Architecting Scalable Web Applications Using Google App Engine

INTRODUCING GOOGLE APP ENGINE

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Hosted applications with built-in load balancing and autoscaling

Services and versions

Instances, instance classes and scaling

Choosing between the standard and flexible environment

Deploying and scaling a simple App Engine application

Prerequisites and Course Outline

Prerequisites

Basic understanding of cloud computing

Understanding how web applications are built, GET and POST HTTP requests

Working with NoSQL databases

Prerequisites: Basic Cloud Computing

Choosing and Implementing Google Cloud Compute Engine Solutions

Architecting Schemaless Scalable NoSQL Databases Using Google Datastore

Course Outline

Introducing Google App Engine

- App Engine vs. other compute options
- Standard and Flexible environments
- Services, versions, instances and scaling

App Engine Standard Environment

- Traffic migration and splitting
- Integrating apps with Cloud Datastore
- Integrating with Memcache for lower latency

App Engine Flexible Environment

- Using custom containers to deploy apps
- Integrating applications with other GCP services such as Pub/Sub

Scenarios: SpikeySales.com

Hypothetical online retailer

- Flash sales of trending products
- Spikes in user traffic

SpikeySales on the GCP

- Cloud computing fits perfectly
- Pay-as-you-go
- No idle capacity during off-sale periods

Introducing Google App Engine

Choices in (Any) Computing

Compute

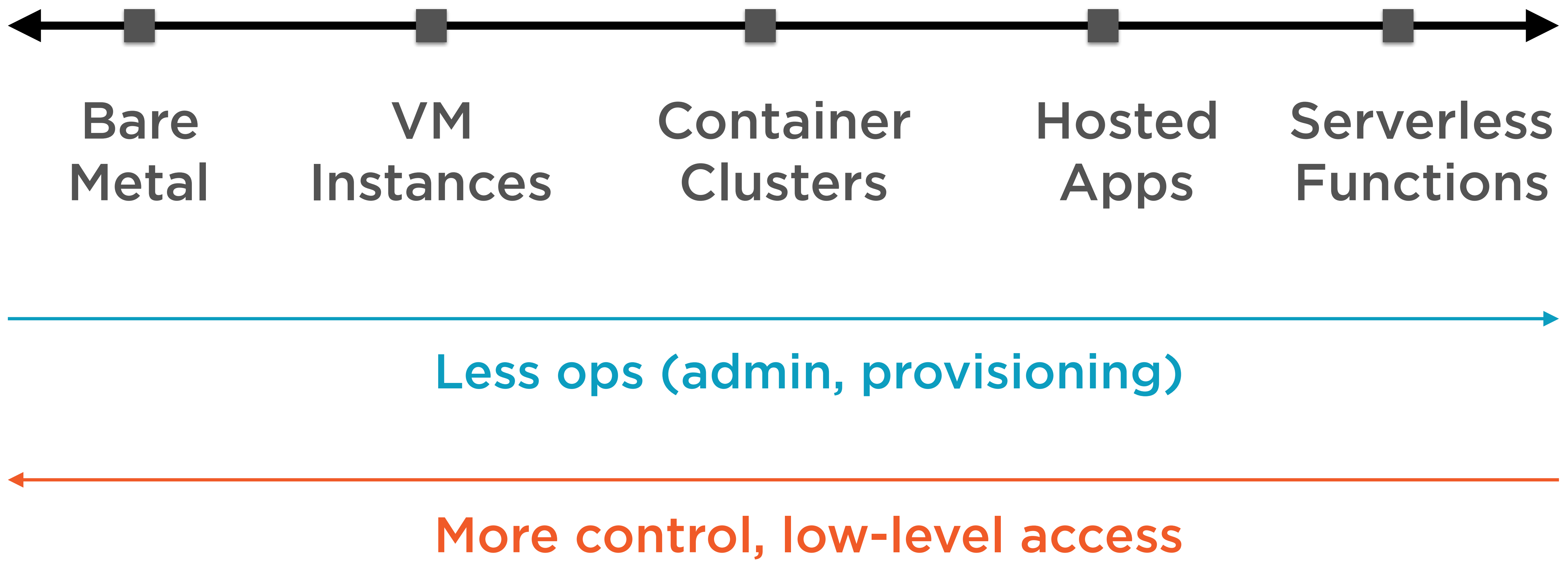
Where and how does code run?

Storage

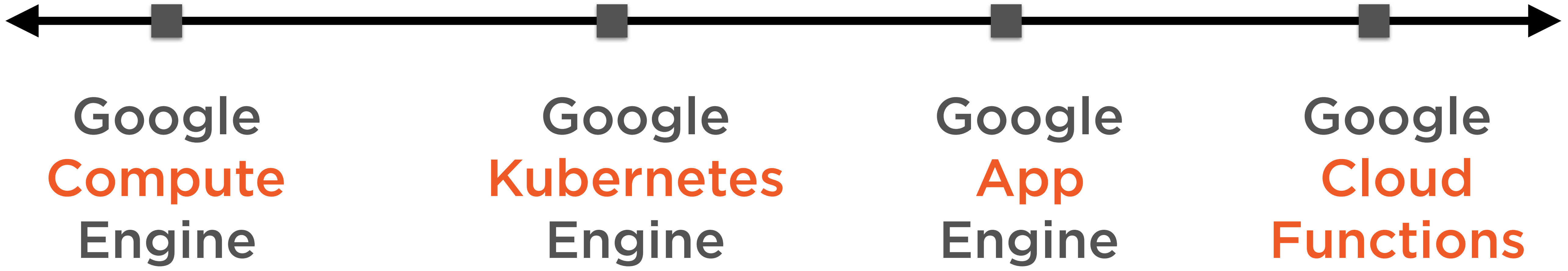
Where and how is the data stored?

Other choices - networking, logging etc. - are less important

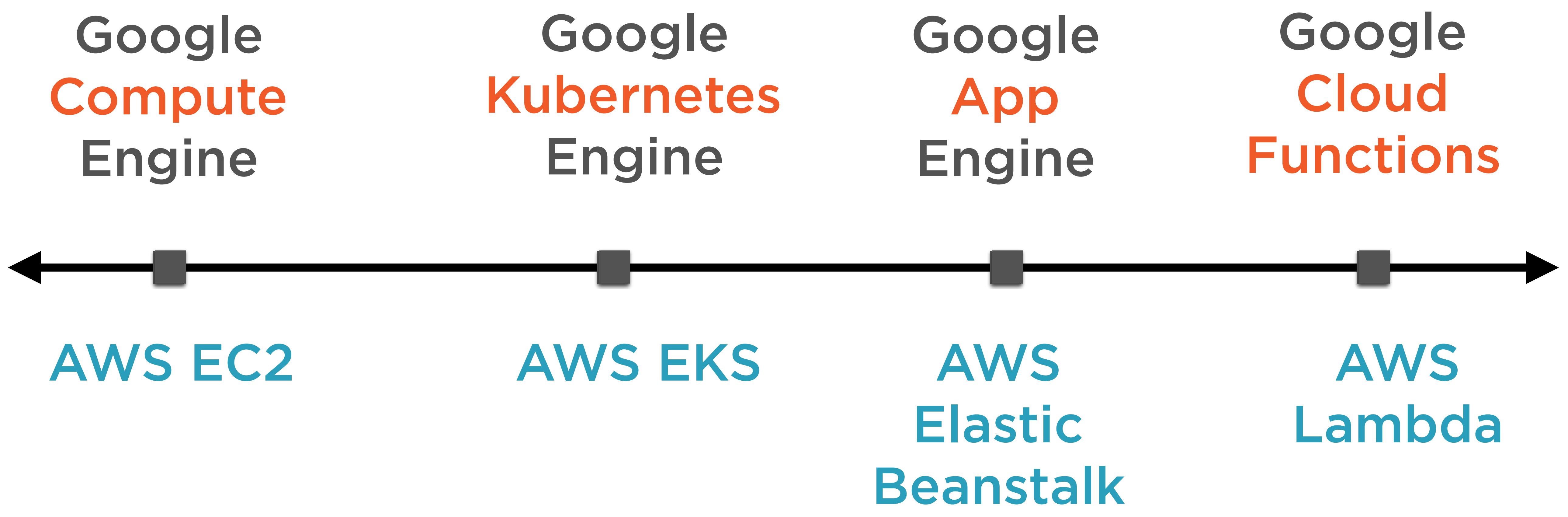
Compute Choices



GCP Compute Choices

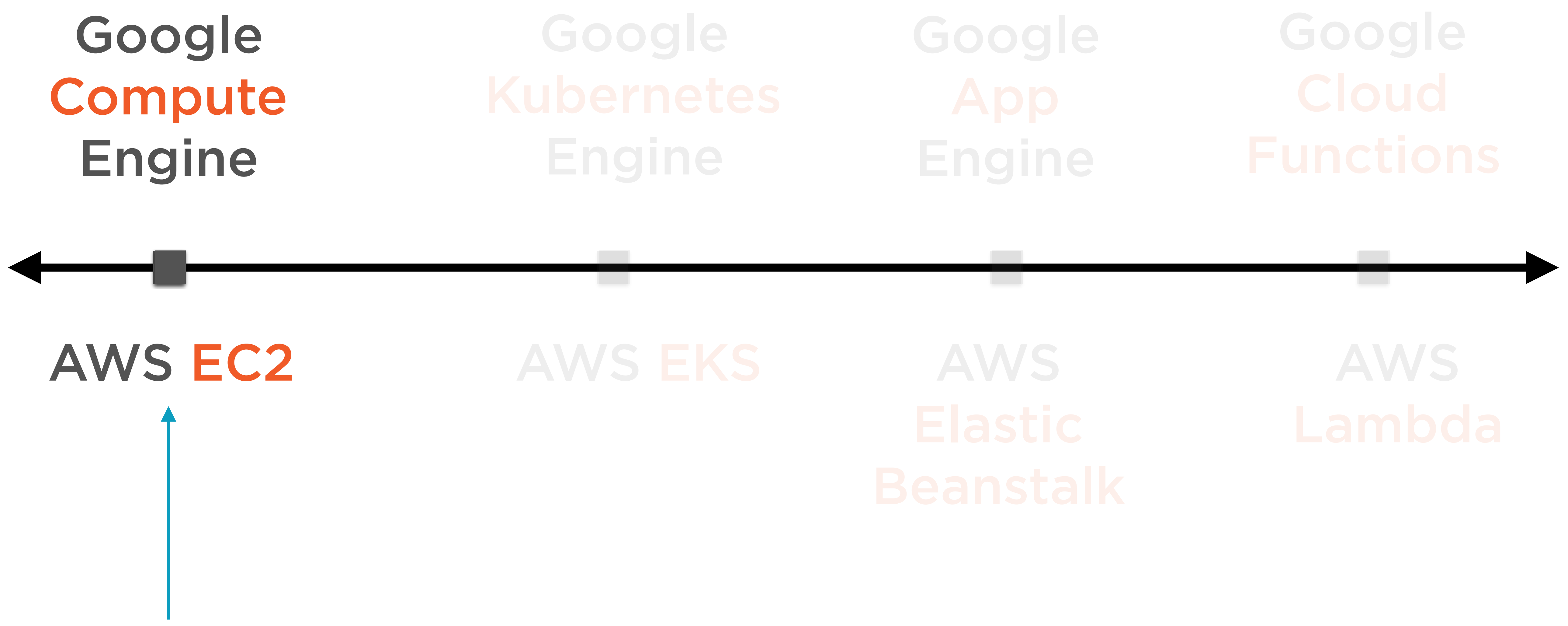


GCP Compute Choices



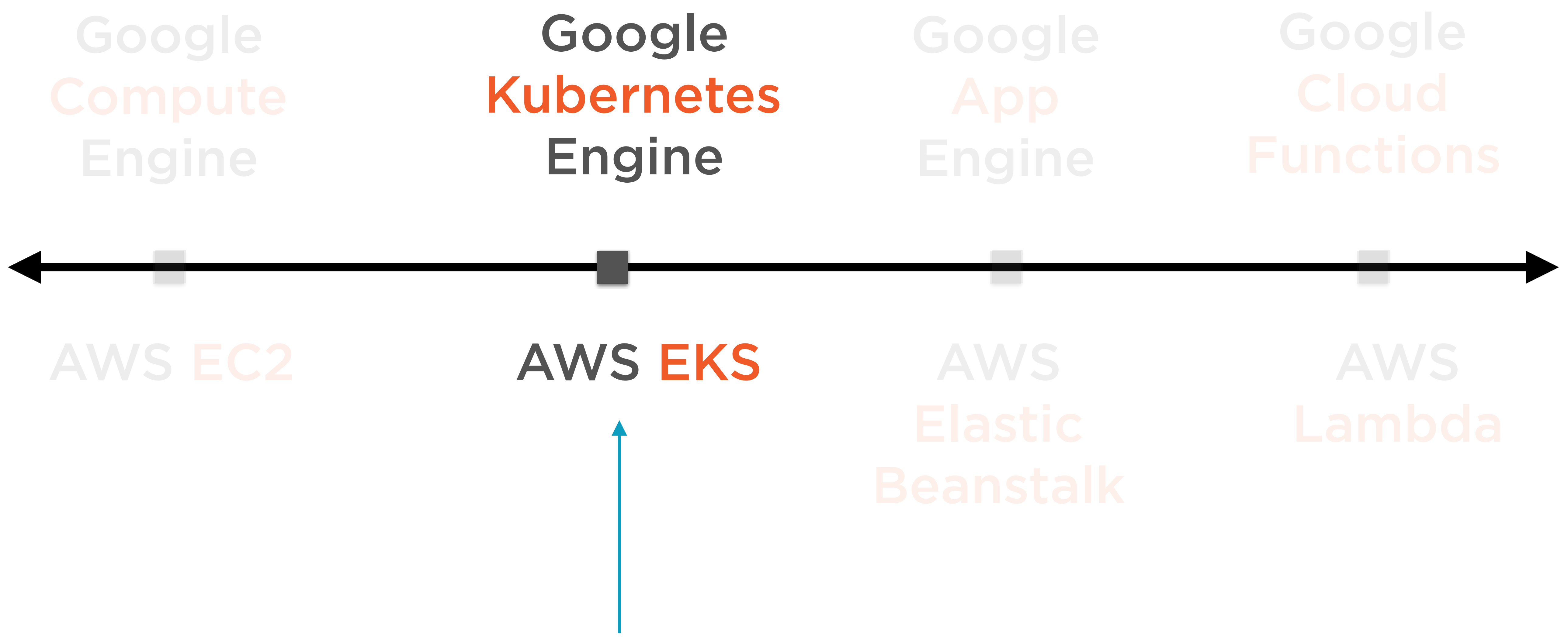
Every major cloud platform supports the same range of compute choices

GCP Compute Choices



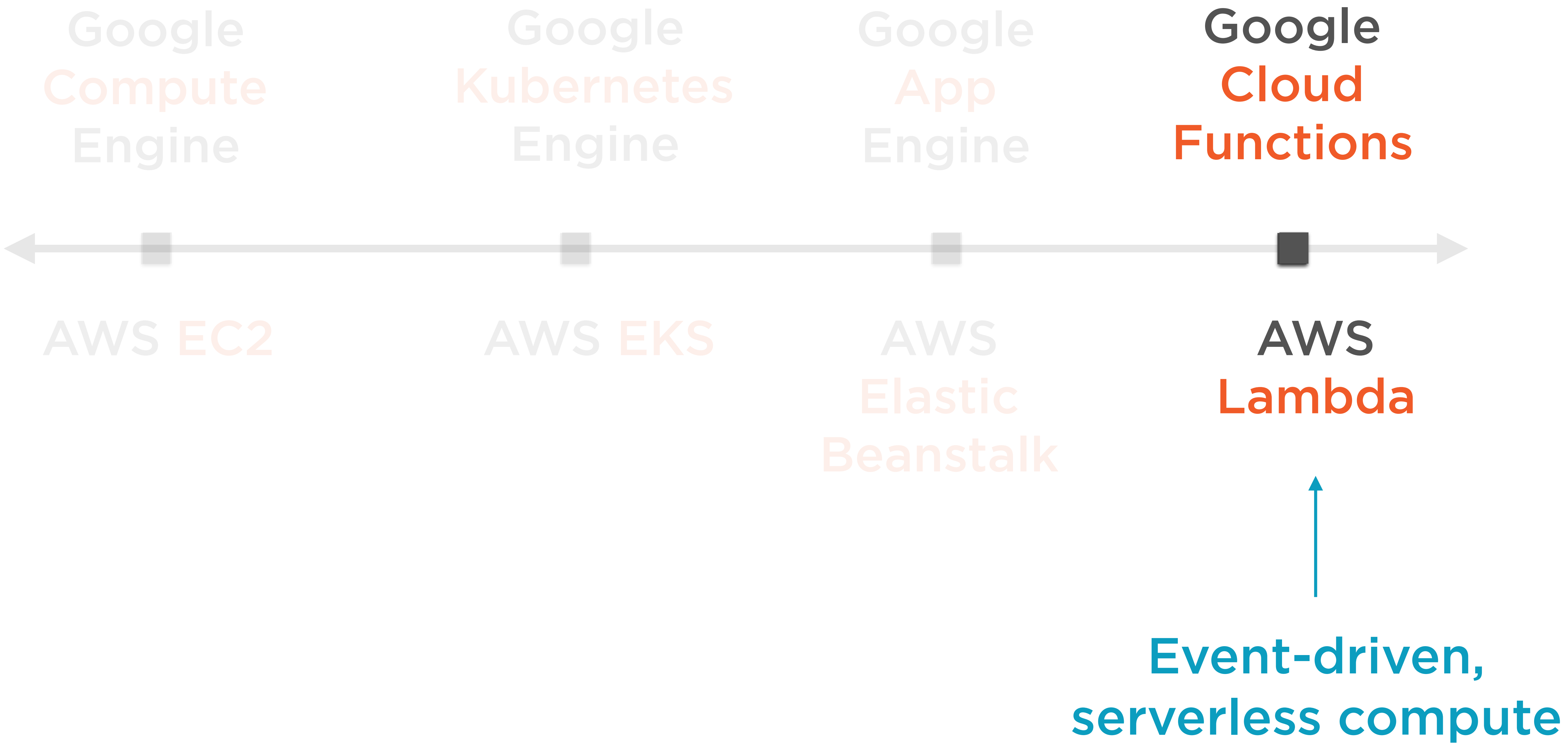
“*I*nfrastructure-*a*s-*a*-*S*ervice” (IaaS)

GCP Compute Choices

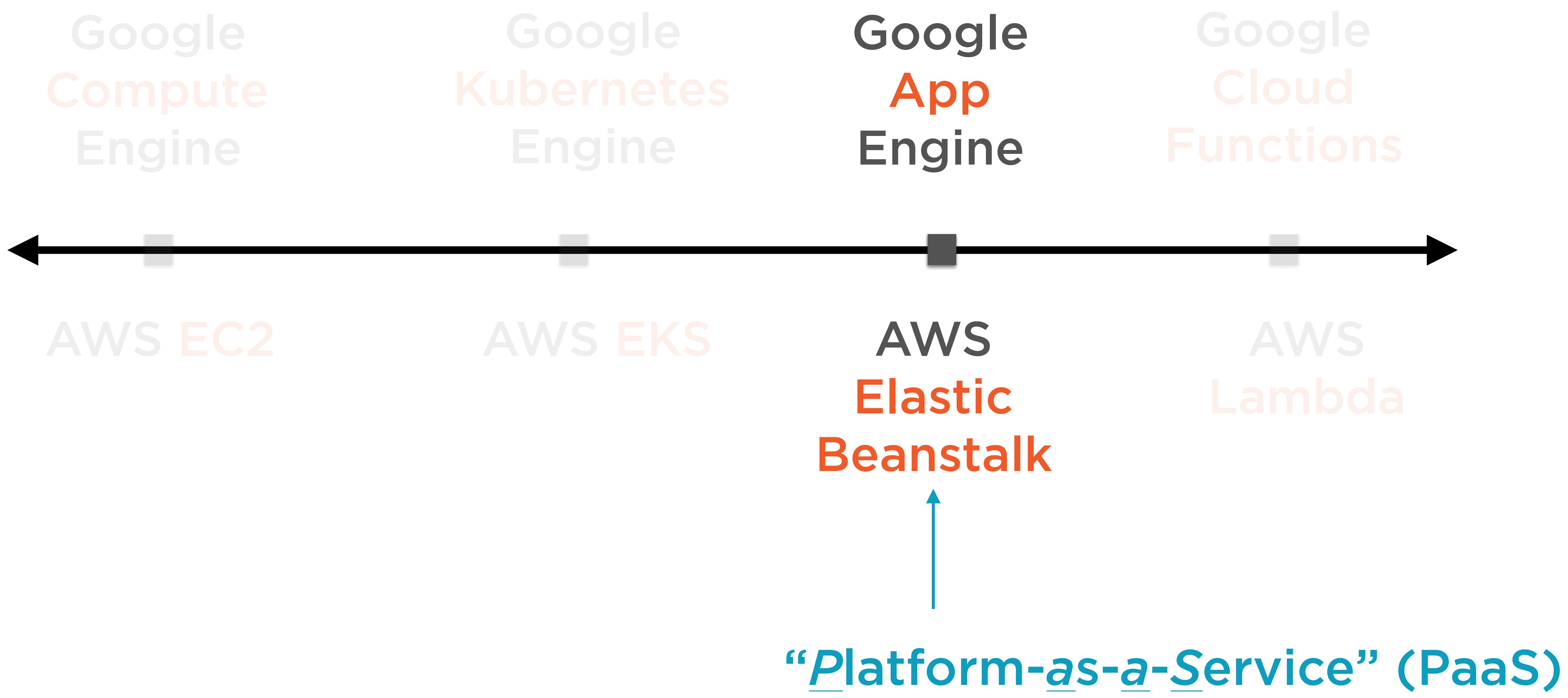


Container clusters - best in a hybrid multi-cloud environment

GCP Compute Choices



GCP Compute Choices



Google App Engine

Web framework and platform for hosting web applications on the Google Cloud Platform

App Engine Environments

Standard Environment

Flexible Environment

App Engine Environments

Standard	Flexible
App runs in a proprietary sandbox	Runs in Docker container on GCE VM
Code in few languages/versions only	Code in far more languages/versions
No other runtimes possible	Custom runtimes possible (Docker)
Apps cannot access Compute Engine resources	Apps can access Compute Engine resources and some OS packages

App Engine Environments

Standard	Flexible
Instance startup in seconds	Instance startup in minutes
Scaling: Manual, Basic or Automatic	Scaling: Manual or Automatic
No background processes	Background processes supported
No SSH debugging	SSH debugging supported
Scale to zero	No scaling to zero (minimum 1 instance)
No installation of third-party binaries	Installation of third-party binaries allowed

App Engine Environments

Standard	Flexible
Apps that experience traffic spikes	Apps that experience consistent traffic
Usually stateless HTTP web apps	General purpose apps
All instances in same zone (moved in case of zone outage)	Instances are part of regional Managed Instance Group

App Engine Standard Runtimes

Python 2.7, Python 3.7

Java 8, Java 7

Node.js 8 (beta), 10 (beta)

PHP 5.5, 7.2 (beta)

Go 1.9, 1.11 (beta)

App Engine Flexible Runtimes

Python 2.7, Python 3.6

Java 8

Node.js

Go 1.9, 1.10, 1.11

Ruby

PHP 5.6, 7.0, 7.1, 7.2

.NET

Custom runtimes

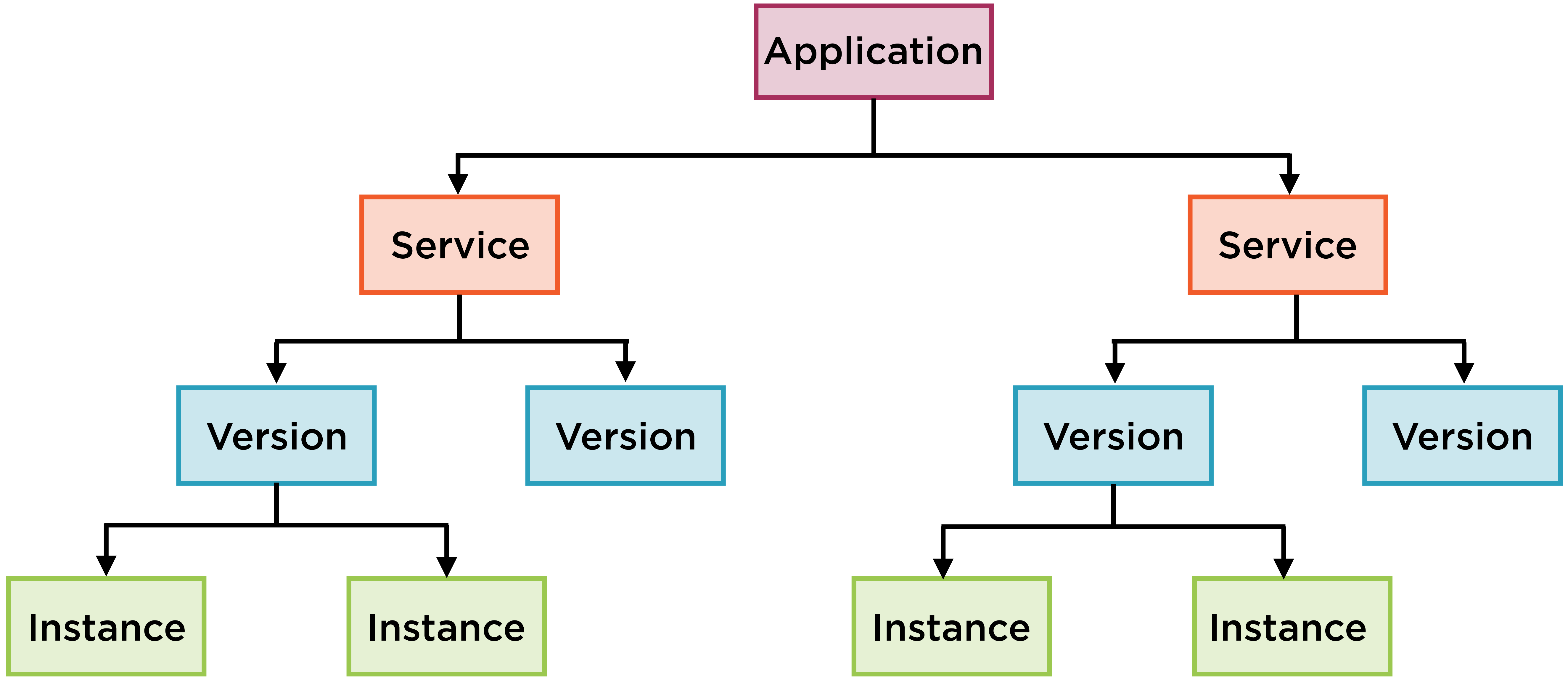
Choose the flexible environment for applications that require custom runtimes or third-party libraries not supported in the standard environment

App Engine Components

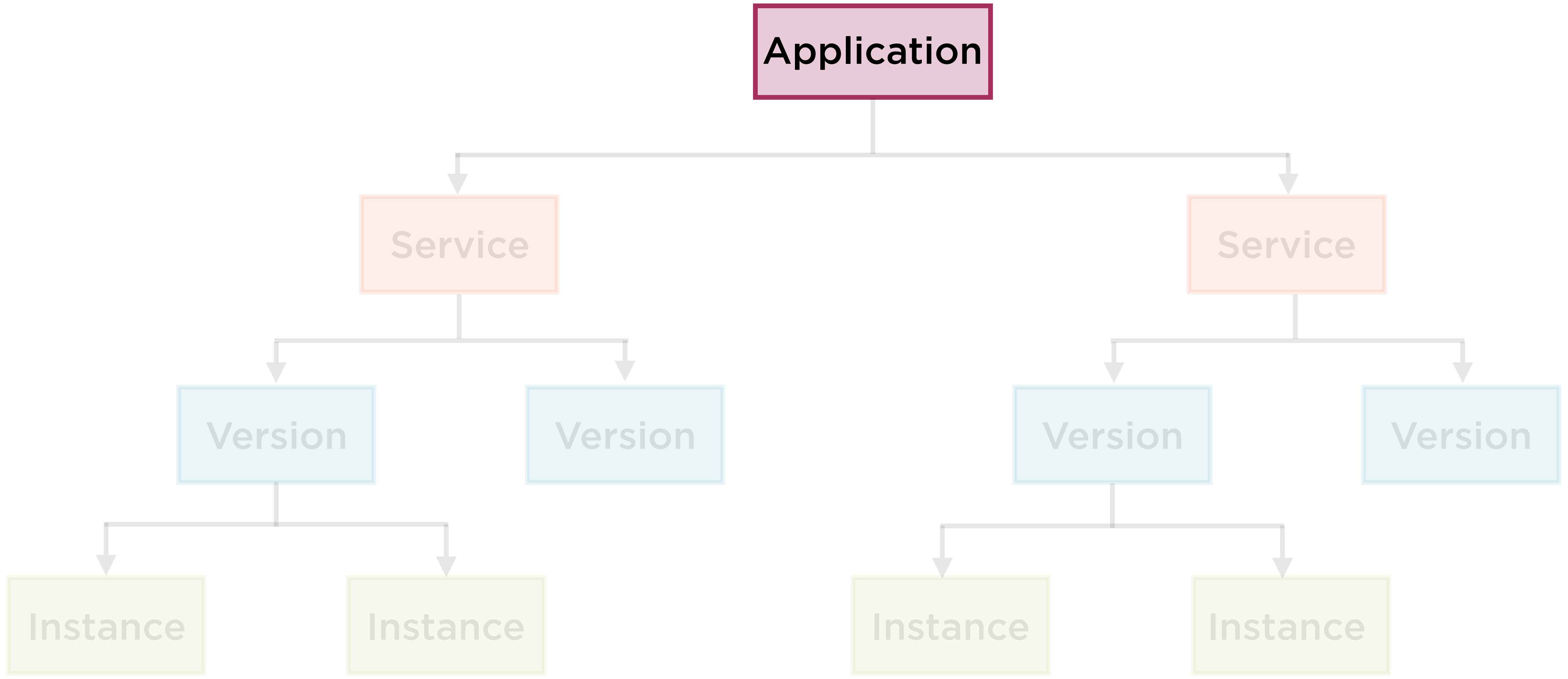
App Engine App

Single regional application resource consisting of hierarchy of services, versions and instances

Components of an Application



Application

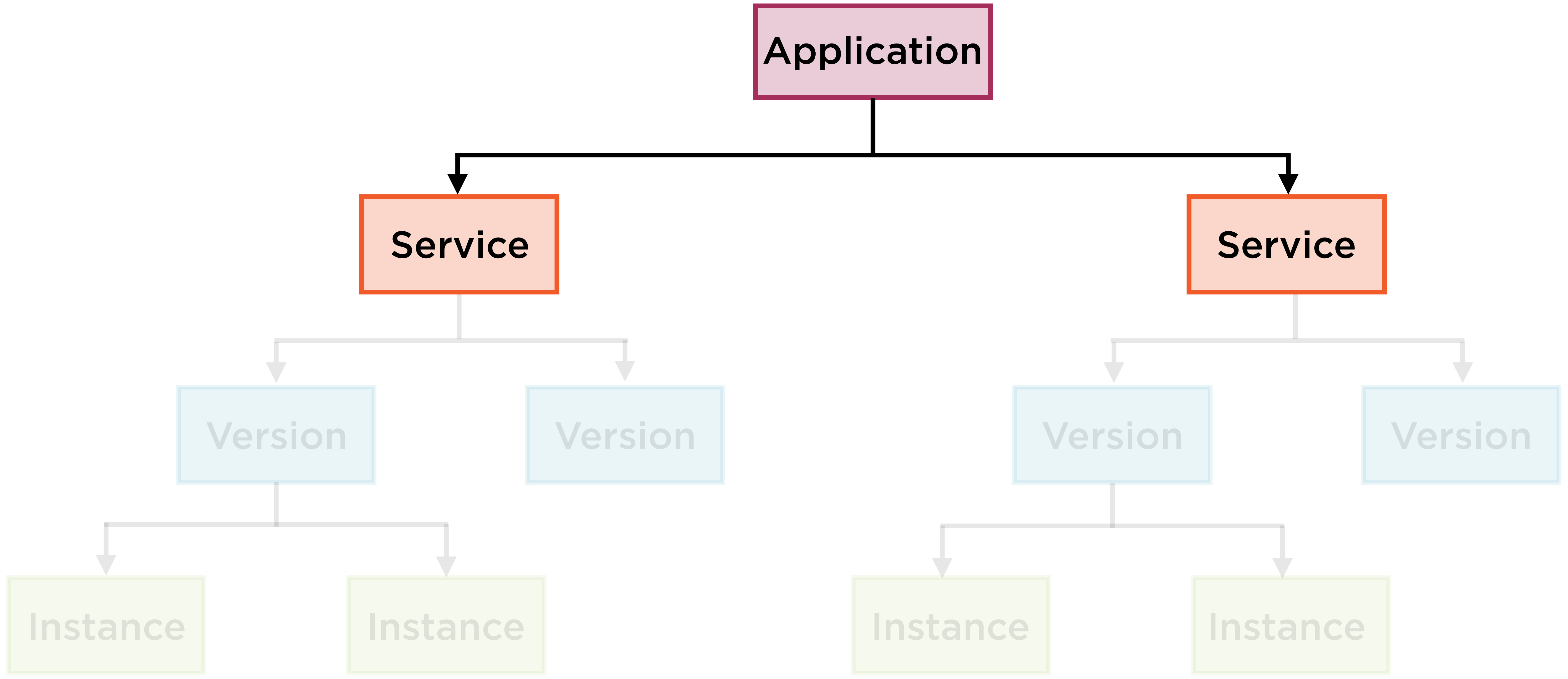


Top-level container for multiple services, their versions and instances

Services

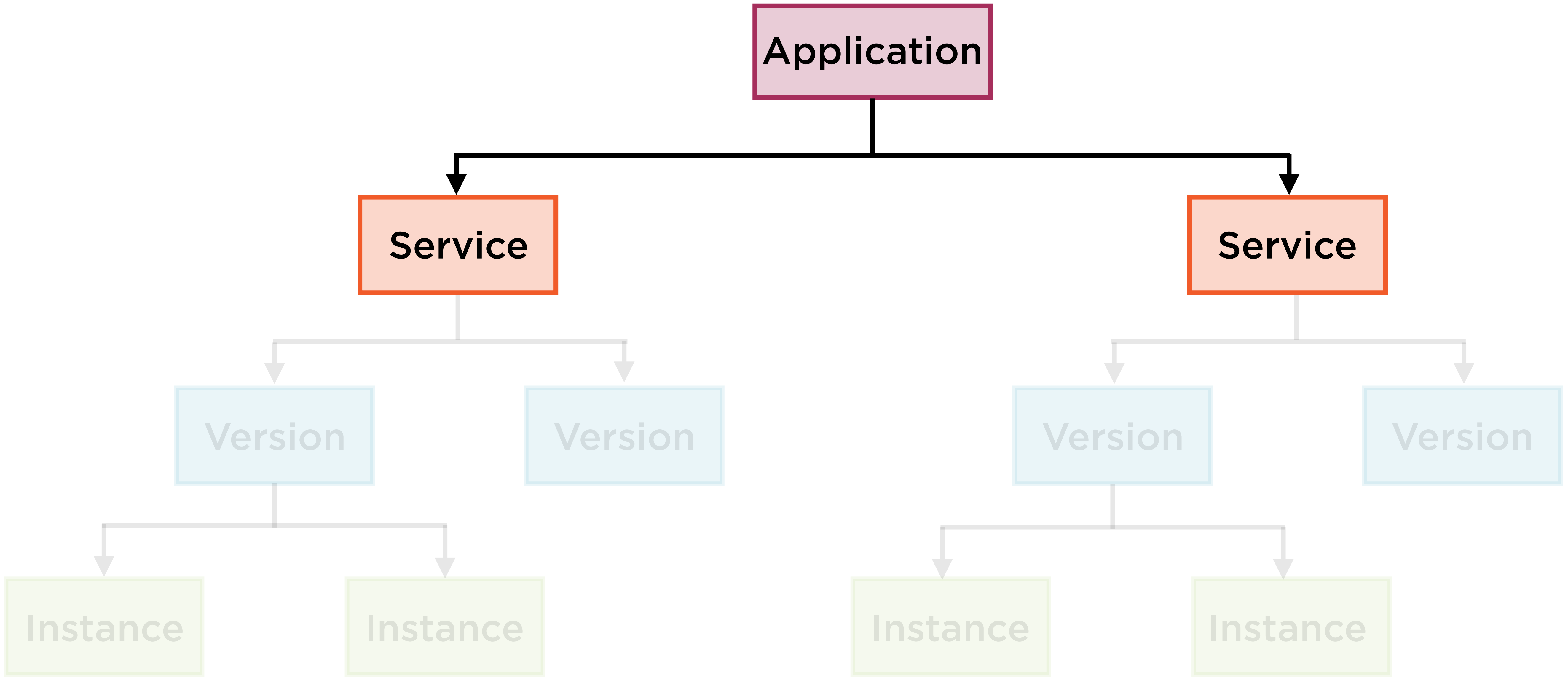
Each service in an App Engine app behaves like a microservice and scales independently

Single Application, Multiple Services



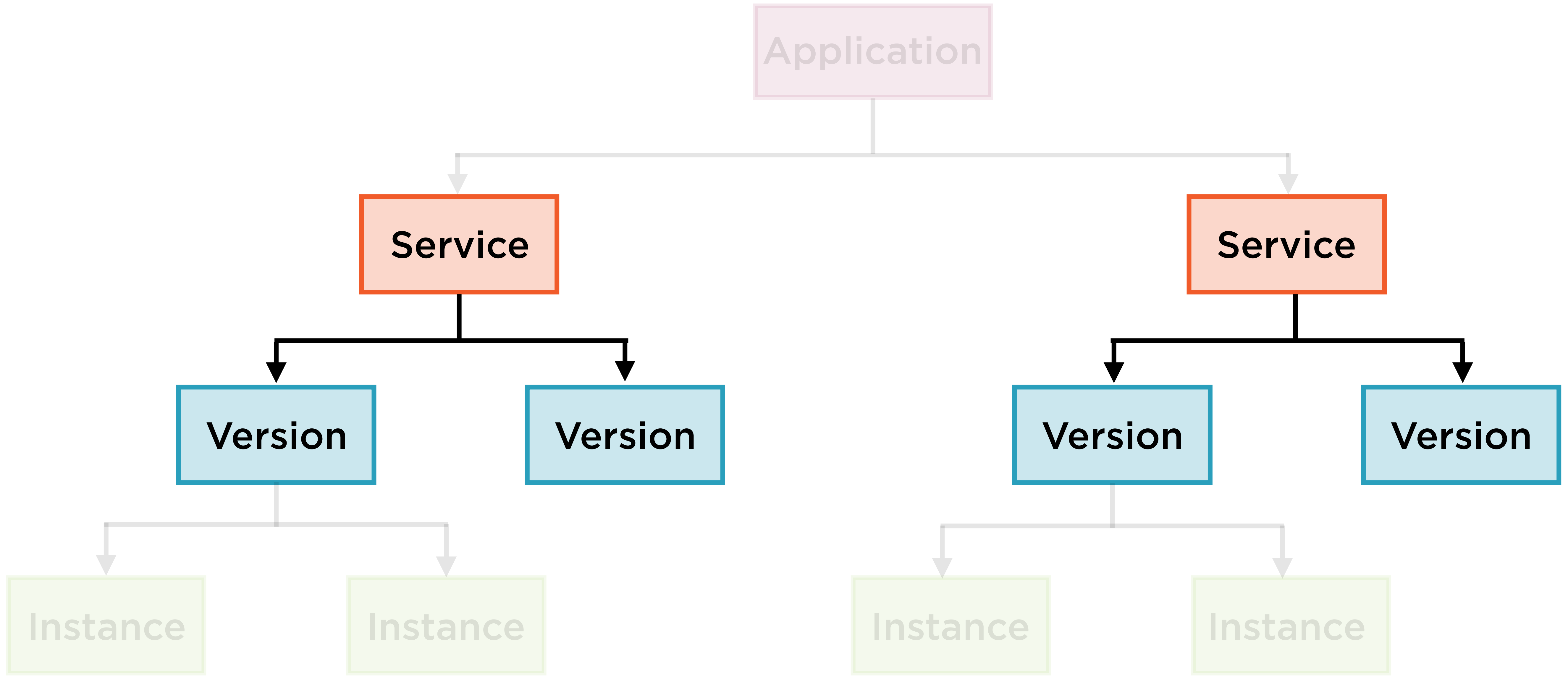
Each service is a logical component which can share features and communicate with one another

Default Service



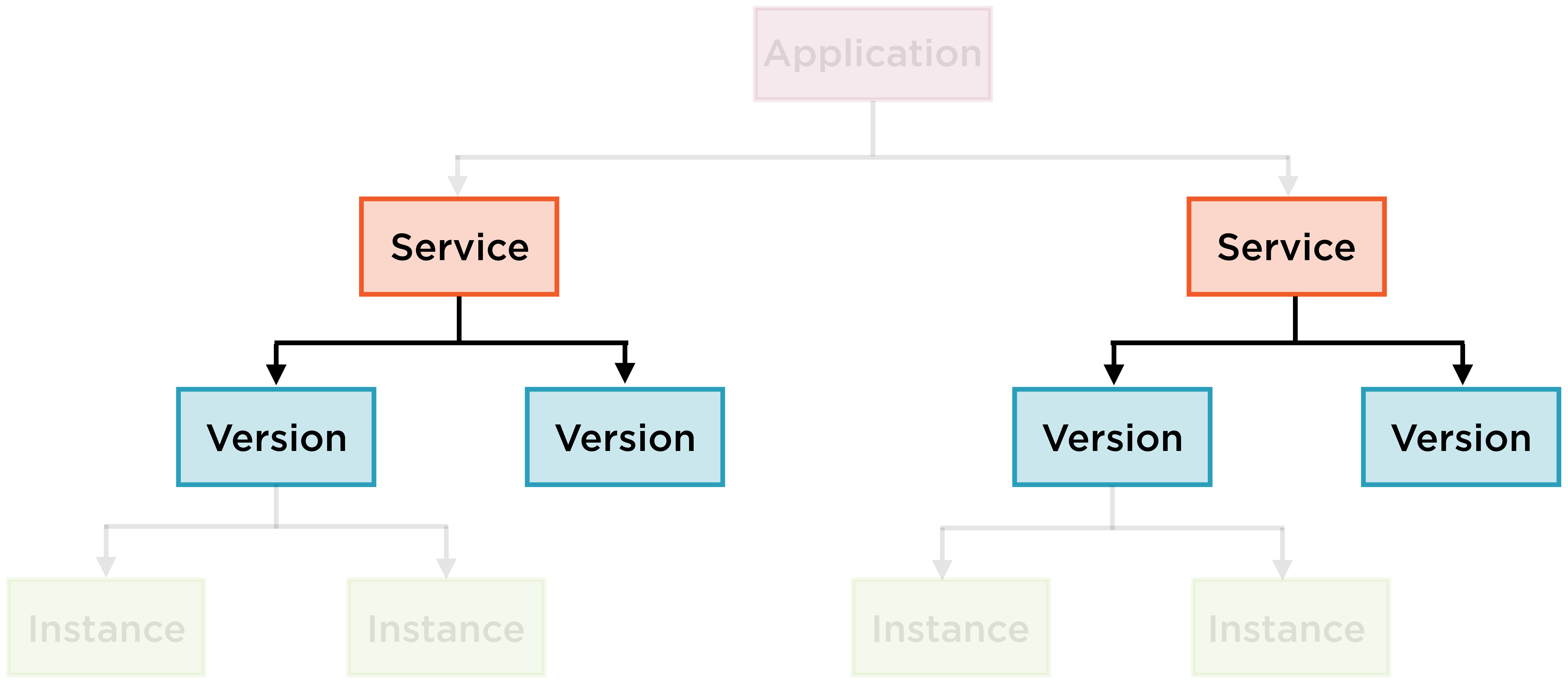
Every application includes at least one service, the default service

Versions



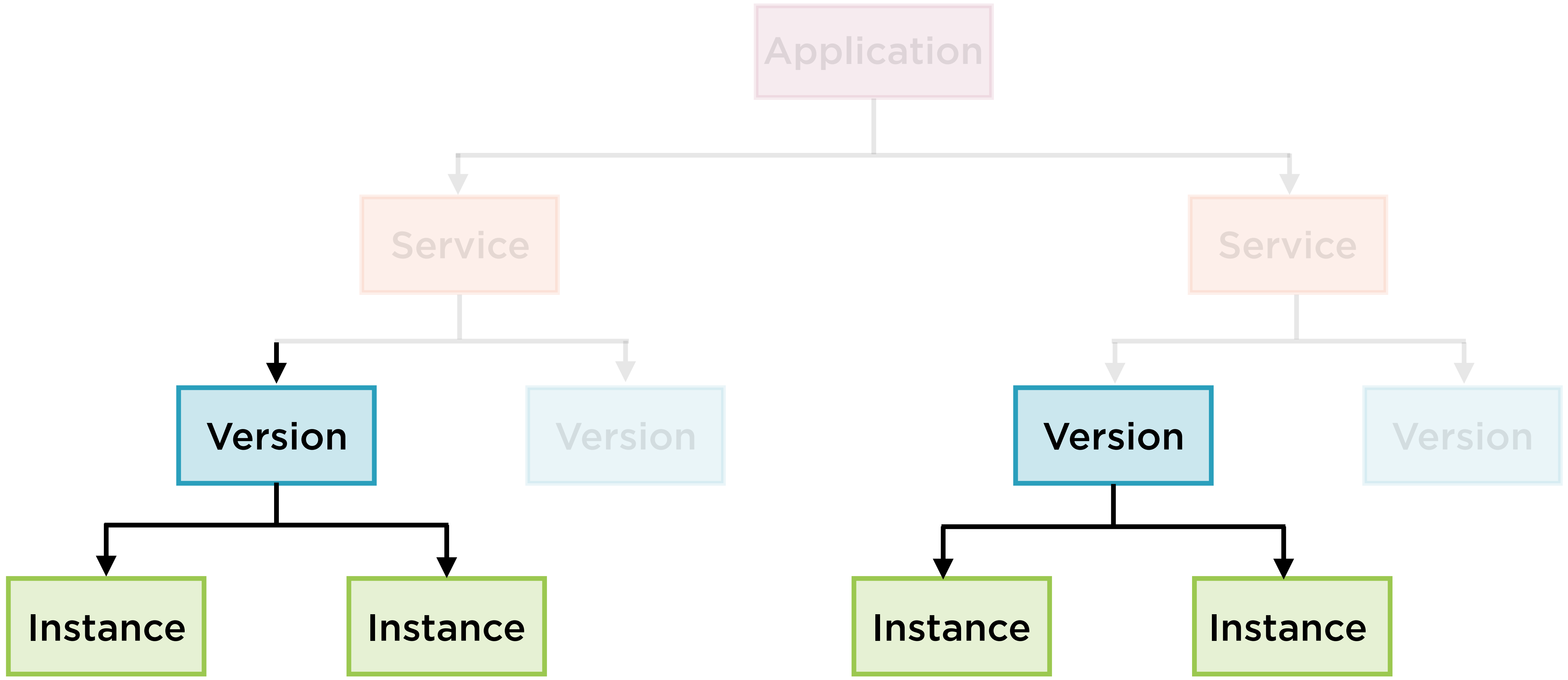
Multiple versions of every service can be deployed, traffic is automatically sent to the latest version

Versions



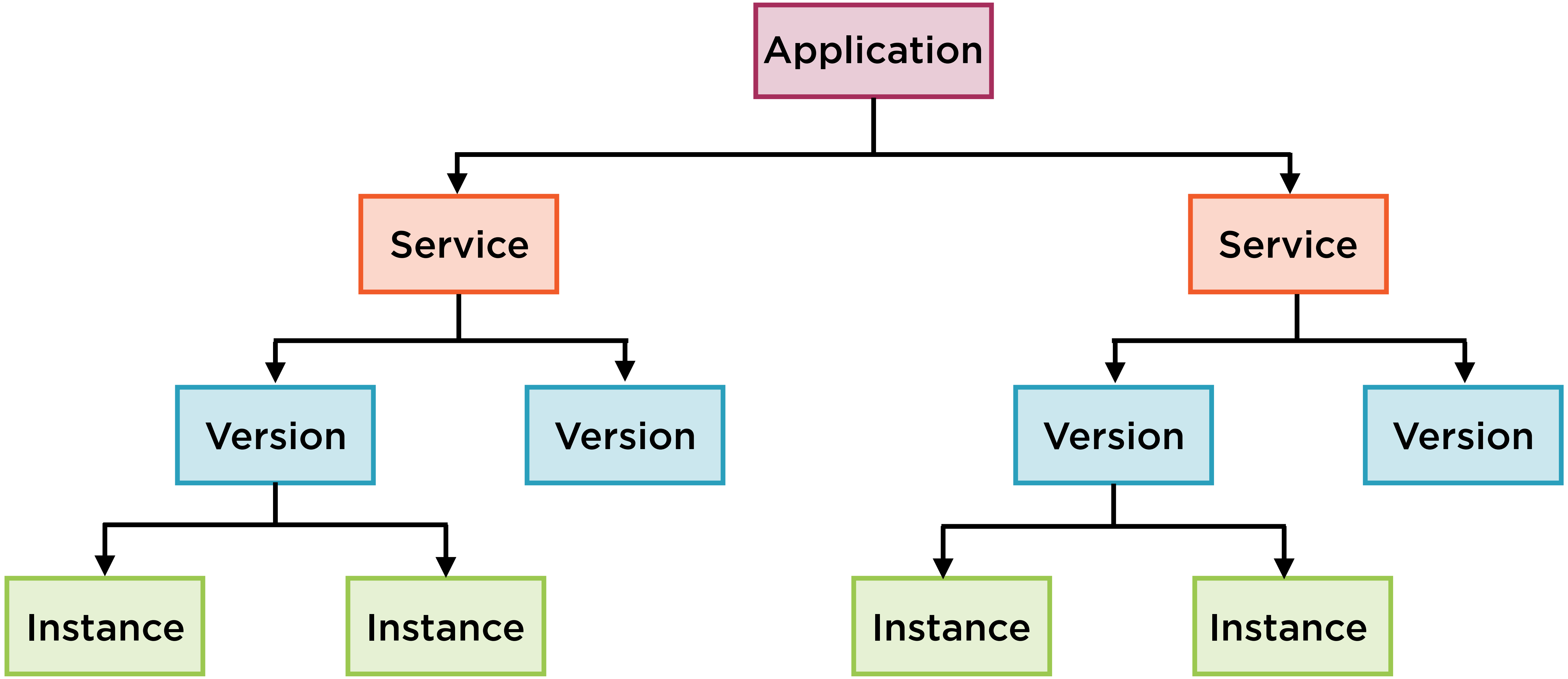
Every new deployment to a particular service creates a new version

Instances



Versions run on one or more instances - can configure App Engine to automatically scale instance based on load

Components of an Application



Requests

Routed to the services or versions that are configured to handle traffic

Target and route requests to specific services and versions

Instance Management

App Engine Environments

Standard	Flexible
Apps that experience traffic spikes	Apps that experience consistent traffic
Instance startup in seconds	Instance startup in minutes
Scaling: Manual, Basic or Automatic	Scaling: Manual or Automatic
No background processes	Background processes supported
No SSH debugging	SSH debugging supported
Scale to zero	No scaling to zero (minimum 1 instance)
No installation of third-party binaries	Installation of third-party binaries allowed

App Engine Environments

Standard	Flexible
Apps that experience traffic spikes	Apps that experience consistent traffic
Instance startup in seconds	**Instance startup in minutes**
Scaling: Manual, Basic or Automatic	**Scaling: Manual or Automatic**
No background processes	Background processes supported
No SSH debugging	SSH debugging supported
Scale to zero	No scaling to zero (minimum 1 instance)
No installation of third-party binaries	Installation of third-party binaries allowed

Instances

Computing units on which the application is hosted. Includes the language runtime, the App Engine APIs, the application code and memory

$$\begin{tikzcd}
 & \text{Scaling and Instance Types} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Manual} \arrow[d] & \text{Automatic} \arrow[dr] & \text{Basic} \arrow[d] \\
\text{Resident Instances} & & \text{Dynamic Instances}
\end{tikzcd}$$

Two Types of Instances

Resident

Dynamic

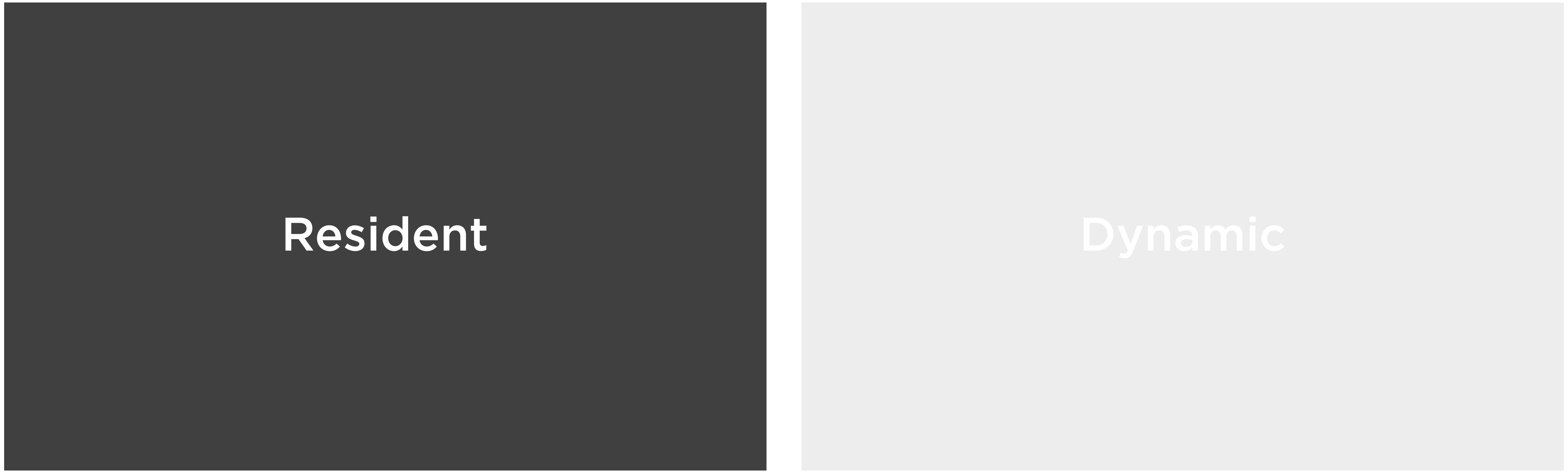
Three Ways to Scale

Manual

Automatic

Basic

Resident Instances



Run all the time and are always ready to serve traffic - having resident instances run your app can improve performance

Dynamic Instances

Resident

Dynamic

Starts up and shuts down automatically based on your application’s needs

Three Ways to Scale

Manual

Automatic

Basic

Manual Scaling

Manual

Automatic

Basic

Only uses resident instances to serve traffic, useful for applications that rely on the state of memory over time

Automatic Scaling

Manual

Automatic

Basic

Use dynamic instance to scale up and down based on the load, can specify a minimum number of resident instances

Basic Scaling

Manual

Automatic

Basic

Only uses dynamic instances which are turned down when app is idle - best for intermittent jobs

Instance Classes

Every instance belongs to an instance class which determines its compute resources and pricing

Instance Classes

Instance Class	Memory Limit	CPU Limit	Supported Scaling Types
F1 (default)	128 MB	600 MHz	automatic
F2	256 MB	1.2 GHz	automatic
F4	512 MB	2.4 GHz	automatic
F4_1G	1024 MB	2.4 GHz	automatic
B1	128 MB	600 MHz	manual, basic
B2 (default)	256 MB	1.2 GHz	manual, basic
B4	512 MB	2.4 GHz	manual, basic
B4_1G	1024 MB	2.4 GHz	manual, basic
B8	1024 MB	4.8 GHz	manual, basic

Instance Classes

Instance classes starting with F support automatic scaling

Instance classes starting with B support manual and basic scaling

Instance Lifecycle

Instance Lifecycle

Running	Stopped	Startup
Warmup	Loading request	Shutdown

All instances of same service and version share same state

Running

Stopped

Auto-scaled instances are always running

Manual/basic scaled instances may be running or stopped

Can stop instances using gcloud, web console or programmatically

Startup

Each service instance created in response to start request

Empty HTTP GET request to /_ah/start

Sent by platform, not users

Handled differently in manual, basic and automatic scaling

Warmup

Specific type of loading request

Sent by platform to load app

In advance of live requests

Only for automatic scaling instances

Loading request

Occurs during first request

This is called the loading request

Instance loads libraries and resources

Initialization slows down request processing

Shutdown

Usually shutdown hook can run before instance terminates

Sometimes may not happen

Possible reasons

- Manual stop
- Deployment of new version
- Max memory exceeded
- Instance moved for system reasons

Demo

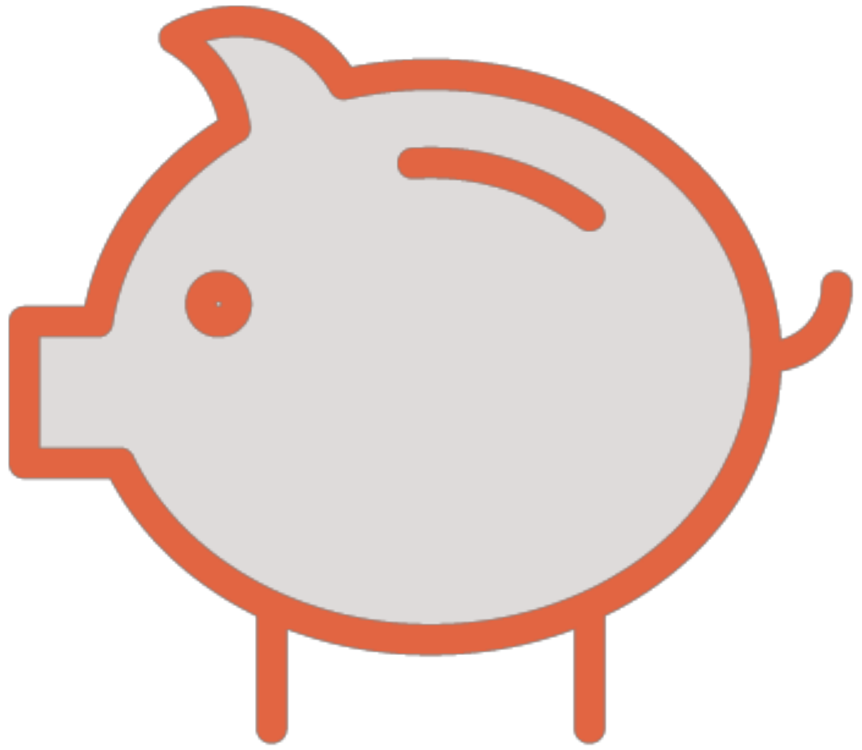
Building and deploying a simple App Engine application

Demo

Autoscaling in App Engine

Pricing

Pricing



For the standard environment based on instance class

Cost per-instance, per-hour

Flexible environment pay for vCPU, memory and persistent disk

Pricing

Additional charges for Cloud Datastore, Memcache, network traffic

https://cloud.google.com/appengine/pricing

Summary

Hosted applications with built-in load balancing and autoscaling

Services and versions

Instances, instance classes and scaling

Choosing between the standard and flexible environment

Deploying and scaling a simple App Engine application