Application #: 593

Section I: Applicant Information

Applicant name
Doe, John

Email address
support@api2pdf.com

Job title
Senior Software Engineer

Company name
Api2Pdf

Company address
44 Fake St, Arlington, VA, 00000

Company website
https://www.api2pdf.com

Company twitter
https://www.twitter.com/api2pdf_status

Phone #
555-555-5555

Section II: Entry Information

Product name
Api2Pdf

Brief product description
Api2Pdf

Why did you create this product?

Api2Pdf is a REST API that helps application developers generate PDFs at massive scale. It was co-founded by myself, Zack Schwartz, and my partner Kunal Johar. The two of us also run another company called OpenWater. OpenWater's customers live and die by PDFs. And they are not normal PDFs either. Often times, the PDFs are hundreds of pages long and contain high-res photos. As our customer base grew, our costs to generate PDFs ballooned. Eventually, we had a whole server dedicated to producing PDFs and nothing else. Scaled all the way up, it was costing us over $1000 a month.

That server eventually crashed too, and customers couldn't generate PDFs anymore. One of our customers relying on this capability had a deadline to send the PDFs to their publisher for printing the next day for their main event of the year.

In response to this meltdown, we needed a completely fresh approach.

We knew from the get-go that our stack had to be fast and cheap. We love .NET and wanted to explore the brand new .NET Core. What attracted us to .NET Core was how lightweight and fast it is, and that it's cross platform. Our web portal and API key management is completely written in .NET Core. Our logs are stored in Azure Table Storage. We considered a Python + Flask stack, but quickly dismissed the idea.

To keep costs down, PDF generation had to be built on a serverless architecture. Our API endpoints are built in .NET on Azure Functions, and handles all of the incoming requests. These requests, should they be valid, are then forwarded on to AWS Lambda for PDF generation. AWS Lambda is the serverless architecture that allows us to scale to millions of requests at very low cost. The PDFs themselves are stored on Amazon S3.

When we began using this internally, OpenWater's costs went from $1000 per month to $60 per month and had no downtime whatsoever. We realized we built a solid product and decided to retool it so that any developer out there can use it, and that's why we launched Api2Pdf as its own company.

Our Api2Pdf customers tend to find us for one of two reasons. First, just trying to get any of the PDF generating libraries like wkhtmltopdf or Headless Chrome working in a cloud environment can ruin your day. And second, PDF generation is quite CPU intensive, and if you need to generate thousands of them, the costs to have a dedicated server soley for PDFs will skyrocket as what happened to us.

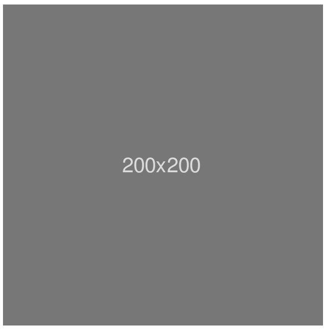
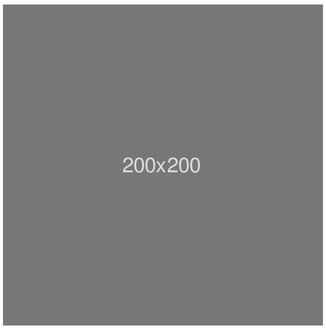
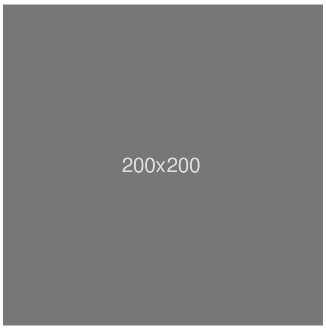
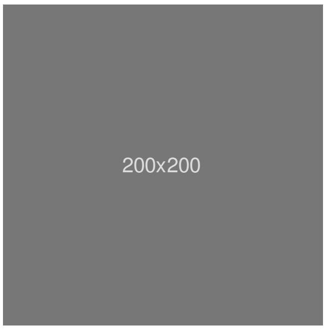
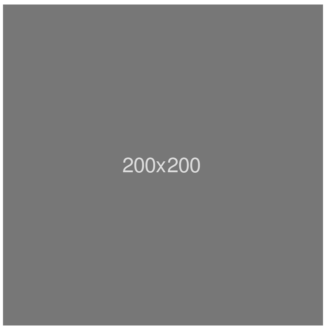
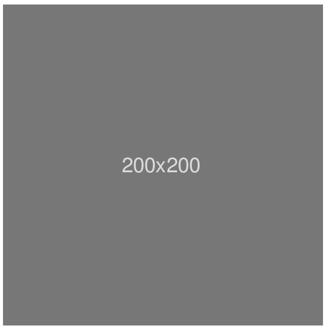
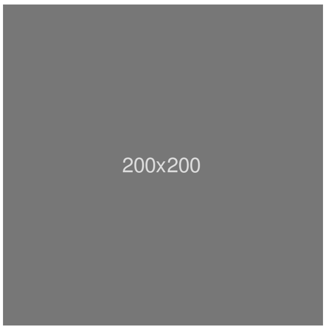
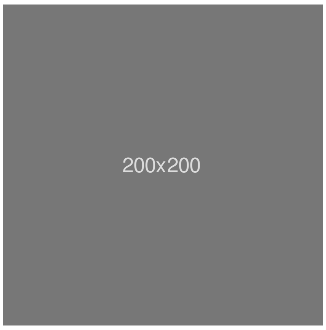
The most common use-case is to convert HTML to PDF for the purpose of printing invoices, event tickets, resumes, packing slips, etc, but we also provide endpoints for converting Microsoft Office documents to PDF and merging multiple PDFs together. We have such a wide variety of customers, ranging from online clothing stores to web design companies.

If you're a new .NET startup, using Microsoft Azure as your cloud hosting environment is a no-brainer. .NET Core is great, but it is also still very new and currently missing some key functionality. We decided it was safe to use since our web portal is a small app and relatively low risk. But the best advice I can give to new startups is that you just have to grind, day in, and day out. Chart your path and avoid distractions.

Api2Pdf is growing rapidly. We have five team members that contribute, all .NET developers. We are excited to make some noise is this oddly specific niche which is PDF generation. But it has been a lot of fun and we are learning something new every day.

Section III: Supplemental Materials

Images


200x200

200x200

200x200

200x200

200x200

200x200

200x200

200x200