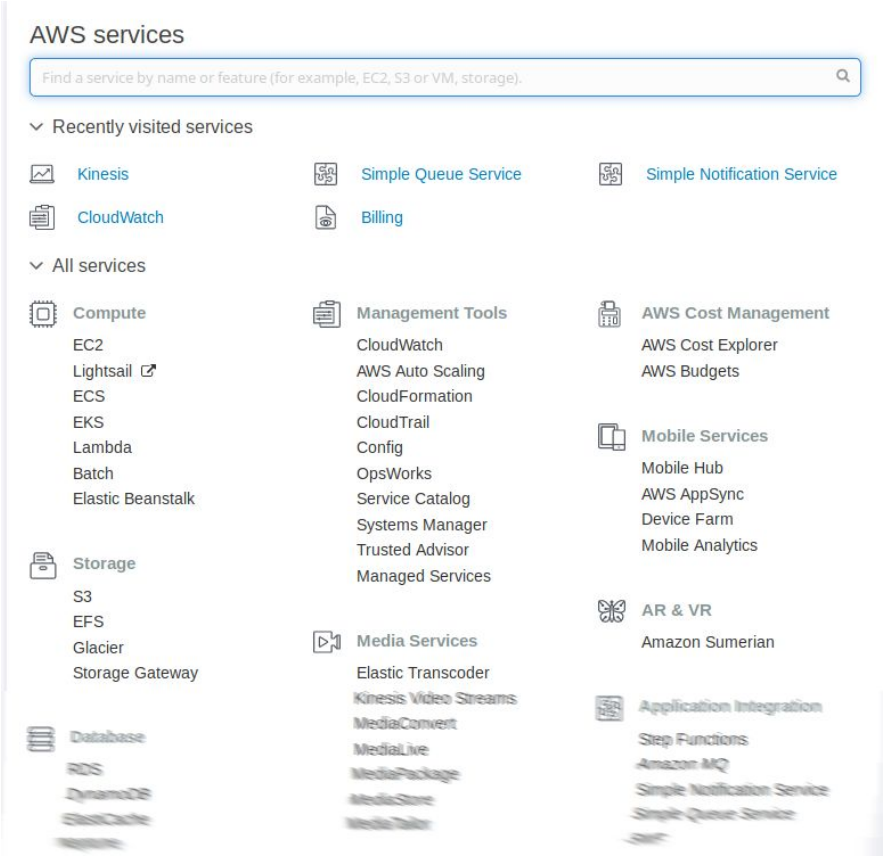

So you want to move
to the AWS Cloud…

The Promise

+

=

The Reality


AWS services
Find a service by name or feature (for example, EC2, S3 or VM, storage).
Recently visited services
Kinesis
Simple Queue Service
Simple Notification Service
CloudWatch
Billing
All services
Compute
EC2
Lightsail
ECS
EKS
Lambda
Batch
Elastic Beanstalk
Storage
S3
EFS
Glacier
Storage Gateway
Management Tools
CloudWatch
AWS Auto Scaling
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Systems Manager
Trusted Advisor
Managed Services
Media Services
Elastic Transcoder
AWS Cost Management
AWS Cost Explorer
AWS Budgets
Mobile Services
Mobile Hub
AWS AppSync
Device Farm
Mobile Analytics
AR & VR
Amazon Sumerian

Preparation

Reasons to Run In The Cloud

Adapt to changing workloads without maintaining hardware

Create replicated, isolated environments for Development / QA / Prod

Create replicated environments for Prod

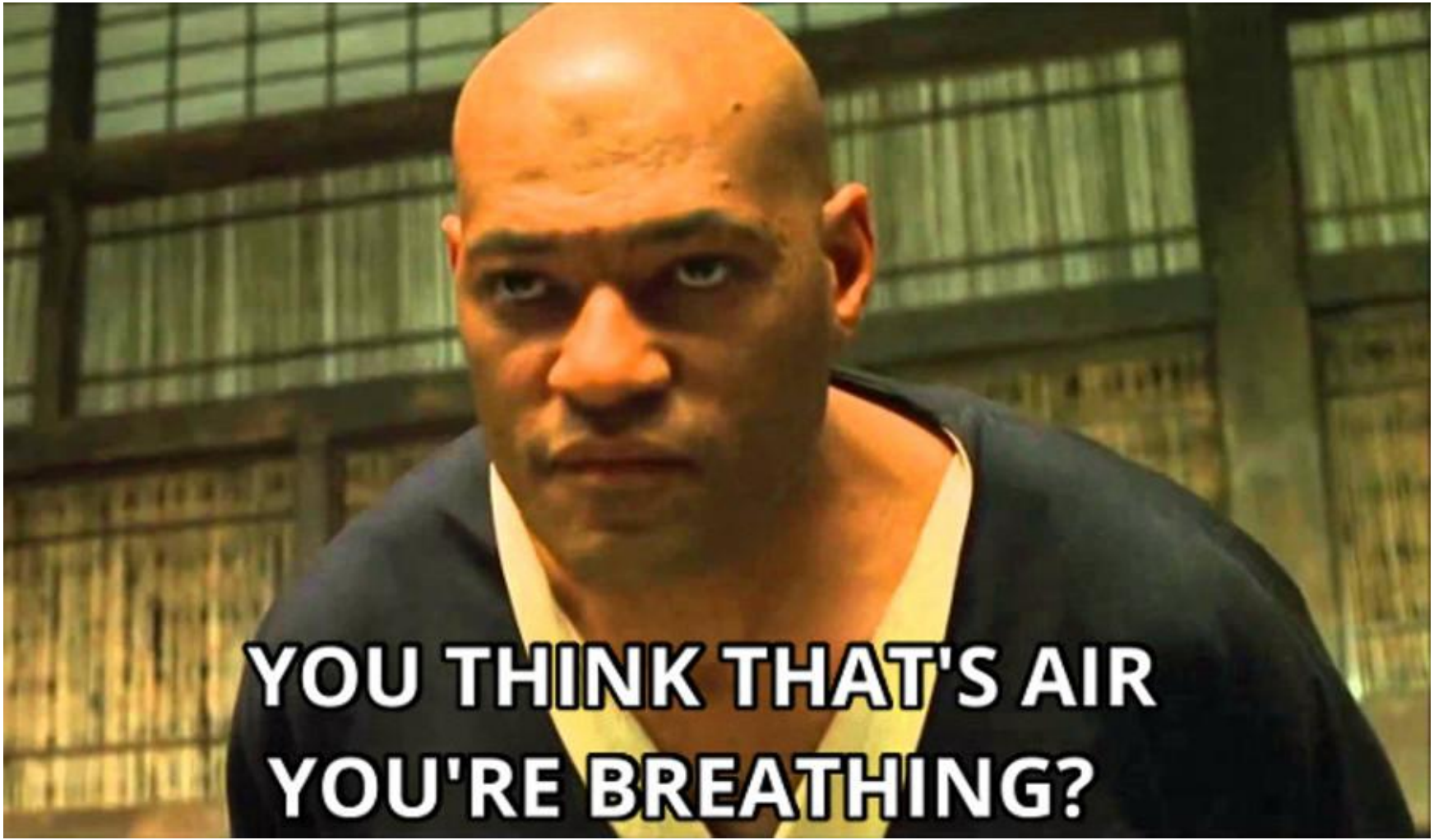
Deploy your applications in your users' geographic region

Use managed services to simplify your deployment architecture

~~Reduce or eliminate your Operations staff~~

If you take away nothing else…

Most of what you know about operations in data centers is irrelevant or harmful when running in the cloud


YOU THINK THAT'S AIR
YOU'RE BREATHING?

These are not pets

Strategy 1: Lift and Shift

Can be done incrementally

Beware that cloud hardware does not correspond to physical hardware

- There are subtle differences in performance, especially disk IO

Beware that connecting from your data center to Amazon's will introduce lags

Plan your network and security first!

Strategy 2: Cloud Native

Starting here makes the most sense for companies without existing infrastructure

Leverage managed services -- don’t stand up your own queue brokers!

> But recognize that your developers still work outside the cloud

Spend time learning about configuration options

Don’t expect your developers to be good at Ops

Identity and Access Management (IAM)

Users and Groups

Users are an entity that uses resources in an AWS account

Users typically also have access credentials (access key and secret key) for using command-line or programmatic access to resources.

Best practice: use multi-factor authentication for user logins, *especially if they can create or delete resources*

For multi-account deployments, define users in one account and let them assume roles

Groups are collections of users

A user can be a member of up to 10 groups

Both users and groups can be assigned Policies

Policies

Allows or denies specific operations on specific resources

- “Operation” means AWS API call -- very granular
- Can also apply conditions such as origination IP address

Amazon provides predefined policies for general resource permissions

- These are generally not granular enough for production use, but are fine for sandboxes

Limited number of policies may be attached to a user/role/group

- This implies that you shouldn’t create a unique policy for each resource

Roles

A named collection of policies

May be assigned to compute resources

- EC2 uses the term “instance profile” for a role that is assigned to an instance

May be assumed by users

- For command-line/programmatic access, this involves creation of temporary credentials
- Best practice: require role assumption to perform any destructive acts
- Possible best practice: users should not have inherent privileges other than ability to assume roles

Accounts and Organizations

Unit of billing, permissions management

- IAM users, groups, roles, and policies are unique within an account
- Can use roles/policies to allow cross-account access

Organizations allow multiple accounts to be combined

- Single account in organization receives billing
- Can control inter-account access

Best practice: create separate accounts to isolate deployment environments

Summary

Create policies based on capability, roles based on application

Use accounts as a hard barrier between dev, test, and prod

Give your developers a sandbox (or many) with full permissions

Use assumed roles to control access to non-dev environments

The Network

Regions

Geographic grouping of data centers

Resources namespaced by region

Not all regions support all AWS services

source: https://aws.amazon.com/about-aws/global-infrastructure/

Availability Zones

“Regions provide multiple, physically separated and isolated Availability Zones”

- Normally considered to mean “data center” but Amazon docs do not state this outright

Within a region, AZ’s are “connected with low latency, high throughput, and highly redundant networking”

- You are charged $0.01 per GB for cross-AZ traffic

High availability designs are based on nodes in multiple AZs

- Open issue: how many AZs do you need?

For disaster recovery you need resources in a different region -- or outside AWS

Virtual Private Cloud (VPC)

A software-defined network infrastructure

Isolated from each other and the rest of the Internet -- unless you allow access

Small number of VPCs allowed per region and account

Forget everything you know about physical networking, it doesn’t apply

Subnets

Isolated *logical* network segment

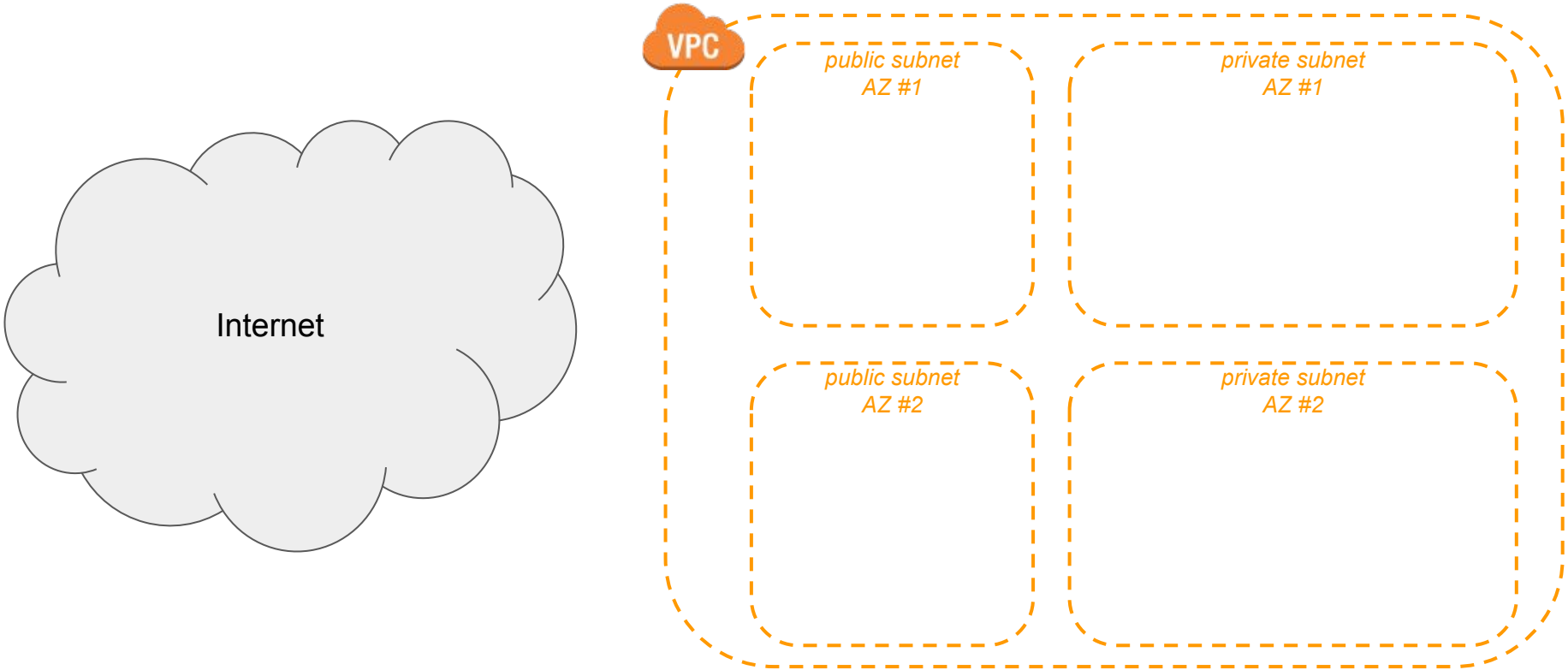
“Public” subnets allow use of public IP addresses, “Private” subnets do not

- You need a public IP or a NAT to make outbound connections

Can control routing of traffic between subnets

Don’t be stingy with subnet address space!

Archetypal Network Configuration



Security Groups

Equivalent to a firewall for inbound traffic

- You define which sources of traffic are allowed in based on CIDR and port

Multiple security groups can be applied to single node

- Result is union of all rules, increasing access

Security groups can reference other security groups as a source

- This allows fine-grained access within your VPC: for example, app-server talking to database

In 99% of cases, security groups are sufficient

- Alternative is network ACLs, which often get people into trouble

Other Stuff

Internet Gateway

The connection between your VPC and the Internet: all traffic goes through it

The thing that actually differentiates private and

Network Address Translation (NAT)

Hosts on Private subnet can’t talk to Internet -- this includes many AWS services

Two options: NAT Instance or Nat Gateway -- use NAT Gateway unless your IO volumes are high

Bastion Host

Small EC2 instance that has public IP address, accepts SSH connections/tunnels

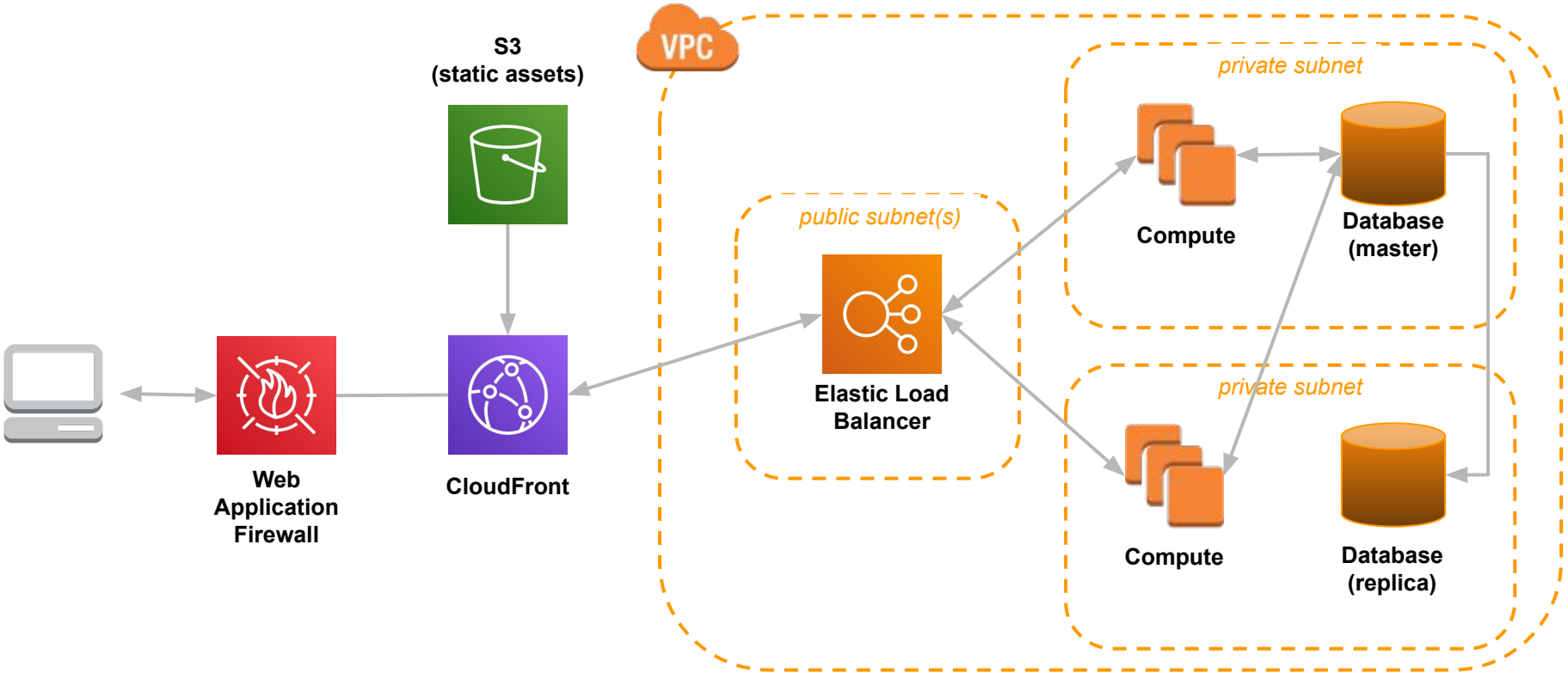
Alternative to VPN

Deployment

AWS lets you trade convenience for control

Managed services don’t give you as many knobs, and cost more, but reduce how often you wake up to the pager’s call

Archetypal Web-App Deployment



Web-App Front End

Web Application Firewall

Applies rules to inbound requests and blocks those considered malicious

Cloudfront

Caches that are physically close to users, reducing latency

Can load static assets from S3, reducing load on servers

Elastic Load Balancer

Directs client requests to appropriate server

Compute

Virtual Machines (EC2, Elastic Beanstalk, Lightsail)

You are responsible for picking the appropriate machine type, installing all software, monitoring, patching, backup, …

Containers (ECS, EKS, Fargate)

You are responsible for packaging your application on a base image and configuring it.

Serverless (Lambda)

You are responsible for packaging and configuring your application, not the environment.

Very scalable, but beware “first request” startup time

Databases

Relational Database Service (RDS)

- Managed installations of popular open-source and closed-source relational databases
- Doesn’t give you all features available if you self-host

Aurora

- Managed Postgres/MySQL cluster: multiple nodes, automatic failover
- You pay for each IO, so don’t skimp on RAM

DynamoDB

- Schema-less, distributed, document-oriented
- Appropriate when you can get/put by primary key

Configuration Management

Parameter Store

- Stores configuration using hierarchical keys
- Values may be encrypted
- Maintains history for each item
- Access controlled via IAM role

Secrets Manager

- Intended for secrets, not general configuration -- you pay $0.40/month per item
- Can automatically rotate credentials for RDS databases

General Advice

Elastic Beanstalk is a good starting place for simple deployments

- Includes load balancer, pre-configured instances, auto-scaling

For more complex EC2-based deployments, pre-build your AMIs

- Installing software takes time -- don’t do this when you start your instances

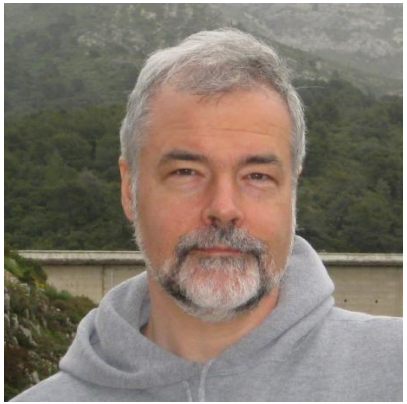
“Set it and forget it”

Final Thought

The average life expectancy of a non-instrument-rated pilot who flies into clouds is 178 seconds

source: Federal Aviation Administration
https://www.faa.gov/about/office_org/field_offices/fsdo/fai/local_more/alaskan_articles/media/178-Seconds_to_Live.pdf

About me



Programming computers since 1977, making a living at it since 1984, doing it on AWS since 2008.

https://www.kdgregory.com/

https://blog.kdgregory.com/

https://github.com/kdgregory