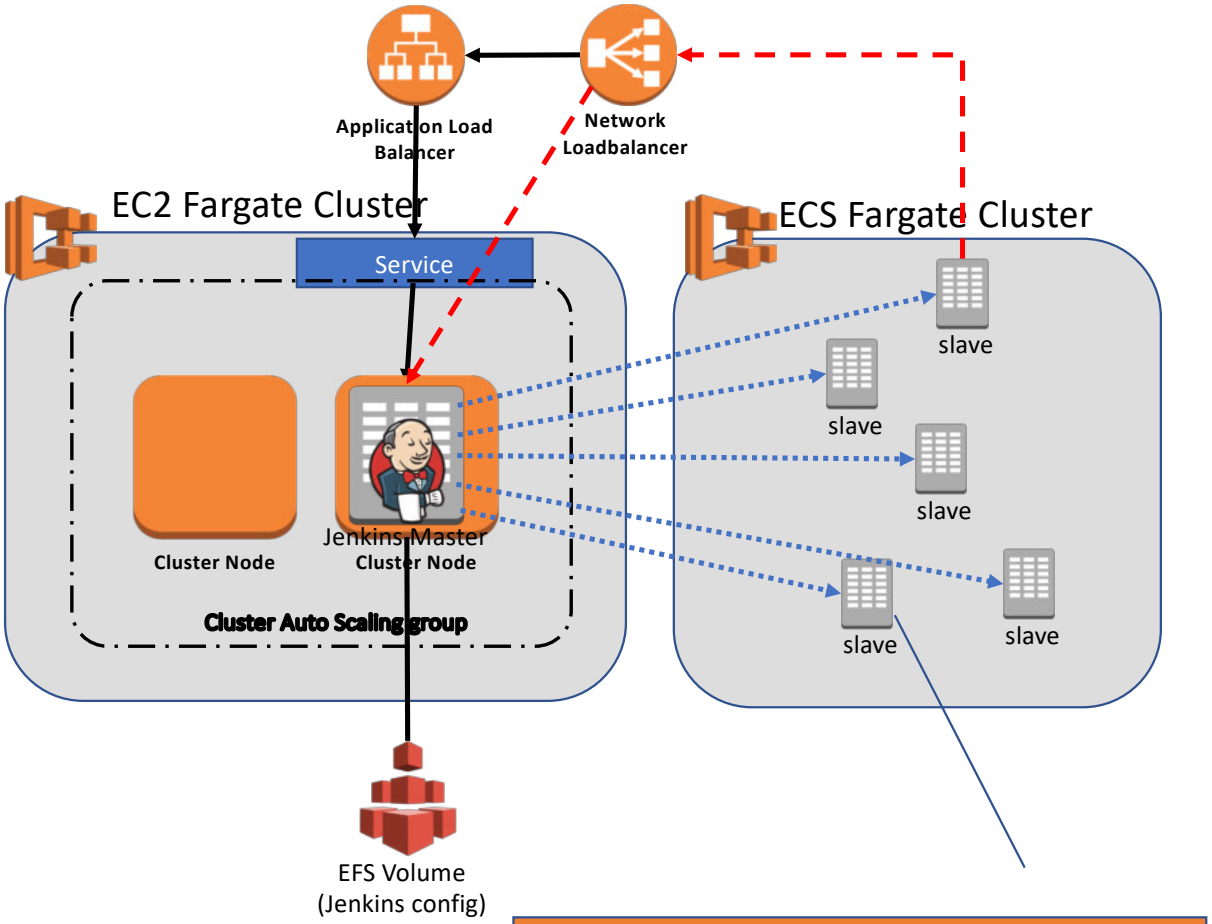
AWS provides a deployable Backup solution

https://aws.amazon.com/de/answers/infrastructure-management/efs-backup/

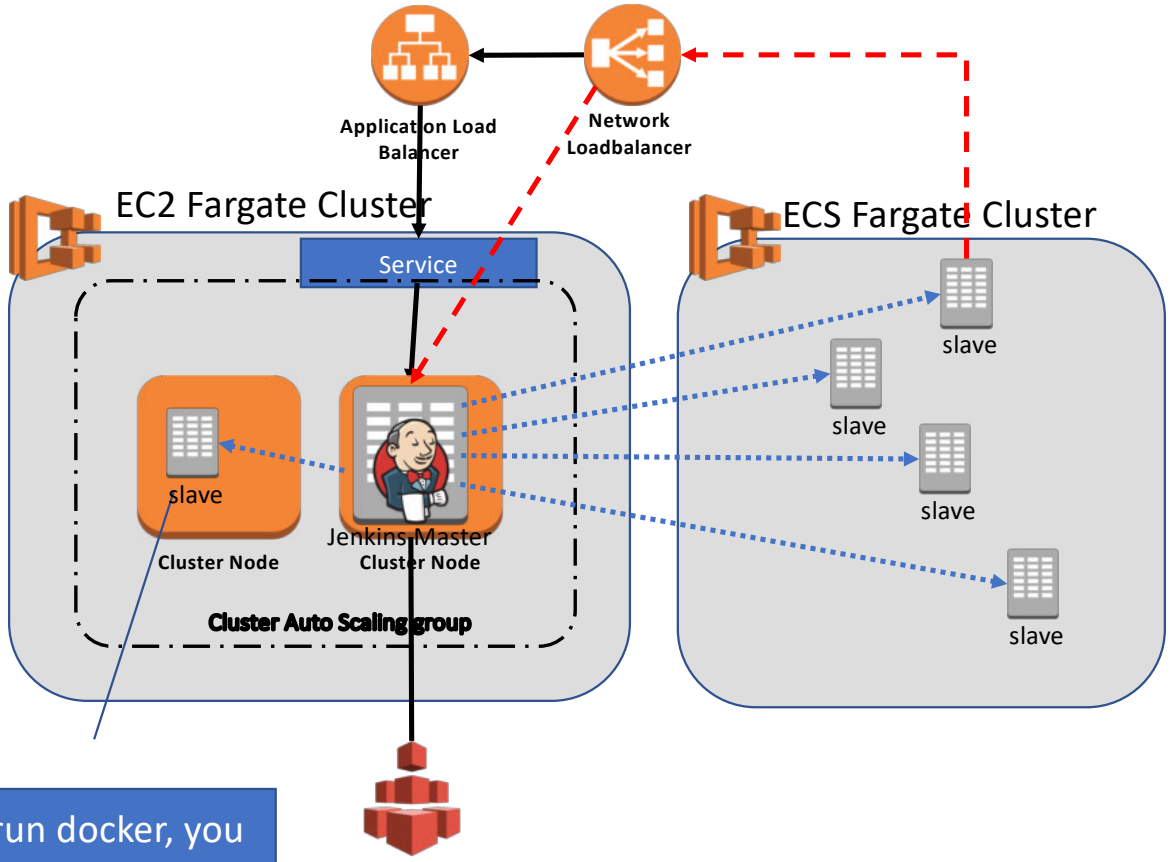

EC2 Fargate Cluster
Service
Cluster Node
Jenkins Master
Cluster Node
Cluster Auto Scaling group
EFS Volume
(Jenkins config)
EFS Backup Volume
(Jenkins config)
We deployed this solution
for an periodically backup

Fargate does not allow docker in docker





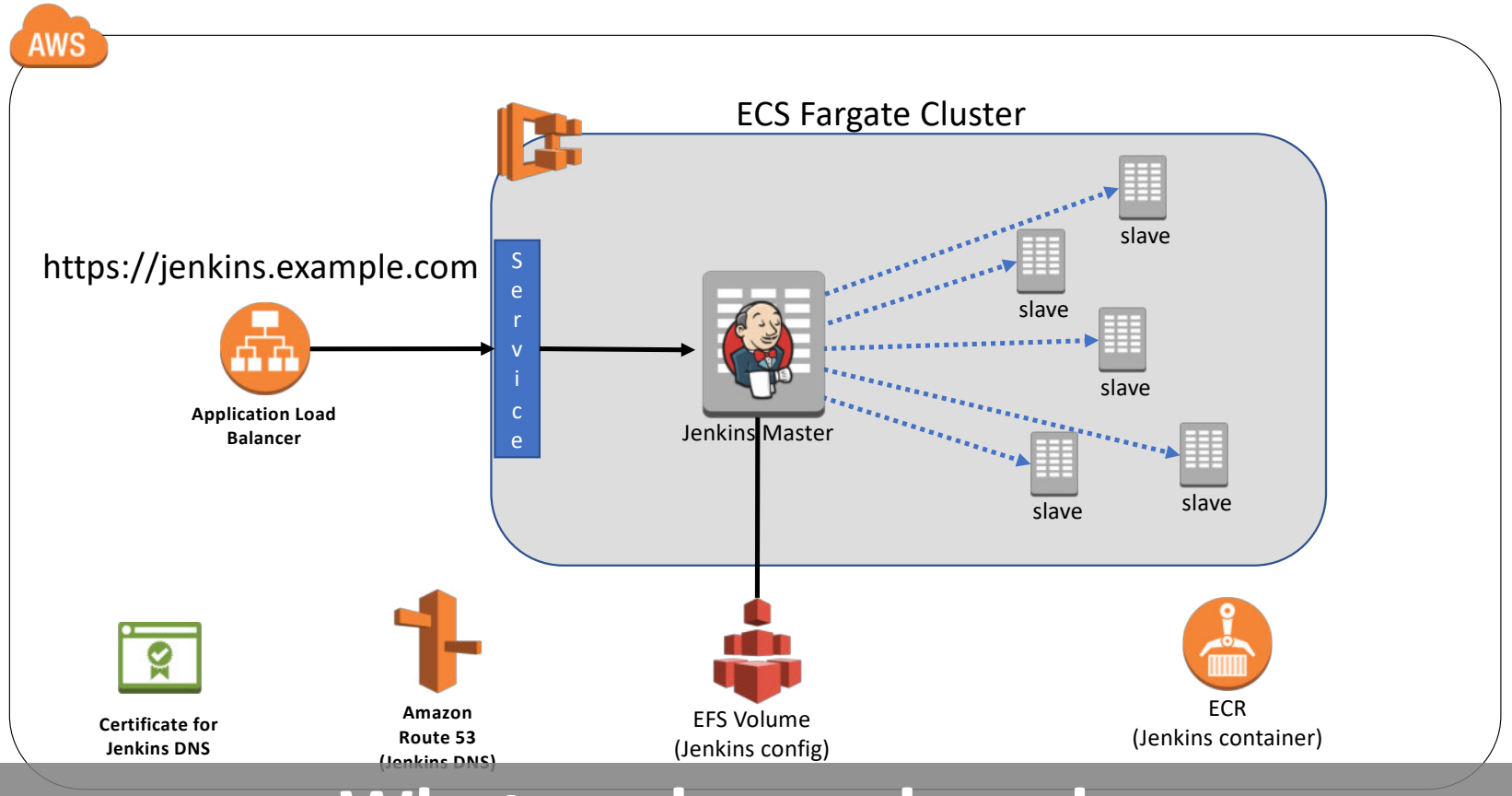
Without the ability to mount volumes, you can not mount the docker socket to an fargate contianer



If you need to build or run docker, you have to run this specific build-step in an task in the EC2 Cluster.

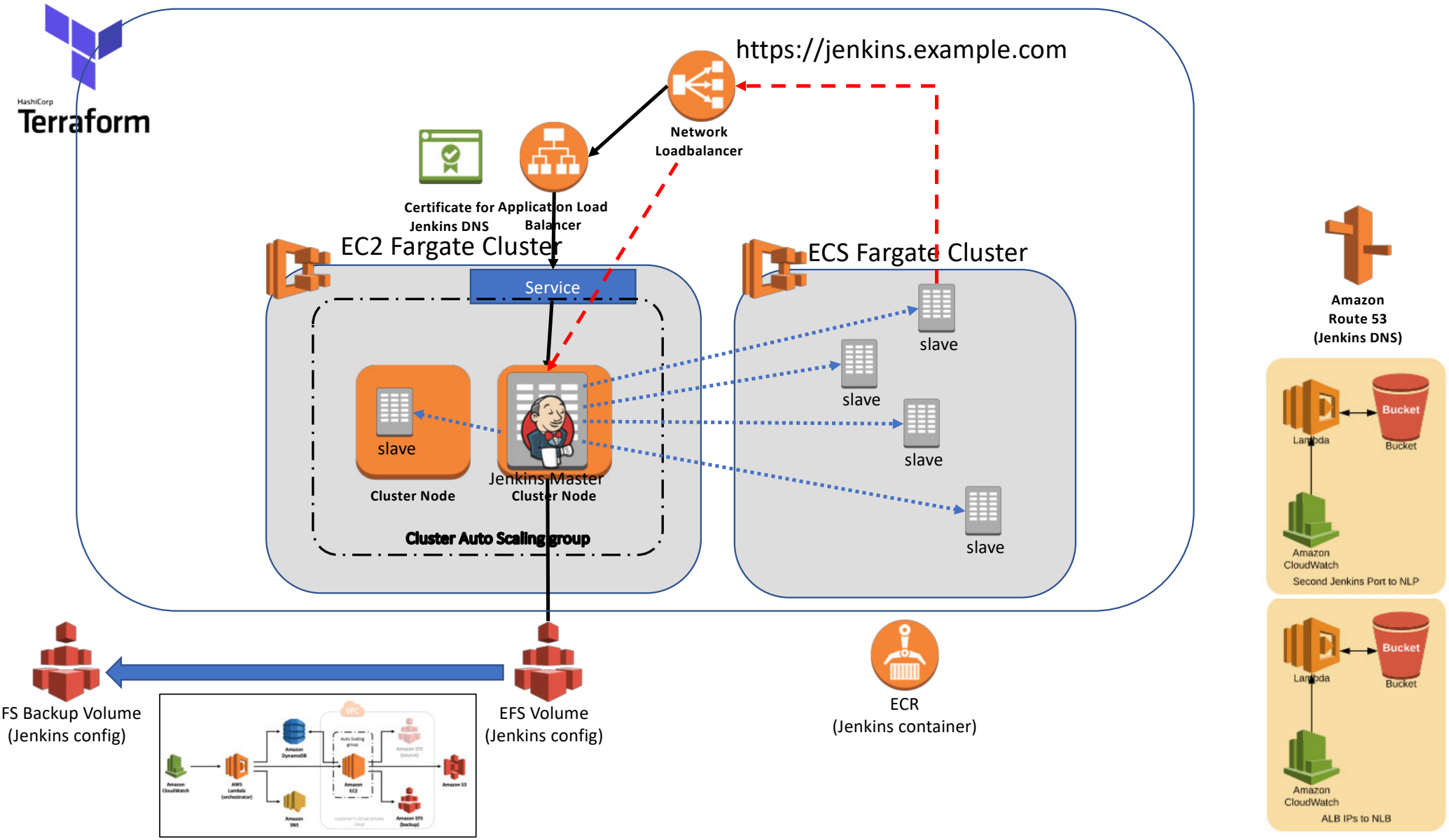

Yes we did it!

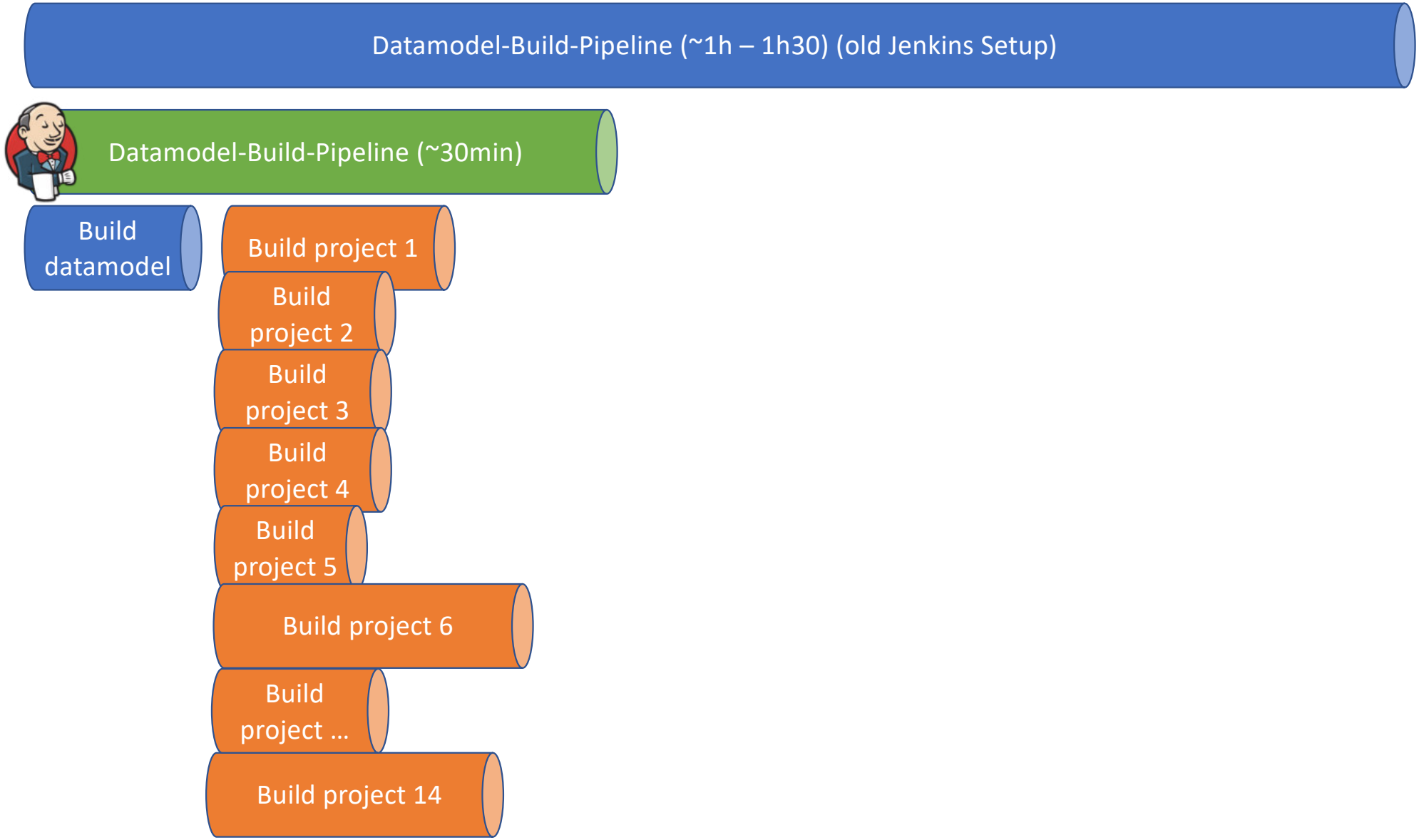

Lets enjoy the view!

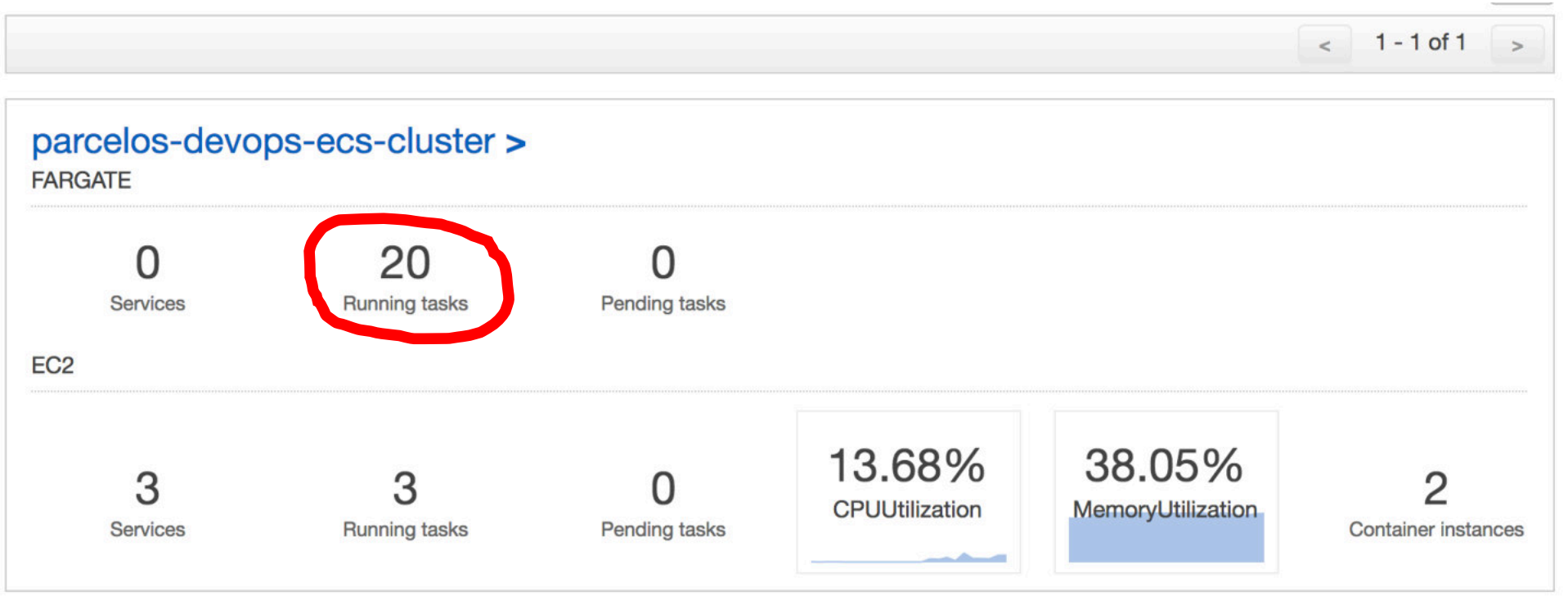


What we have planed...

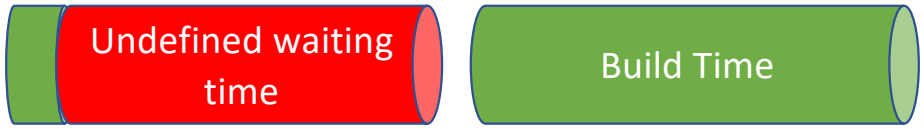
HashiCorp Terraform


HashiCorp
Terraform
https://jenkins.example.com
Network
Loadbalancer
Certificate for
Jenkins DNS
Application Load
Balancer
EC2 Fargate Cluster
ECS Fargate Cluster
Service
slave
Cluster Node
Jenkins Master
Cluster Node
Cluster Auto Scaling group
slave
slave
slave
slave
Amazon
Route 53
(Jenkins DNS)
Bucket
Bucket
Amazon
CloudWatch
Second Jenkins Port to NLP
Bucket
Bucket
Amazon
CloudWatch
ALB IPs to NLB
EFS Backup Volume
(Jenkins config)
EFS Volume
(Jenkins config)
ECR
(Jenkins container)


Datamodel-Build-Pipeline (~1h – 1h30) (old Jenkins Setup)
Datamodel-Build-Pipeline (~30min)
Build datamodel
Build project 1
Build project 2
Build project 3
Build project 4
Build project 5
Build project 6
Build project …
Build project 14


< 1 - 1 of 1 >
parcelos-devops-ecs-cluster >
FARGATE
0
Services
20
Running tasks
0
Pending tasks
EC2
3
Services
3
Running tasks
0
Pending tasks
13.68%
CPUUtilization
38.05%
MemoryUtilization
2
Container instances

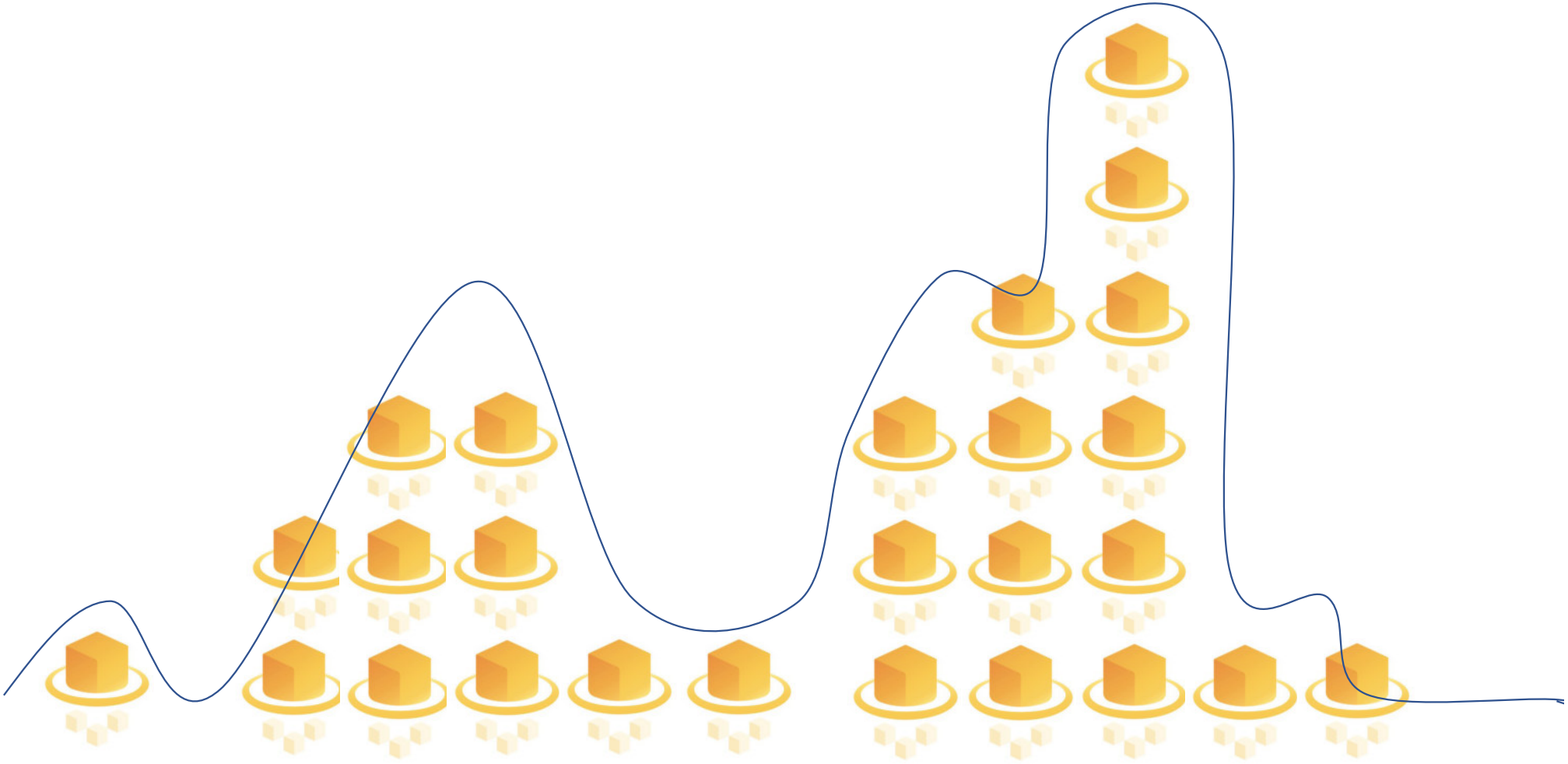
Classic Build system

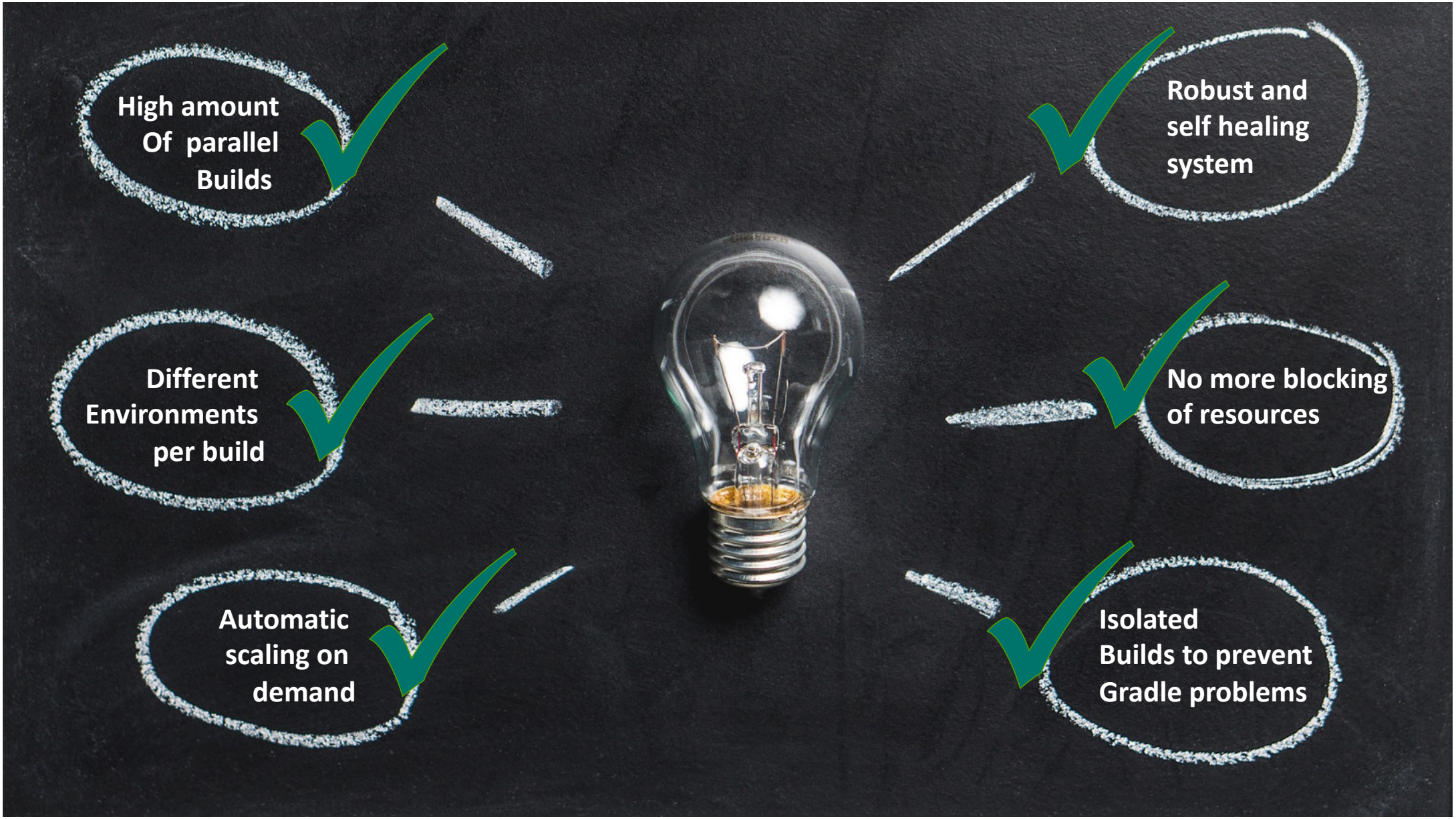


VS

Fargate Build system

Fargate is ideal for dynamic requirements




High amount Of parallel Builds
Robust and self healing system
Different Environments per build
No more blocking of resources
Automatic scaling on demand
Isolated Builds to prevent Gradle problems


It is good but not perfect


Thank you!

Bitte geben Sie uns jetzt Ihr Feedback!

Nie wieder Schlange stehen! Dank Jenkins auf AWS ECS und Fargate

Philipp Koch

Nächste Vorträge in diesem Raum

14:30 Schnelle und zuverlässige Builds mit Gradle und Maven, *Marc Philipp*

15:45 Infrastructure as Code - Jenkins up and running mit Ansible und Docker, *Max Riechelmann, Dr. Stefan Schlott*

16:45 Zusammenarbeitskultur in der digitalen Arbeitswelt – Netzwerkaufbau mit Working Out Loud, *Petra Hock*