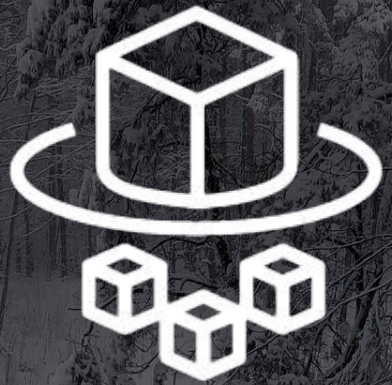


Fargate

Aurelijus Banelis

VilniusPHP 0x63
2021-02-04

Aurelijus Banelis

Senior Software Engineer

aurelijus.banelis.lt
aurelijus@banelis.lt

PGP 0x320205E7539B6203
130D C446 1F1A 2E50 D6E3
3DA8 3202 05E7 539B 6203

.NET

aws certified SysOps Administrator Associate

aws certified Solutions Architect Associate

aws certified Developer Associate

zend CERTIFIED PHP ENGINEER

home 24

Aurelijus Banelis

Senior Software Engineer

aurelijus.banelis.lt
aurelijus@banelis.lt

PGP 0x320205E7539B6203
130D C446 1F1A 2E50 D6E3
3DA8 3202 05E7 539B 6203

.NET

aws certified SysOps Administrator Associate

aws certified Solutions Architect Associate

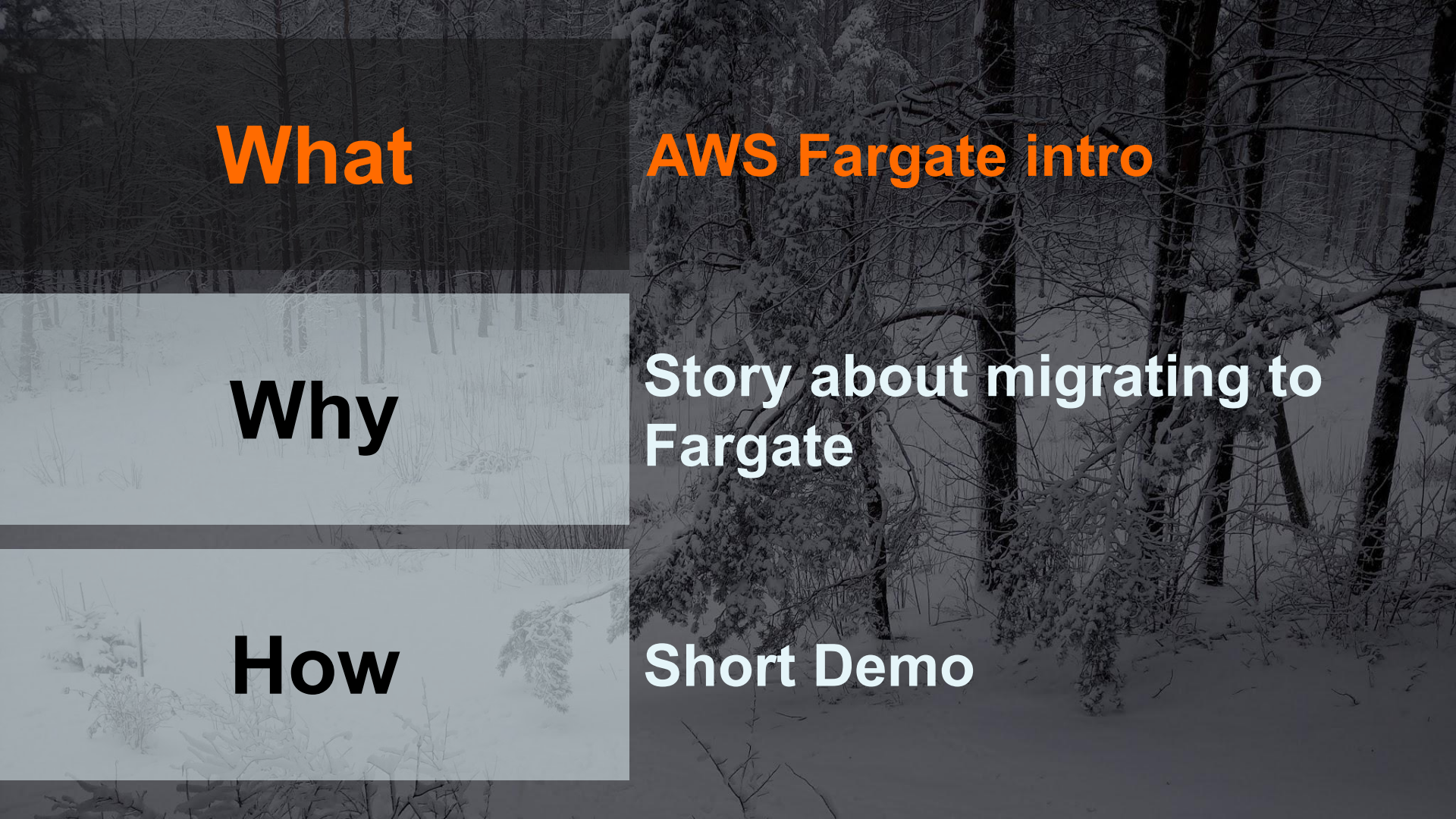
aws certified Developer Associate

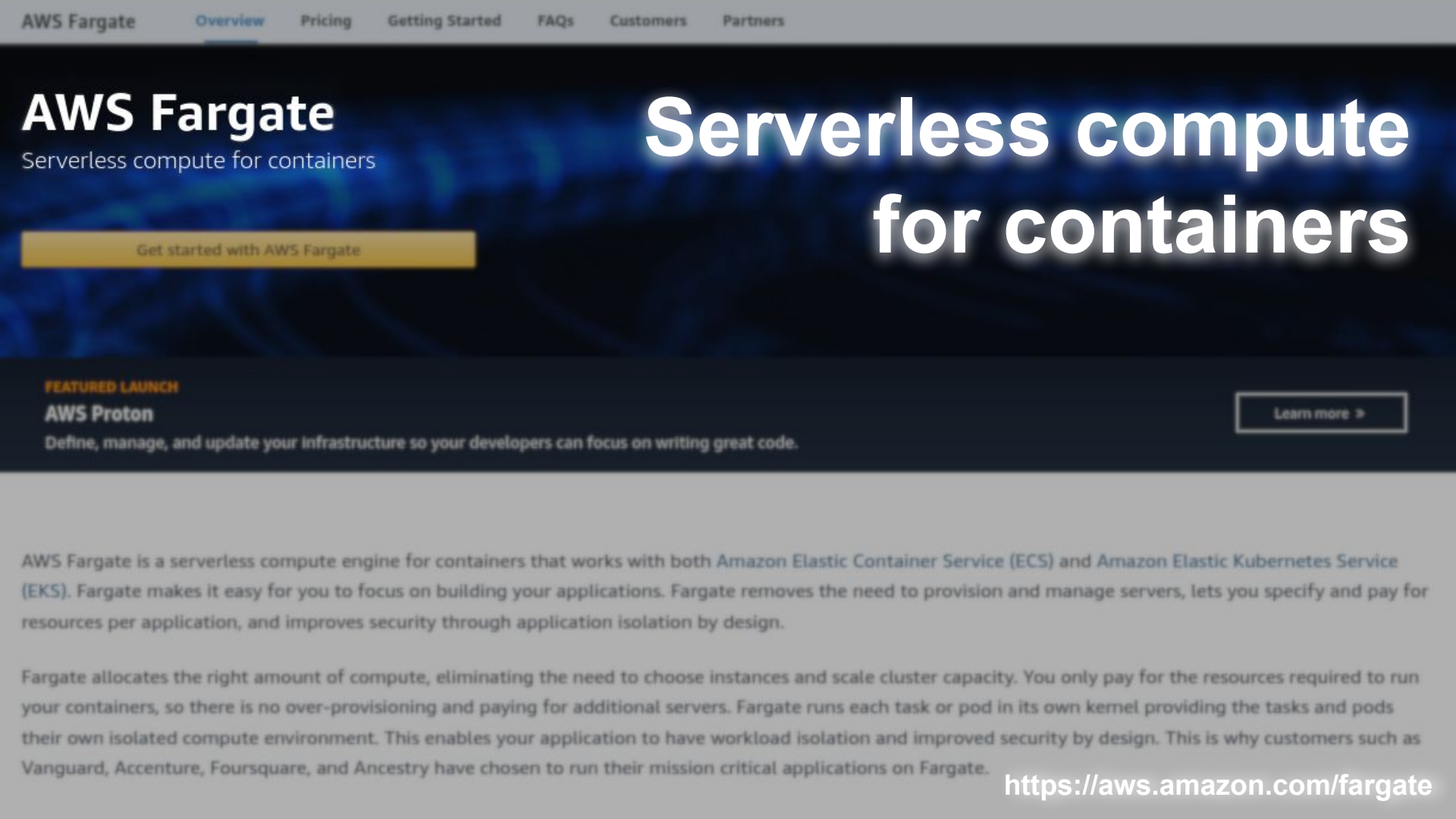
zend CERTIFIED PHP ENGINEER

home 24


Development
with
infrastructure


What
AWS Fargate intro
Why
Story about migrating to Fargate
How
Short Demo


What
AWS Fargate intro
Why
Story about migrating to Fargate
How
Short Demo


AWS Fargate
Overview
Pricing
Getting Started
FAQs
Customers
Partners
AWS Fargate
Serverless compute for containers
Get started with AWS Fargate
Serverless compute for containers
FEATURED LAUNCH
AWS Proton
Define, manage, and update your infrastructure so your developers can focus on writing great code.
Learn more »
AWS Fargate is a serverless compute engine for containers that works with both Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS). Fargate makes it easy for you to focus on building your applications. Fargate removes the need to provision and manage servers, lets you specify and pay for resources per application, and improves security through application isolation by design.
Fargate allocates the right amount of compute, eliminating the need to choose instances and scale cluster capacity. You only pay for the resources required to run your containers, so there is no over-provisioning and paying for additional servers. Fargate runs each task or pod in its own kernel providing the tasks and pods their own isolated compute environment. This enables your application to have workload isolation and improved security by design. This is why customers such as Vanguard, Accenture, Foursquare, and Ancestry have chosen to run their mission critical applications on Fargate.
https://aws.amazon.com/fargate

Serverless compute

CloudWatch monitoring
Default Monitoring

FARGATE

0	0	0
Services	Running tasks	Pending tasks

EC2

1	3	0	%	%	2
Services	Running tasks	Pending tasks	CPUUtilization	MemoryUtilization	Container instances

CloudWatch monitoring
Default Monitoring

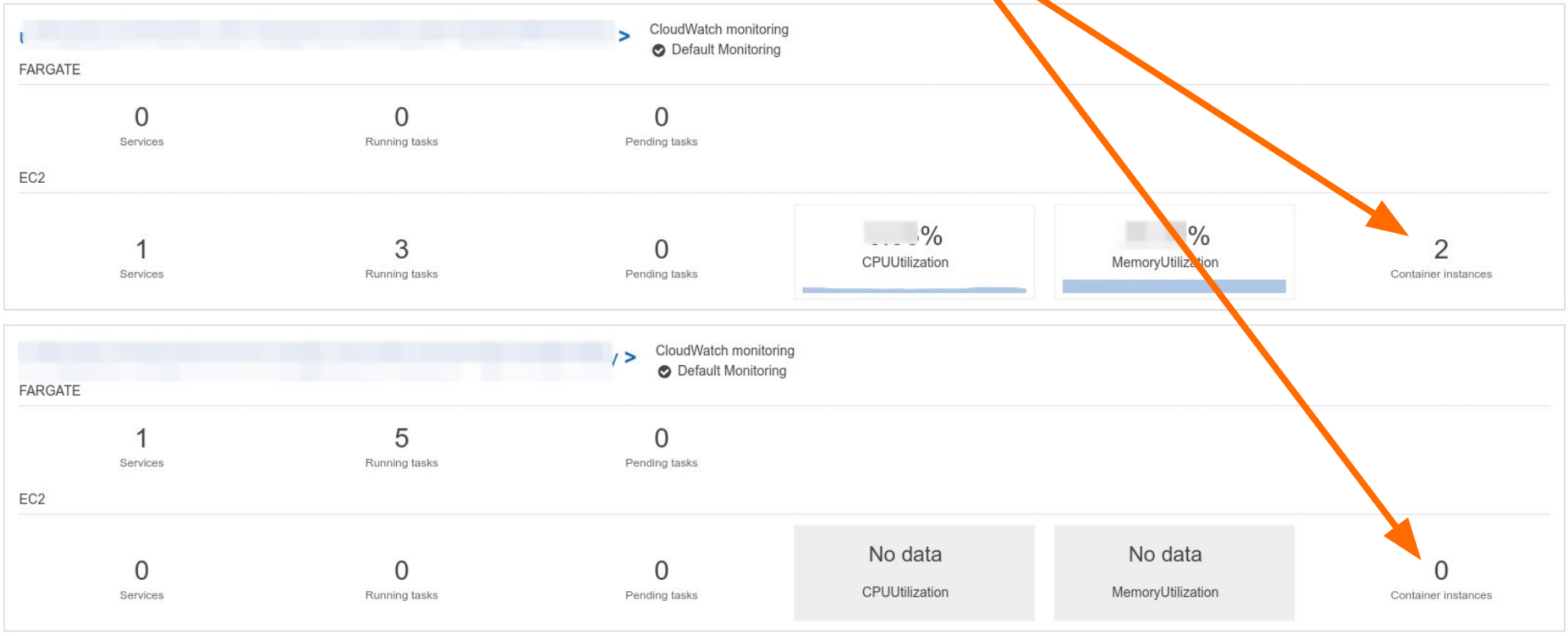
FARGATE

1	5	0
Services	Running tasks	Pending tasks

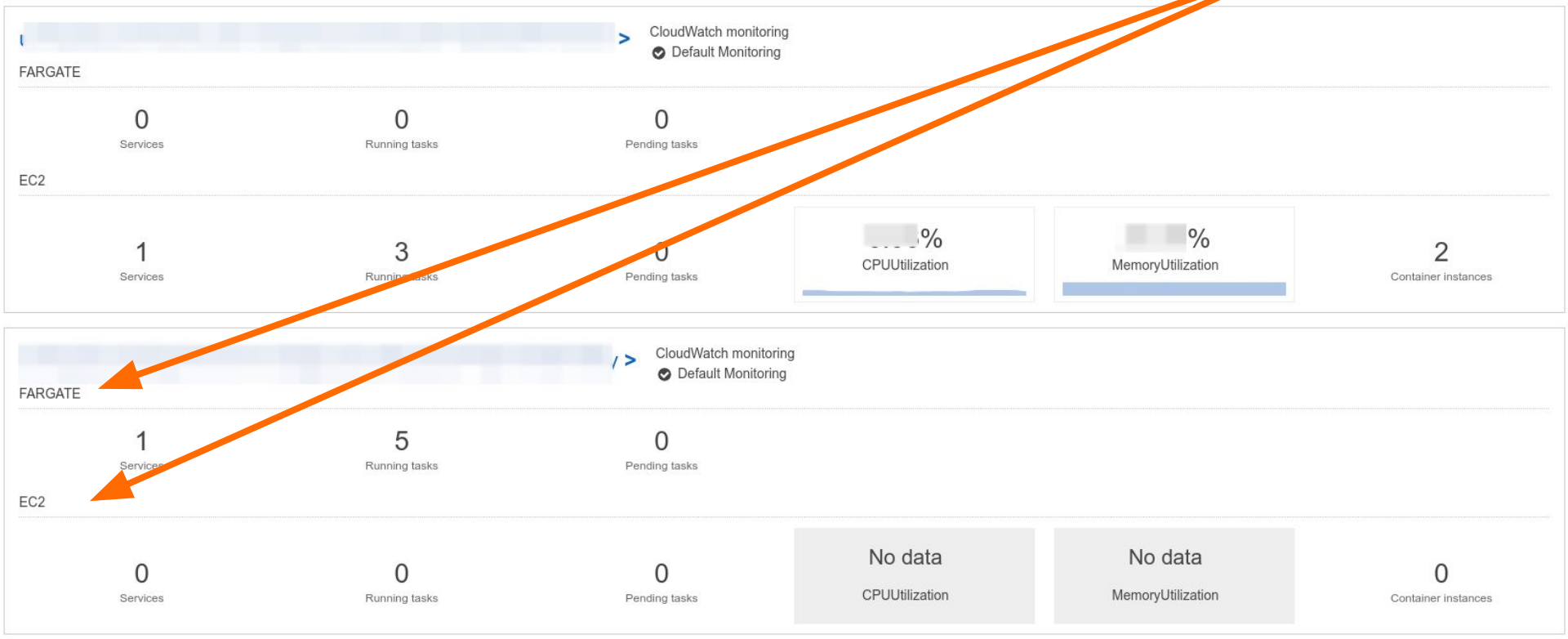
EC2

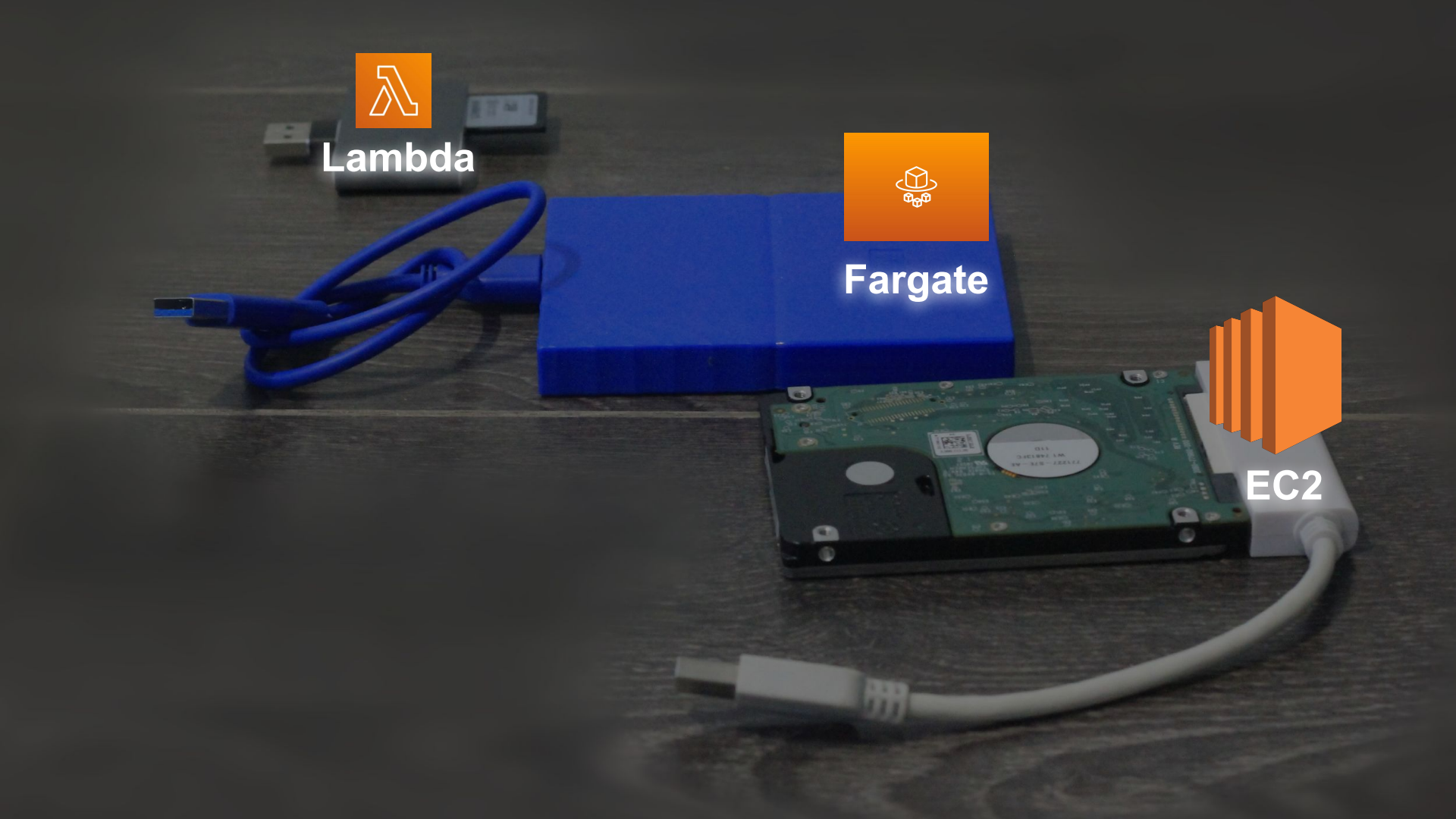
0	0	0	No data	No data	0
Services	Running tasks	Pending tasks	CPUUtilization	MemoryUtilization	Container instances

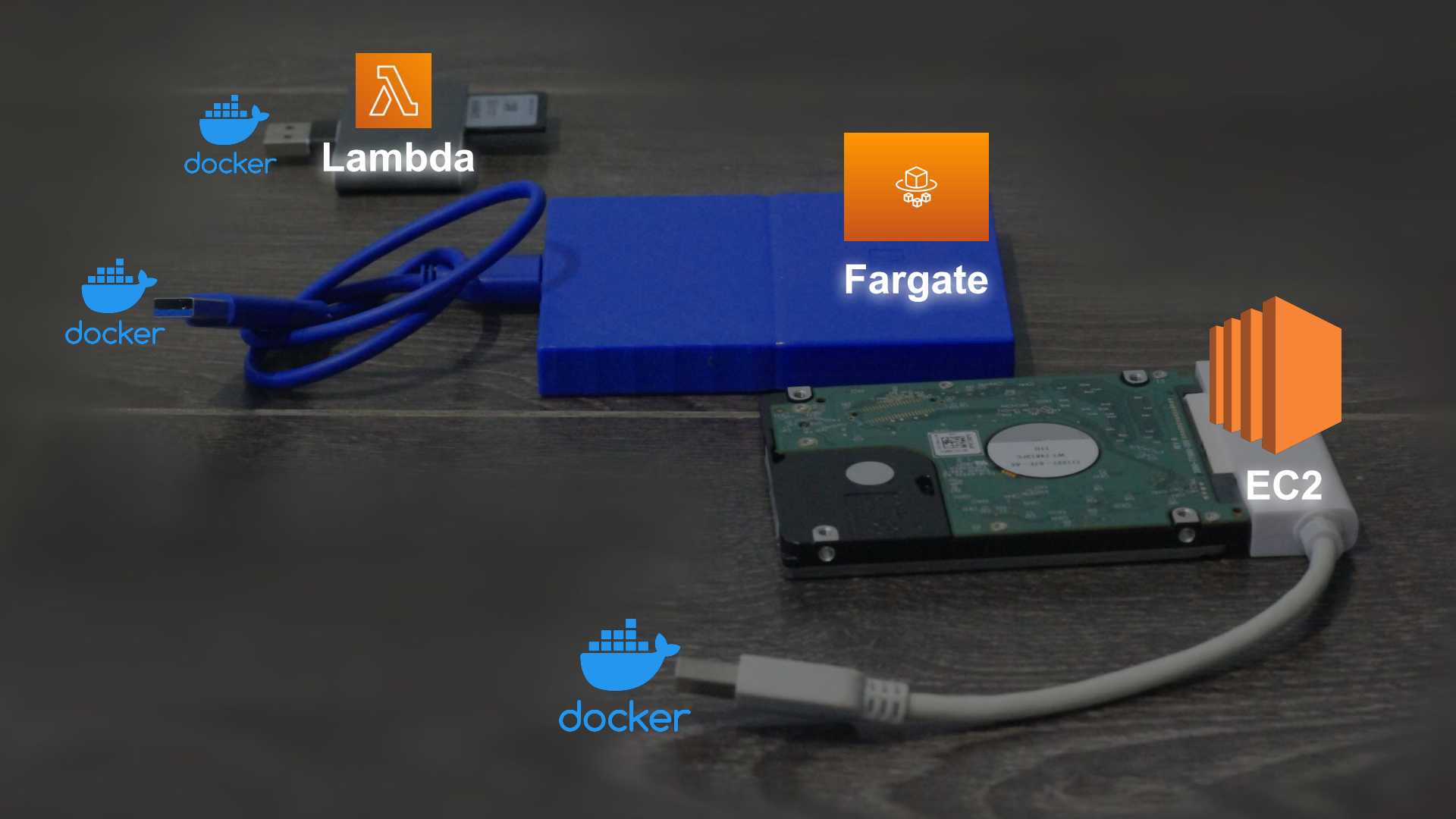
Serverless compute



Serverless compute



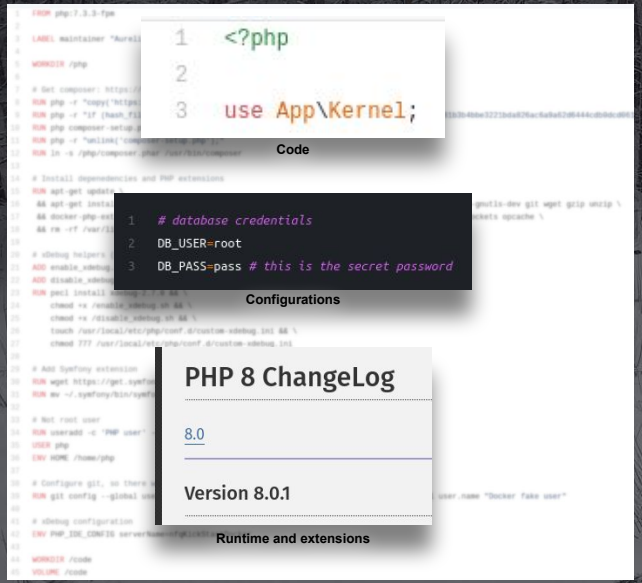

Lambda
Fargate
EC2


docker
Lambda
docker
Fargate
EC2
docker

For containers

```
FROM php:7.3.3-fpm

LABEL maintainer "Aurelijus Banelis <aurelijus@banelis.lt>"

WORKDIR /php

# Get composer: https://getcomposer.org/download/
RUN php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
RUN php -r "if (hash_file('sha384', 'composer-setup.php') === '48e3236262b34d30969dca3c37281b3b4bbe3221bda826ac6a9a62d6444cdb0dcd0615ac06b7ed7
RUN php composer-setup.php
RUN php -r "unlink('composer-setup.php');"
RUN ln -s /php/composer.phar /usr/bin/composer

# Install dependencies and PHP extensions
RUN apt-get update \
 && apt-get install -y librip-dev bash-completion procps nano libicu-dev g++ libpng-dev libcurl4-gnutls-dev git wget gzip unzip \
 && docker-php-ext-install -j$(nproc) zip mysqli pdo_mysql intl calendar exif gd gettext pcntl sockets opcache \
 && rm -rf /var/lib/apt/lists/*

# xDebug helpers (do not use this in real production)
ADD enable_xdebug.sh /enable_xdebug.sh
ADD disable_xdebug.sh /disable_xdebug.sh
RUN pecl install xdebug-2.7.0 && \
    chmod +x /enable_xdebug.sh && \
    chmod +x /disable_xdebug.sh && \
    touch /usr/local/etc/php/conf.d/custom-xdebug.ini && \
    chmod 777 /usr/local/etc/php/conf.d/custom-xdebug.ini

# Add Symfony extension
RUN wget https://get.symfony.com/cli/installer -O - | bash
RUN mv ~/.symfony/bin/symfony /usr/local/bin/symfony

# Not root user
RUN useradd -c 'PHP user' -m -d /home/php -s /bin/bash php
USER php
ENV HOME /home/php

# Configure git, so there would not be errors running "symfony new my_project"
RUN git config --global user.email "docker.fake.user@example.com" && git config --global user.name "Docker fake user"

# xDebug configuration
ENV PHP_IDE_CONFIG serverName=nfqKickStartDocker

WORKDIR /code
VOLUME /code
```

For containers



For containers

docker

For containers

```php
<?php

use App\Kernel;
```

Code

```
# database credentials
DB_USER=root
DB_PASS=pass # this is the secret password
```

PHP 8 ChangeLog

8.0

Version 8.0.1

Runtime and extensions

Run the same on

Azure VM

AWS EC2

Google Compute Engine

For containers

docker

Easier on

VM
Azure VM

Azure Container Instances

AWS EC2

AWS Fargate

Google Compute Engine

Google Cloud Run

Serverless compute for containers

docker

Via container orchestration

VM
Azure VM

Azure Container Instances

AKS

AWS EC2

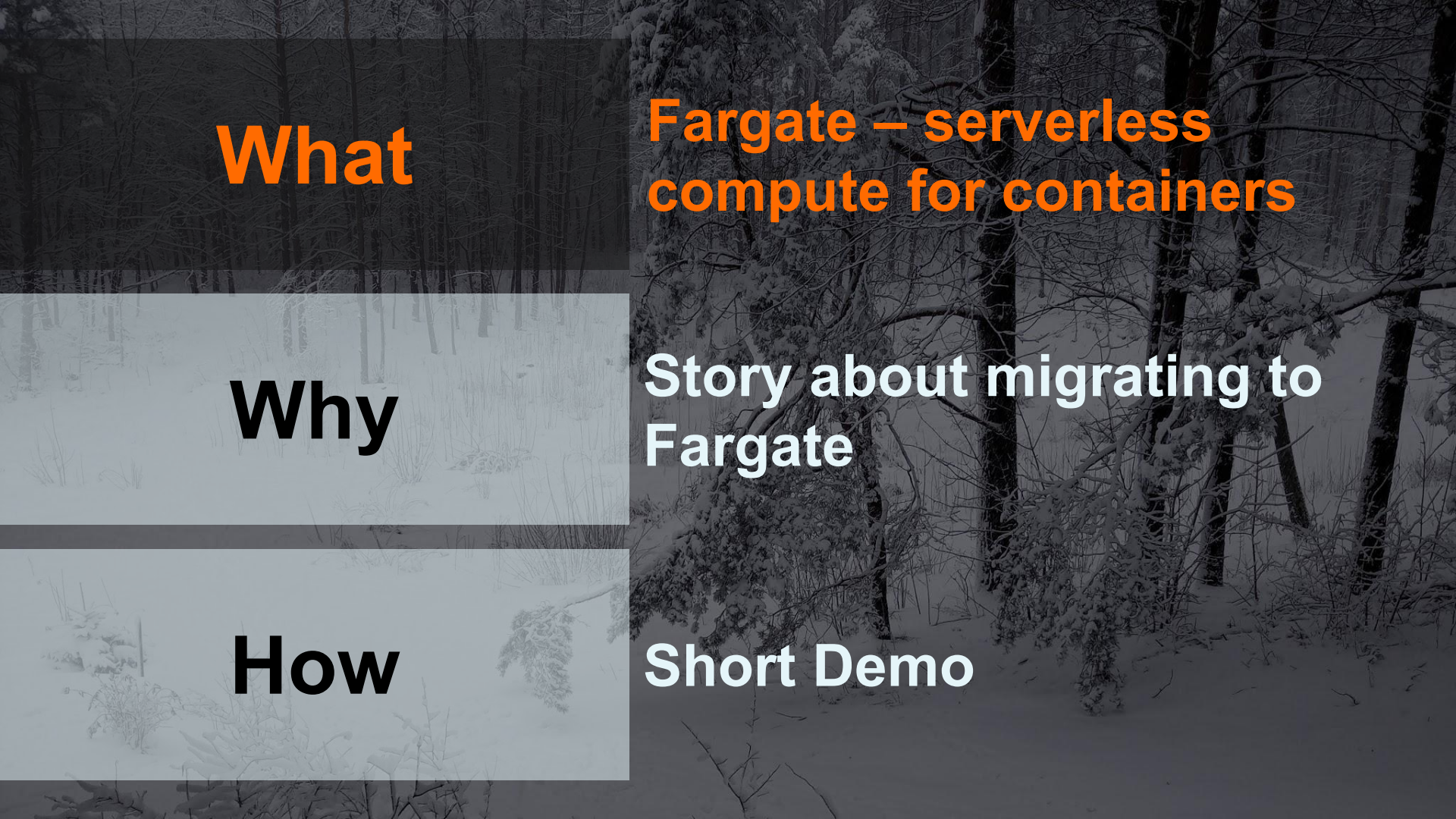
EC2 / Fargate
x
ECS / EKS

Google Compute Engine

Google Cloud Run

Google Kubernetes

Think just as an
alternative hardware


What
Fargate – serverless compute for containers
Why
Story about migrating to Fargate
How
Short Demo


What
Fargate – serverless compute for containers
Why
Story about migrating to Fargate
How
Short Demo

Introducing AWS Fargate

Posted On: Nov 29, 2017

AWS Fargate is a compute engine for deploying and managing containers without having to manage any of the underlying infrastructure. Fargate makes it easy to scale your applications. You no longer have to worry about provisioning enough compute resources for your container applications. You can launch tens or tens of thousands of containers in seconds.

Previously, you needed to manage a cluster of Amazon EC2 instances, pick the instance types, manage the scheduling of the containers, and optimize cluster utilization. With Fargate, all of this goes away. Fargate seamlessly integrates with Amazon ECS. You just define your application as you do today for Amazon ECS. You package your application into task definitions, specify the CPU and memory needed, define the networking and IAM policies each container needs. Once everything is setup, Fargate launches and manages your containers for you.

With Fargate, billing is at a per second granularity and you only pay for what you use. You pay for the amount of vCPU and memory resources your containerized application requests. vCPU and memory resources are calculated from the time your container images are pulled until the Amazon ECS Task terminates, rounded up to the nearest second.

AWS Fargate is available with Amazon ECS and support for Amazon EKS will be available in 2018. Fargate is available in US East (N. Virginia) with other regions coming soon.

Visit the detail page to learn more.

AWS Fargate launches platform version 1.4.0

by Massimo Re Ferre | on 08 APR 2020 | in Amazon Elastic Container Service, Amazon Elastic Kubernetes Service, AWS Fargate, Containers | Permalink | Comments | Share

AWS Fargate is a managed service to run containers. Fargate allows customers to use Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS) to launch applications without the burden of having to deal with the undifferentiated heavy lifting of maintaining, patching, scaling, securing, life-cycling the infrastructure. While Amazon EC2 abstracts away hypervisors and physical servers from customers, AWS Fargate does the same for container runtimes and EC2 instances. If you want to read more about the role of Fargate in the container world, check out this blog post.

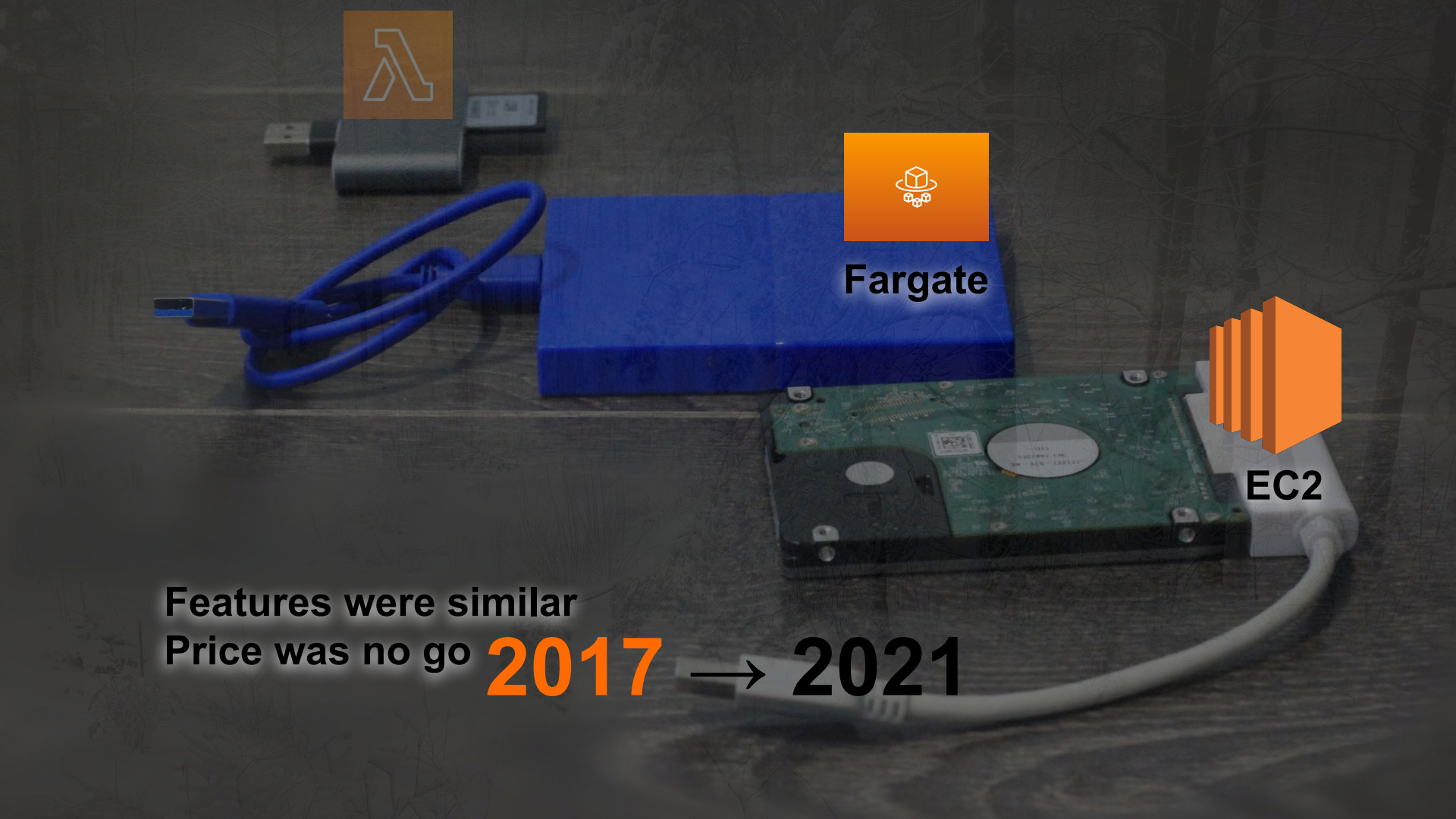
While Fargate makes the infrastructure disappear in the sense that the customer doesn't need to think about it, the infrastructure still exists and it's being managed by AWS. The way the infrastructure features surface to the end users today is through the notion of a Fargate *platform version*. You can read more about it in the Fargate documentation or you can read the Fargate platform versions primer blog post. The primer blog post goes into more detail about the philosophy behind why we introduced Fargate platform versions and, for example, the practical reasons why we are not tagging platform version `1.4.0` as LATEST just yet.

Today we are launching platform version 1.4.0 of AWS Fargate.

In this blog post, we are going to provide you with a summary of the Fargate features we are enabling with this release and some of the changes we are making underneath. These underlying changes don't necessarily have a direct relationship with the new customer-visible features but they are just as important.

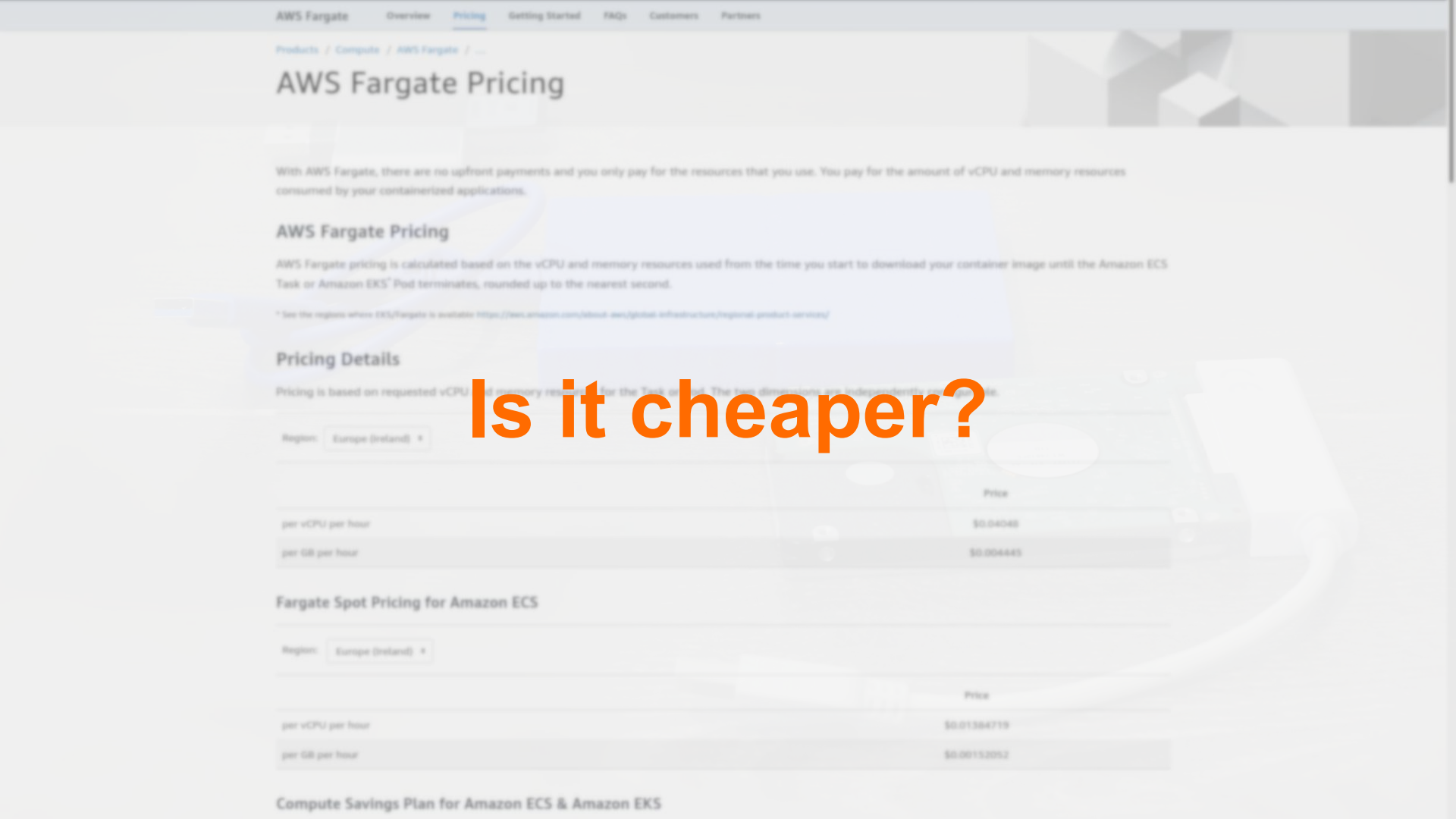
What's new in Fargate platform version 1.4.0?

2017 → 2021

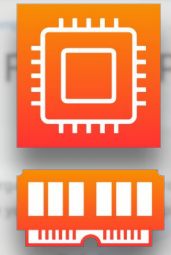


Features were similar
Price was no go

2017 → 2021



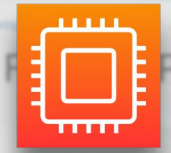
Is it cheaper?



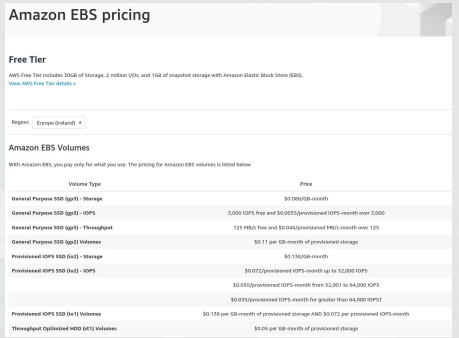
vCPU	Memory	Instance	EC2: On dema	EC2: Reserve	Fargate: Defa	On demand	Reserved
2	4	c5.large	0.096	0.048	0.09874	-97.23%	-91.72%
2	8	m4.large	0.111	0.0564	0.11652	-95.26%	-91.33%
4	16	m4.xlarge	0.222	0.1127	0.23304	-95.26%	-91.25%
2	8	t2.large	0.1008	0.053	0.11652	-86.51%	-85.82%
2	4	t2.medium	0.05	0.0265	0.09874	-50.64%	-50.64%
1	2	t2.small	0.025	0.0132	0.04937	-50.64%	-50.45%
4	16	t2.xlarge	0.2016	0.1059	0.23304	-86.51%	-85.74%

Is it cheaper?

Yes and no

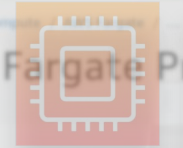


vCPU	Memory	Instance	EC2: On dema	EC2: Reserve	Fargate: Defa	On demand	Reserved
2	4	c5.large	0.096	0.048	0.09874	-97.23%	-91.72%
2	8	m4.large	0.111	0.0564	0.11652	-95.26%	-91.33%
4	16	m4.xlarge	0.222	0.1127	0.23304	-95.26%	-91.25%
2	8	t2.large	0.1008	0.053	0.11652	-86.51%	-85.82%
2	4	t2.medium	0.05	0.0265	0.09874	-50.64%	-50.64%
1	2	t2.small	0.025	0.0132	0.04937	-50.64%	-50.45%
4	16	t2.xlarge	0.2016	0.1059	0.23304	-86.51%	-85.74%

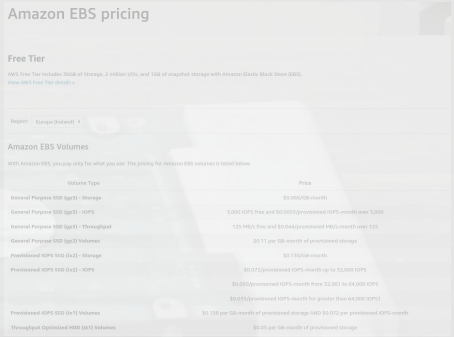


Is it cheaper?

Yes and no



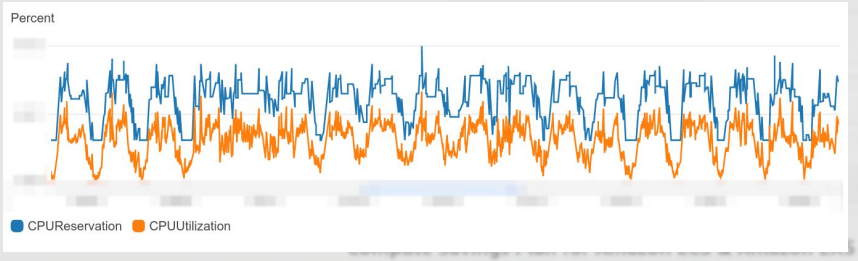
vCPU	Memory	Instance	EC2: On dema	EC2: Reserve	Fargate: Defa	On demand	Reserved
2	4	c5.large	0.096	0.048	0.09874	-97.23%	-91.72%
2	8	m4.large	0.111	0.0564	0.11652	-95.26%	-91.33%
4	16	m4.xlarge	0.222	0.1127	0.23304	-95.26%	-91.25%
2	8	t2.large	0.1008	0.053	0.11652	-86.51%	-85.82%
2	4	t2.medium	0.05	0.0265	0.09874	-50.64%	-50.64%
1	2	t2.small	0.025	0.0132	0.04937	-50.64%	-50.45%
4	16	t2.xlarge	0.2016	0.1059	0.23304	-86.51%	-85.74%



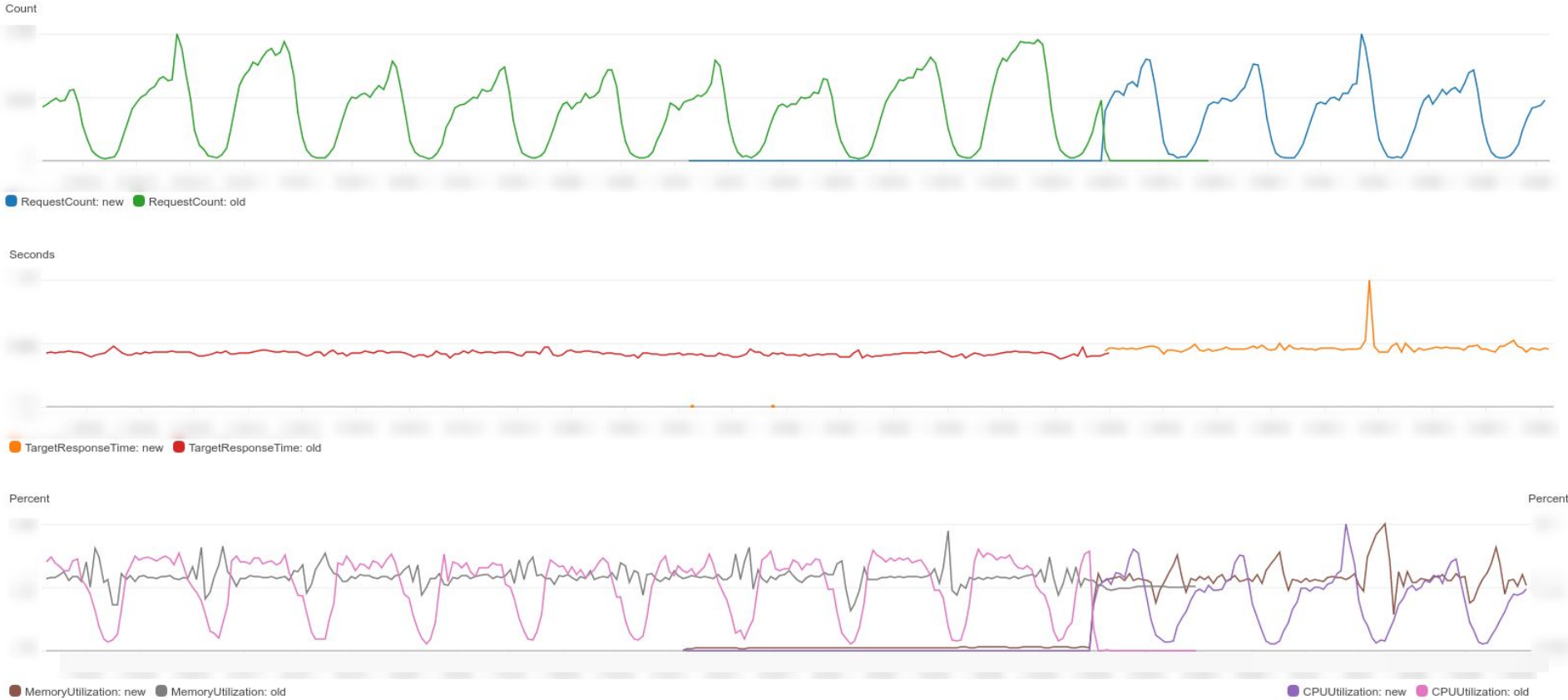
Is it cheaper?

Yes and no

Because overprovisioning



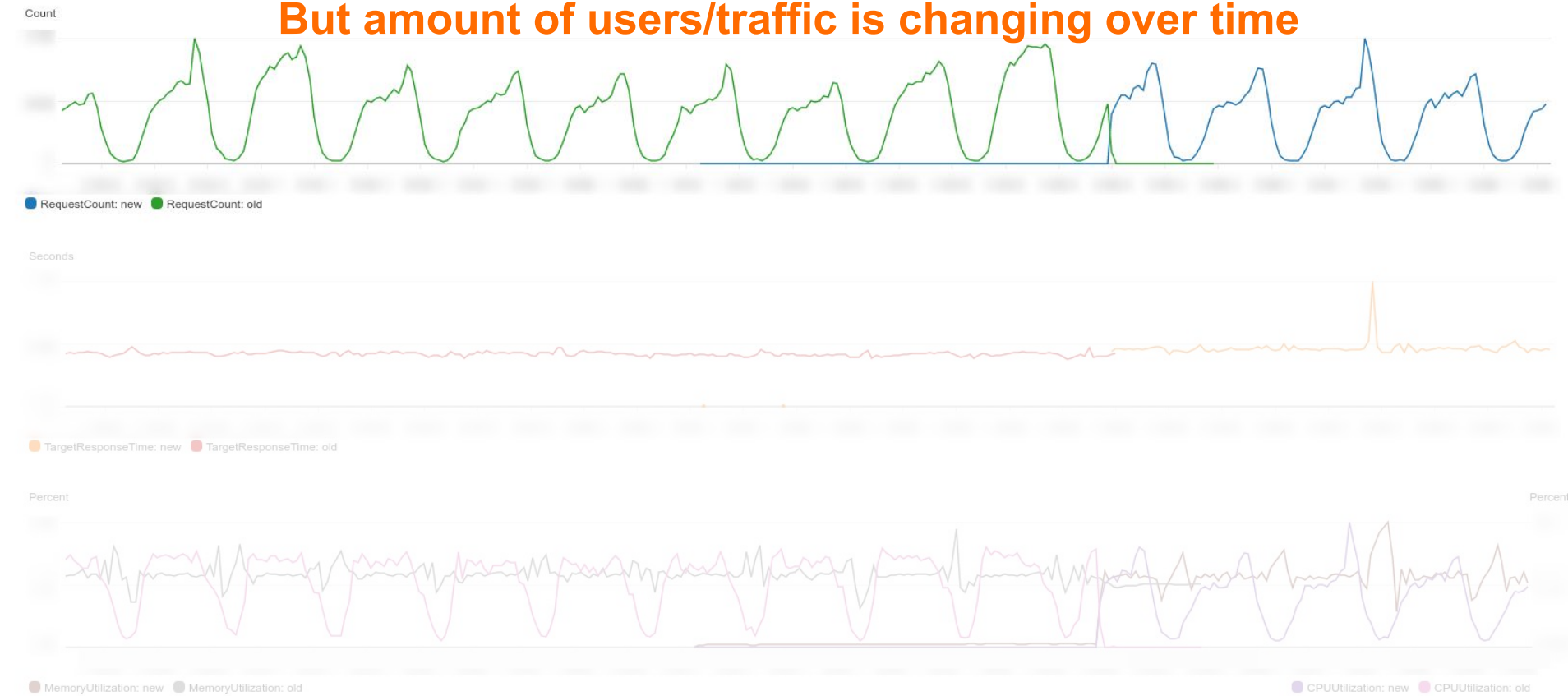
Overprovisioning?



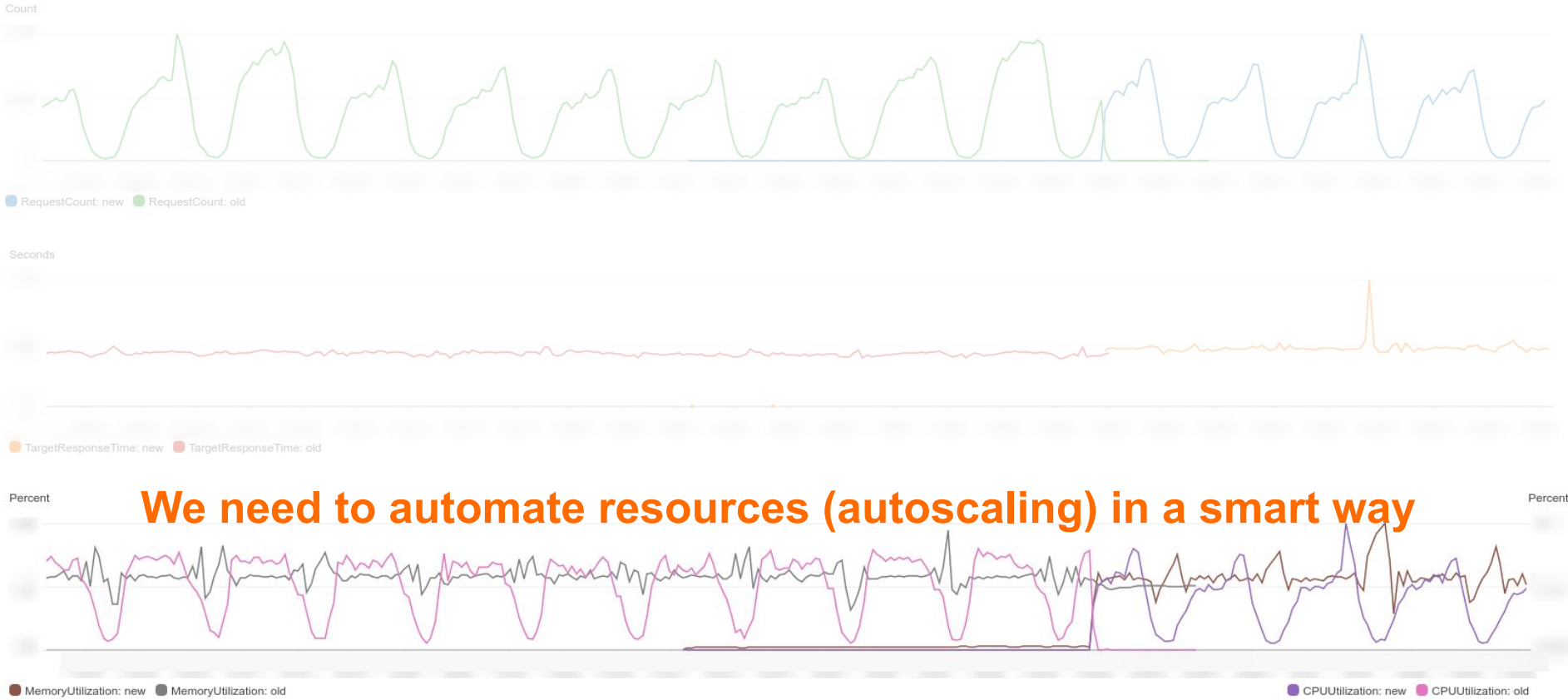
Overprovisioning?

Overprovisioning?

But amount of users/traffic is changing over time

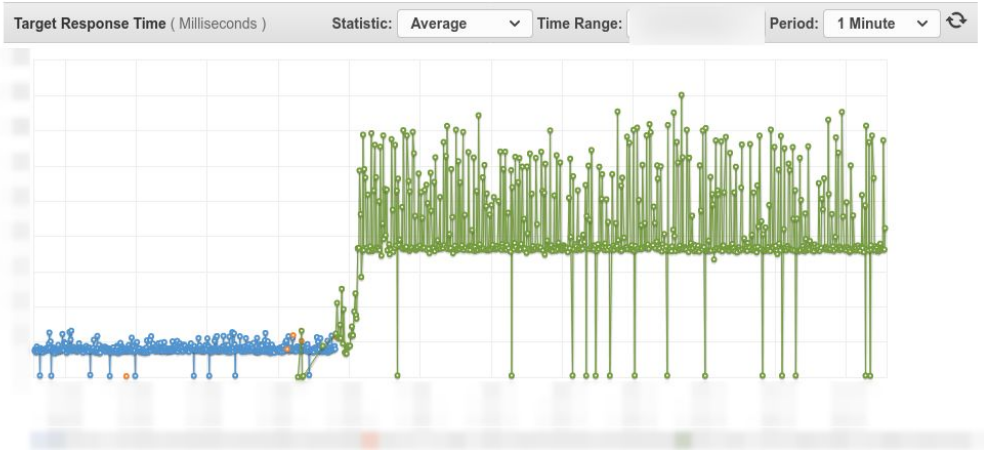


Overprovisioning?

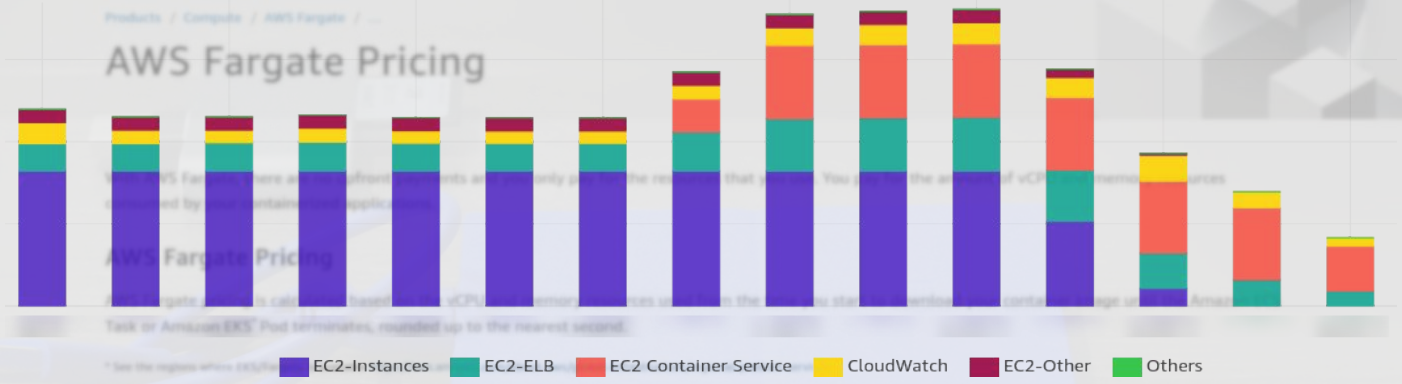


We need to automate resources (autoscaling) in a smart way

Overprovisioning?



Picking the right number is hard



Is it cheaper?

It can be


AWS Fargate Pricing
AWS Fargate Pricing
Pricing Details
Is it cheaper?
Containers
It can be
Containers & instance
← Automating scaling of ⇒


AWS Fargate Pricing
AWS Fargate Pricing
Pricing Details
Is it cheaper?
Containers
It can be
← Automating scaling of ⇒
Less overprovisioning


What about bad things?

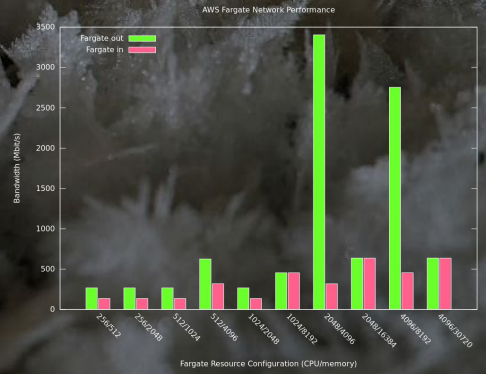
What about bad things?

Costly for small services (<80€ per month)

What about bad things?

Costly for small services (<80€ per month)

Network speed not configurable

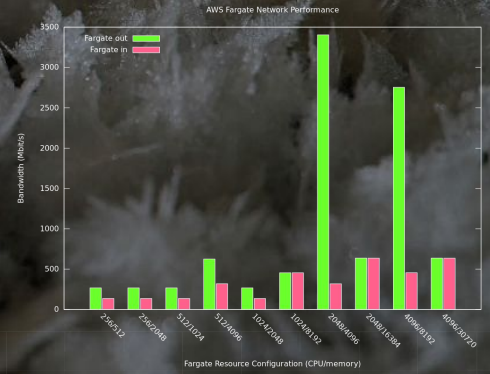


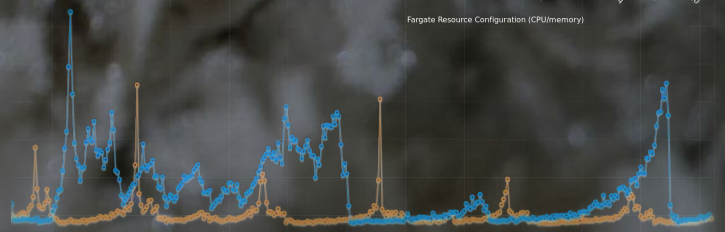
What about bad things?

Costly for small services (<80€ per month)

Network speed not configurable

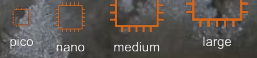
Starts at the same time (for cache invalidation)



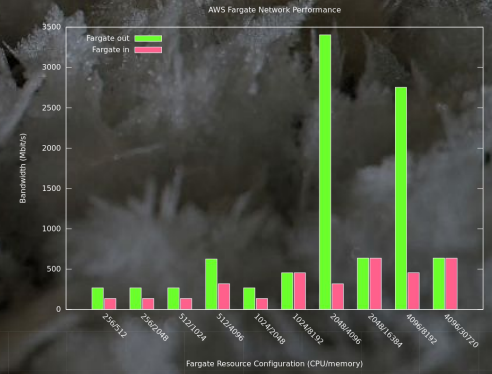


What about bad things?

Costly for small services (<80€ per month)

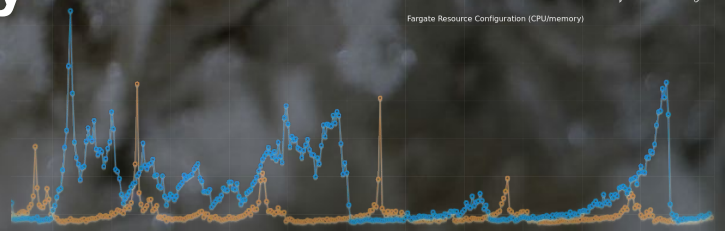


Network speed not configurable



Starts at the same time (for cache invalidation)

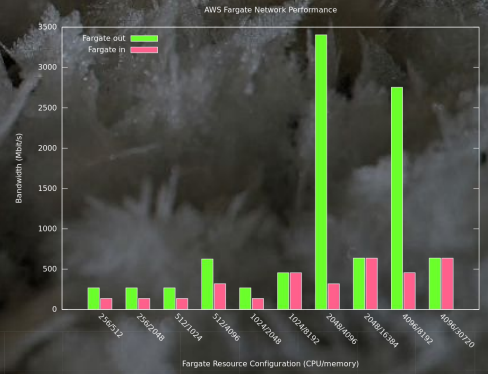
No capacity reservation for Black Friday



What about bad things?

Costly for small services (<80€ per month)

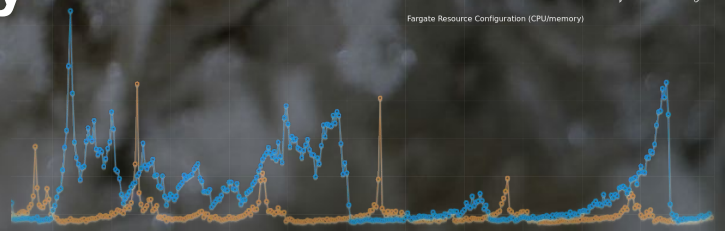
Network speed not configurable



Starts at the same time (for cache invalidation)

No capacity reservation for Black Friday

Other limitations and not battle tested




What
Fargate – serverless compute for containers
Why
Autoscaling (overprovisioning) for spiky traffic
How
Short Demo

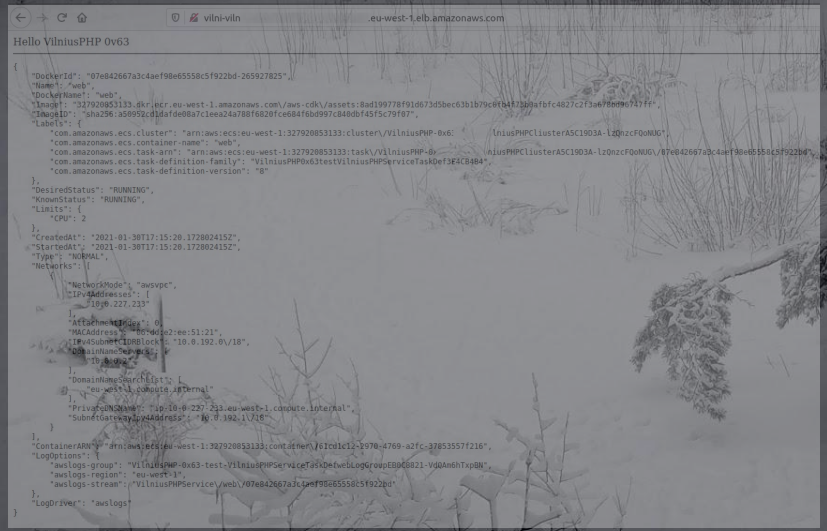

What
Fargate – serverless compute for containers
Why
Autoscaling (overprovisioning) for spiky traffic
How
Short Demo

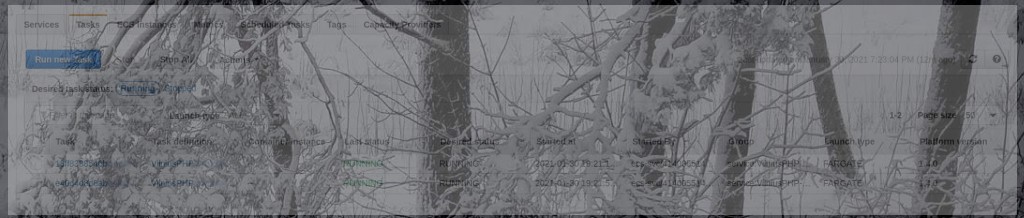
index.php

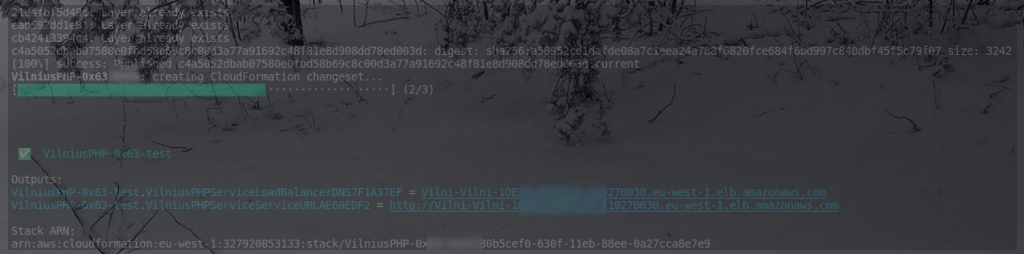
Raw

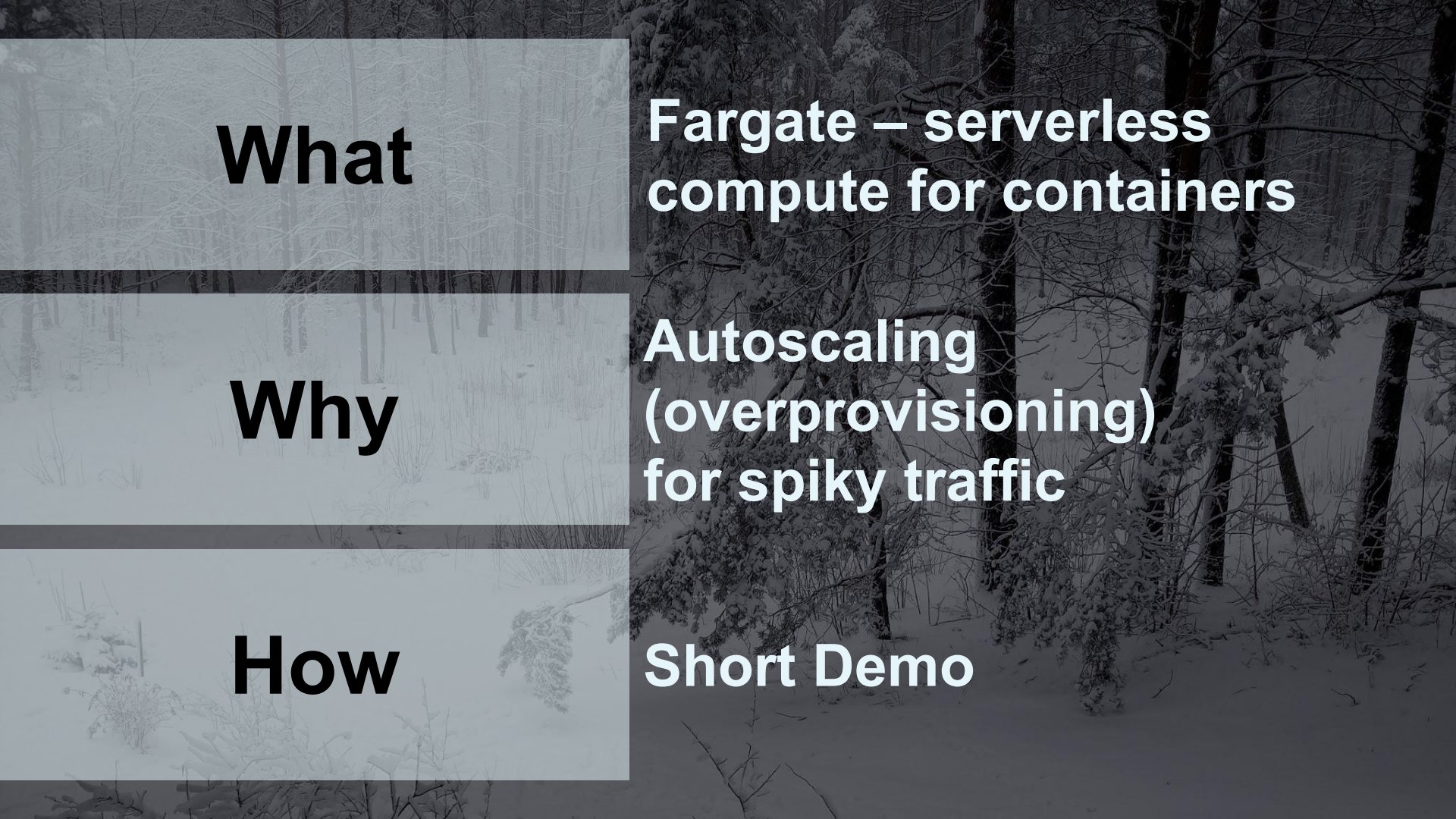
```php
<?php

echo "Hello VilniusPHP 0v63";

$metadata = json_decode(file_get_contents(getenv('ECS_CONTAINER_METADATA_URI_V4')));
echo '<hr/><pre>' . json_encode($metadata, JSON_PRETTY_PRINT) . '</pre>';
```

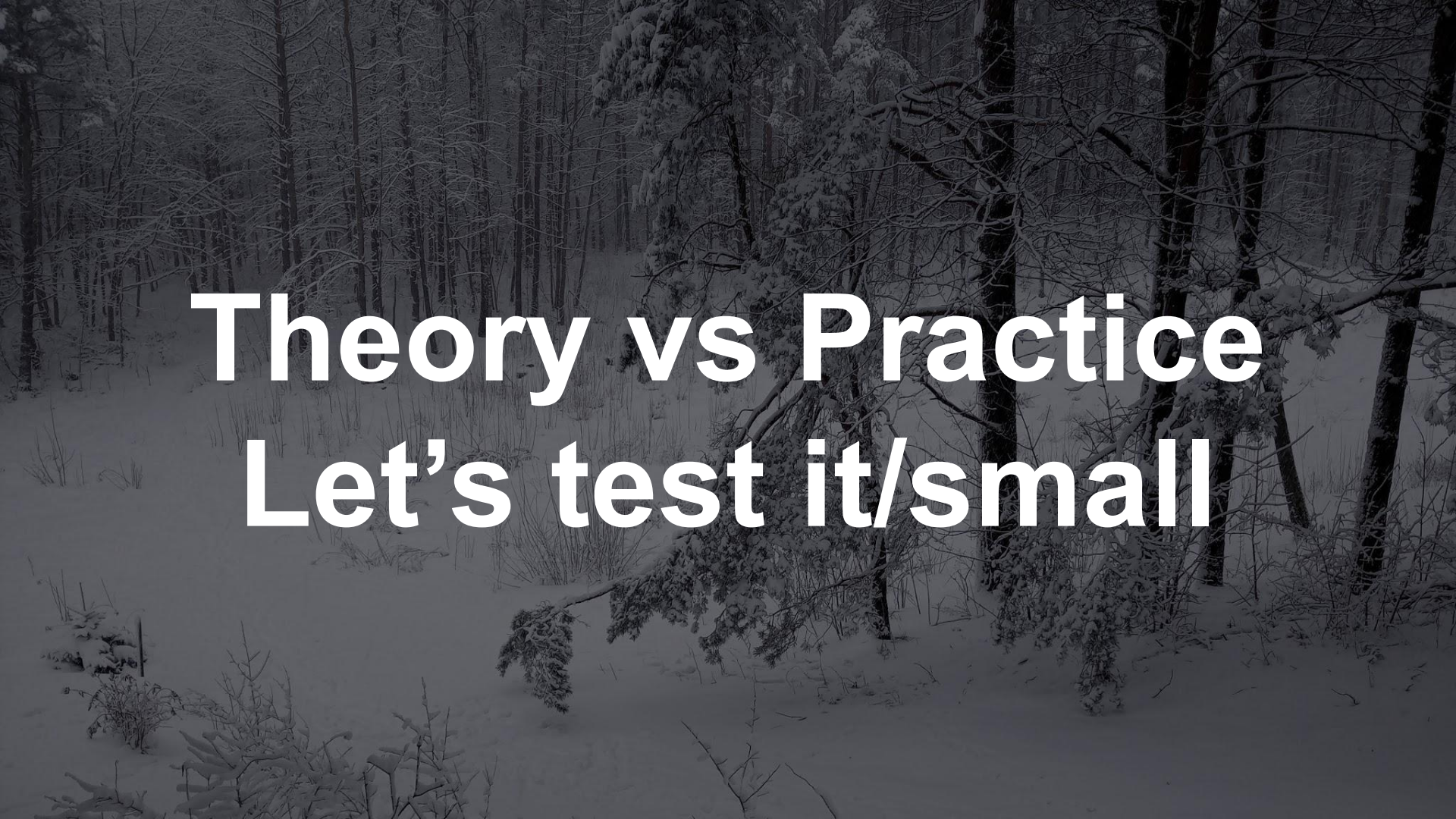
https://gist.github.com/aurelijusb/b3e6ddd6191fe759805efa51f2349135








What
Fargate – serverless compute for containers
Why
Autoscaling (overprovisioning) for spiky traffic
How
Short Demo


Theory vs Practice
Let’s test it/small

Thank you

Fargate

Time for questions

Aurelijus Banelis

VilniusPHP 0x63
2021-02-04