CS 5412/LECTURE 4: FUNCTION (LAMBDA) COMPUTING MODEL

Ken Birman

Fall, 2022

MANAGING ELASTIC μ-SERVICES IS HARD

It is very hard to build a completely new elastic application from the ground up.

In practice, these μ-services need to be connected to all sorts of specialized vendor-created cloud infrastructure services and tools.

So… how do cloud customers create new customized solutions?

TODAY'S LECTURE

Suppose you are a cloud vendor like Amazon on Microsoft

Your cloud is amazingly awesome and wonderful! And so easy to sell!

But in fact you know that it is *not* easy for customers to create new μ-services. They will find it super hard, even if you provide lots of demos, documentation and even consulting help.

TODAY'S LECTURE

What should you do?

This question led to an idea: Build a very basic one, but let the customers "specialize" it, so that it can be easily specialized

How can this be accomplished on modern computing systems?

CONCEPT OF CUSTOMIZATION

It centers on having some prebuilt component (for us, a very generic form of μ-service instance), but one that doesn't really "do" anything.

For example, perhaps you can send it a very basic kind of request called ping, and it sends back a reply "ok", nothing more.

This would not be a useful μ-service but it would become the framework users can customize.

THINGS THAT ARE NOT CUSTOMIZATION

Telling a data analysis program which files to read

Giving a parameter to a program to tell it to reverse the sort order

Typing a command to a shell like bash to run some program

WHAT MAKES IT CUSTOMIZATION?

When we talk about customization, you *always write actual code*.

Some customization frameworks require specific languages, while others allow customization with code in any language you like (dotnet is very flexible, and has 44 languages built in!)

But the central idea is that when certain "events" occur, your logic is executed and temporarily takes over – their platform is under your control

WHY WOULD WE CUSTOMIZE A DATABASE?

Take the example of a database system

Suppose we post some query that will trigger a particular action. Why would this be useful?

- *If the price of Microsoft drops more than 1.5% notify me by text message.*
- *If this patient's blood test comes back positive for infection, immediately start antibiotics*
- *If this car is likely_to_crash into the next car, notify them both instantly … (you may also have to code the logic for "this car", "next car", "likely_to_crash"!)*

SO…

You are bringing arbitrary code into that existing server or platform component and changing what it can do!

Yet you are also "hitching a free ride" on the integration it already supports, with the app manager and with the tools it uses for elasticity. A system like Cascade goes even further: it is a key-value store but with replication, self-repair after failures, consistency. You gain all of this.

Why customize? To "evolve" that existing system to match your need, without even needing to see their code!

CONCEPT OF CUSTOMIZATION

… so, now we should ask what options exist for customizing a μ-service

Before we do a deep dive, let's pause to think about customization as a more general concept, focusing on Linux operating systems

What is customization "about"?

CUSTOMIZING A CAR

You get to pick the colors, and often there are add-on packages

This is a bit like setting configuration parameters

Many cloud services allow (or even require) customers to configure them before use. For example, you might set the minimum or maximum number of instances that will be running… this has cost implications for you!

WHERE DO CONFIGURATIONS "LIVE"

There are a variety of common options, but big cloud vendors try to standardize.

On Azure, most configuration data is held in JSON files. This is just a very simple file format full of **"variable" = "value";** lines (with nesting, in **{ }**)

Some programs want some form of configuration file on the local disk, in the local directory. It might have a name like xxx.rc or xxx.cfg

OTHER FORMS OF CUSTOMIZATION

When you set up a computer for the first time, and install Ubuntu or CentOS, it won't initially have logins, and once you add a login, there won't be any user programs running.

One way to cause a user program to run is to log in and run it yourself

Before you logged in, the machine was "blank". But now it is running the "Apache Web Server", as an example. It can serve web pages

WEB SERVICE… BUT NO WEB SITE

Even for this case you may also need to make firewall "tunnels" to enable requests to reach it, and your machine needs an externally visible IP address so that they can issue http requests.

But if you take those steps, and point a browser at the machine, and the Apache web server is running, it shows a blank screen when users connect.

Why blank? Because there need to be web pages for it to "serve up"

SO, YOU CUSTOMIZE A LITTLE MORE!

Using a web page editor you can create your fancy new web site

You copy the files to the folder your Apache web server is running in

And just like that, you now have a live web site! Anyone who knows theip address can visit http://123.45.67.890/ and this will pop up the home page of your new site

… YOU ACTUALLY WANTED YOUR SITE TO HAVE A NAME

So you can customize a bit more!

You register a new domain name with a name registry company, like Go Daddy. And now

http://www.mylittlepony.com/

is up and running!

YOU COULD DO ALL THIS ON A CLOUD, TOO

Everything just described could be hosted on Amazon or Azure

The Apache Web Service is one possible option for the first tier – you can set up an account exactly this way, copy your web site to Azure, and then instead of using GoDaddy to register your new company web site, register it via the Azure domain registry service.

- The feature is called "buy a custom domain name"
- You access it via the Azure app manager service, at this link.

Buy a custom domain name - Azure App Service | Microsoft Docs

https://docs.microsoft.com/en-us/azure/app-service/manage-custom-dns-buy-domain

Microsoft | Docs Documentation Learn Certifications Q&A Code Samples Shows Events

Search Sign in

Azure Product documentation Architecture Learn Azure Develop Resources

Portal Free account

Filter by title

- App Service Documentation
- Overview
- Quickstarts
  - ASP.NET
  - Node.js
  - PHP
  - Java
  - Python
  - Ruby
  - WordPress
  - Static HTML site
  - ARM template
  - Custom container
  - Multi-container app
  - App on Azure Arc (preview)
- Tutorials
- Samples
- Concepts
- How-To guides
  - Configure app
  - Deploy to Azure
  - Map custom domain
    - Buy domain

Download PDF

Docs / Azure / App Service / Web Apps /

Buy a custom domain name for Azure App Service

Article • 12/16/2021 • 9 minutes to read • 13 contributors

App Service domains are custom domains that are managed directly in Azure. They make it easy to manage custom domains for Azure App Service. This tutorial shows you how to buy an App Service domain and assign DNS names to Azure App Service.

For Azure VM or Azure Storage, see Assign App Service domain to Azure VM or Azure Storage. For Cloud Services, see Configuring a custom domain name for an Azure cloud service.

Prerequisites

To complete this tutorial:

- Create an App Service app, or use an app that you created for another tutorial. The app should be in an Azure Public region. At this time, Azure National Clouds are not supported.
- Remove the spending limit on your subscription. You cannot buy App Service domains with free subscription credits.

Buy an App Service domain

For pricing information on App Service domains, visit the App Service Pricing page and scroll down to App Service Domain.

In this article

- Prerequisites
- Buy an App Service domain
- Prepare the app
- Map App Service domain to your app

Show more

Slide 12 of 12 Accessibility: Investigate

REVIEW DOCUMENTS

Type here to search

80°F Partly sunny 1:18 PM 8/26/2022

OTHER FORMS OF CUSTOMIZATION

Think about coding in Javascript or Python.

Both are *interpreted* programming languages, meaning they don't compile to machine code. Instead, a program called the Javascript runtime or the Python runtime is launched first, and then it reads your code in, parses it, and this "runs your code".

But in a sense, your code isn't running – the runtime system is running.

OTHER FORMS OF CUSTOMIZATION

How do we leverage Python in the Azure cloud?

Most people use a platform called Flask, which can be found here.

Flask was originally created to let you use Python as a GUI solution (graphical user interfaces are hard to build by hand, and Flask automates that, behind a flexible app framework where you can drag and drop "widgets" like buttons and image windows).

OTHER FORMS OF CUSTOMIZATION

Flask also has a built-in framework for talking to the cloud using RestFUL RPC. They call it WSGI and pronounce it "whiskey"

RestFUL RPC encodes (serializes) messages into web pages! Then it uploads the whole web page. The reply comes back as a web page, too

This makes it easy to write programs that can send data to the cloud, although in fact it is kind of slow if you benchmark carefully.

YOU CAN EVEN USE FLASK INSIDE THE CLOUD

Flask also supports a form of lightweight container virtualization

It allows you to package your entire python program into a container that is pretty cheap to launch, or to shut down. Docker supports this, and Azure and AWS both support docker

With this many people create their own first-tier cloud services coded in Python, running in Flask.

CUSTOMIZING A RUNNING PROGRAM

One issue with Python and Javascript is that they are slow compared to a compiled program, that maps directly to machine instructions.

But on the other hand, compiling a program is only possible if you know what machine architecture you plan to run on. AWS and Google and Azure all have different types of servers. There is no single standard!

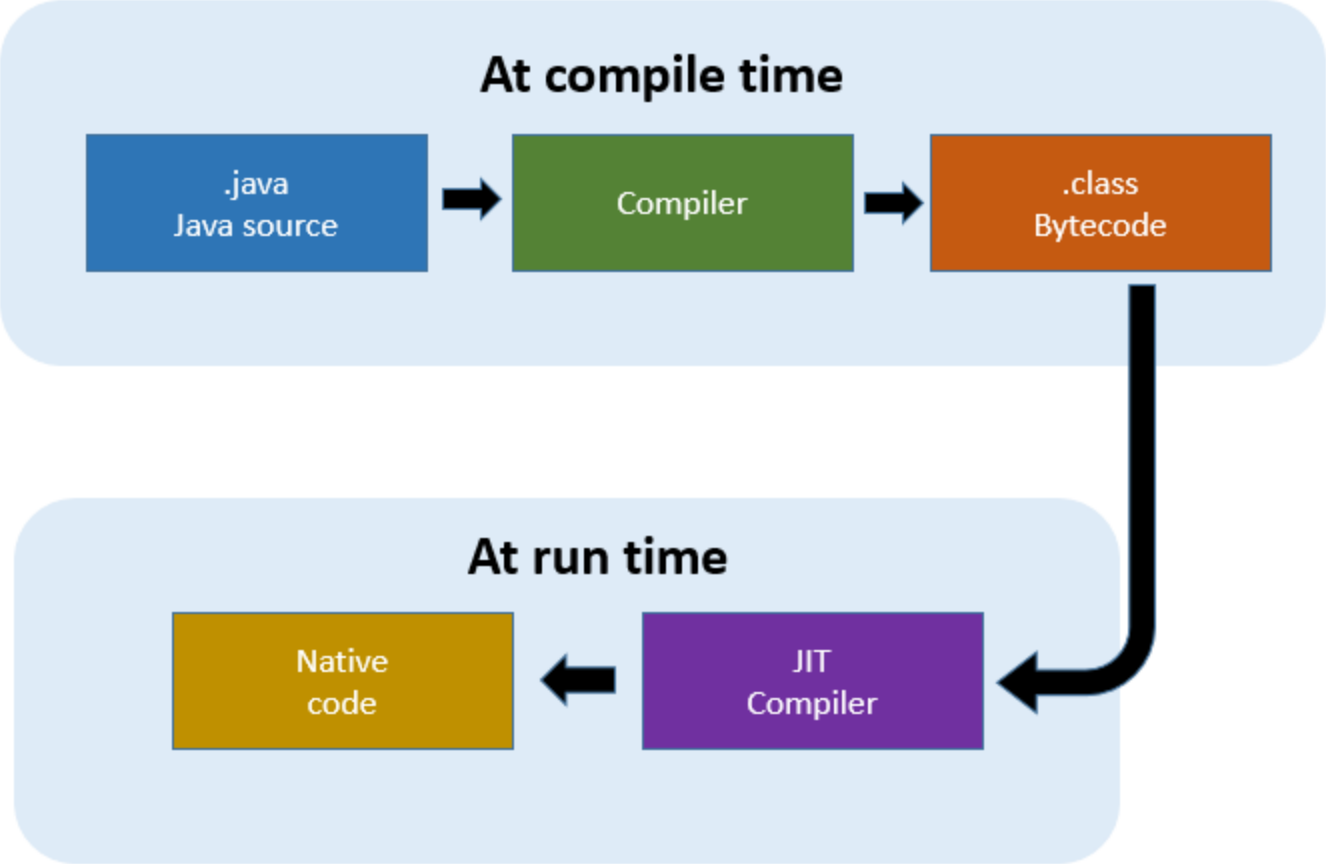
It sounds as if you would need to compile one by one, on the specific servers

FORTUNATELY, THERE ARE MORE OPTIONS!

Sometimes a preexisting running program provided by the vendor can call into code you supply, in Python, Java, C# or other languages

Take Java as an example.

Java compiles in two steps: first we compile into a .class file in so-called byte-code form. At runtime, a "JIT" operation does just in time compilation (this is what JIT stands for) and generates machine code


At compile time
.java
Java source
Compiler
.class
Bytecode
At run time
Native
code
JIT
Compiler

EVEN A PROGRAM WRITTEN IN C++ CAN ISSUE "UPCALLS" TO USER-PROVIDED CODE

C++ 17 is one of the world's fastest programming languages.

Many companies are using it these days, although there are other options too. But C++ 17 (and soon, C++ 20…) are widely standard.

Suppose a program is written in C++ 17 and already compiled, but we want to "extend it".

STEPS TO EXTENSIBILITY

First, the C++ program itself needs to be designed to be extensible

Many databases can support extensions, and Cornell's Cascade key-value storage system can too

The extensible program will have some sort of option or command that tells is to "load and initialize" the user's extension code.

WRITING EXTENSION CODE

You code would be written as a dynamically loadable library, ordll

This is quite easy and supported in most compiled languages.

Then your dll can just have a method called initialize, with void type, taking arguments that can be passed in

LOADING EXTENSION CODE

You tell some server, like Cascade, to load thedll

The Cascade server instance finds the file: mycode.dll, and “maps” it into memory (meaning, the code is accessible right in memory)

Then it looks up the address where initialize() landed. And now it can call your initialize method!

INVOKING EXTENSION CODE

Developer configures Cascade to load her DLL (her code)

WHAT DOES INITIALIZE "DO"?

… it turns around and calls back into Cascade to register to **watch** some keys (recall that a key is filename or directory name)

Now, if anything changes in that file (**put** to the same name), or in that directory (**put** with some extension of that path), the method given in the **watch** request will receive an upcall from the Cascade key-value server

EXTENSIBLE THINGS IN AZURE

In fact, Azure is full of extensible μ-services that can be customized this way, in many programming languages

All the dotnet (used to be .NET) languages can be used: C#, F#, etc.

Cascade takes this even further and also allows C++, standard Java, Python and we will soon support C# dotnet too.

DEVELOPING THESE EXTENSIONS

First, you need to identify the specific kind of μ-service you will use

Then, from docs.Microsoft.com, you learn about its extension options. These are often called "trigger points" or "event triggers".

Then, from Visual Studio Code, you can write a "trigger function" or "lambda" for that kind of event

EVENTS HAVE TYPES

Each kind of event typically has its own associated metadata

The triggered lambda will be passed this metadata as arguments

For example, in Cascade, a lambda will learn the key that it was watching that caused the upcall. Thus one lambda can actually watch many keys. It can use **get** to fetch the actual data for that object.

OTHER μ-SERVERS HAVE OTHER CUSTOMIZATION OPTIONS

Each kind of μ-Server offered by a vendor like Microsoft is owned by a developer team, and these teams each make their own decisions

As a result, there are quite a few places where Azure allows customization by user-supplied lambdas (they definitely prefer that these be in C#, which is a language a lot like Java, but Python generally can work)

You need to read the documentation or do searches to figure out the options. Azure actually has two options: lambda and "functions". Functions are heavier weight and really run entire programs in containers

EVEN VERY ELABORATE CLOUD SERVICES OFTEN SUPPORT TRIGGERED FUNCTIONS

Cloud database products support a kind of query called a "trigger" that will watch for some condition, then run a lambda or function you supply

- They often call these "stored procedures"

Cloud file systems and key-value stores can notify your program or run a function/lambda if the contents of a directory change, or a file is changed

- A "change" includes modifying files, create, delete, rename
- Your program will need to decide what changed and what to do! But often the answer is to just redisplay whatever data was in the file

TERMINOLOGY SUMMARY

Serverless computing: A style of computing in which we customize existing platforms rather than talking to big existing servers such as databases

Function computing: Tends to refer to a whole container (docker) that will contain a program, with its own “main”, that gets event information from its arguments or from the initial environment.

Lambda computing: You attach small lambdas in a language they support, and the platform does upcalls to your code (often via dll loading)

WHAT WILL THE FUTURE BRING? LOW CODE

There is growing interest in a kind of "drag and drop" customization

Suppose that some platform has a GUI for customization where the event upcall "points" are visible, and a widget box with pre-created data source widgets, action widgets for standard tasks like YOLO (machine vision), etc

… building a customized μ-service could feel like making a PowerPoint presentation! Yifan Wang is exploring this as a PhD research topic

WRAPUP

We learned about

- Customizing a pre-existing service
- Differences between configuring it (by setting parameters) and adding our own functionality (attaching event handles, functions, lambdas)

Why customize?

- Building an elastic cloud service from the ground up is feasible, but hard
- Vendors offer standard but very basic μ-services, with built-in elasticity. By customizing them, you transform the basic ones into "specialized" μ-services

WRAPUP

We learned about

- Customizing a pre-existing service
- Differences between configuring [illegible] dding our own functionality (attaching e[illegible] s)

In other courses, cut-and-paste gets you into trouble. In cloud computing, we *require* you to cut-and-paste! But document where things came from.

Why customize?

- Building an elastic clo[illegible] service from the ground up is feasible, but hard
- Vendors offer standard but very basic μ-services, with built-in elasticity. By customizing them, you transform the basic ones into "specialized" μ-services