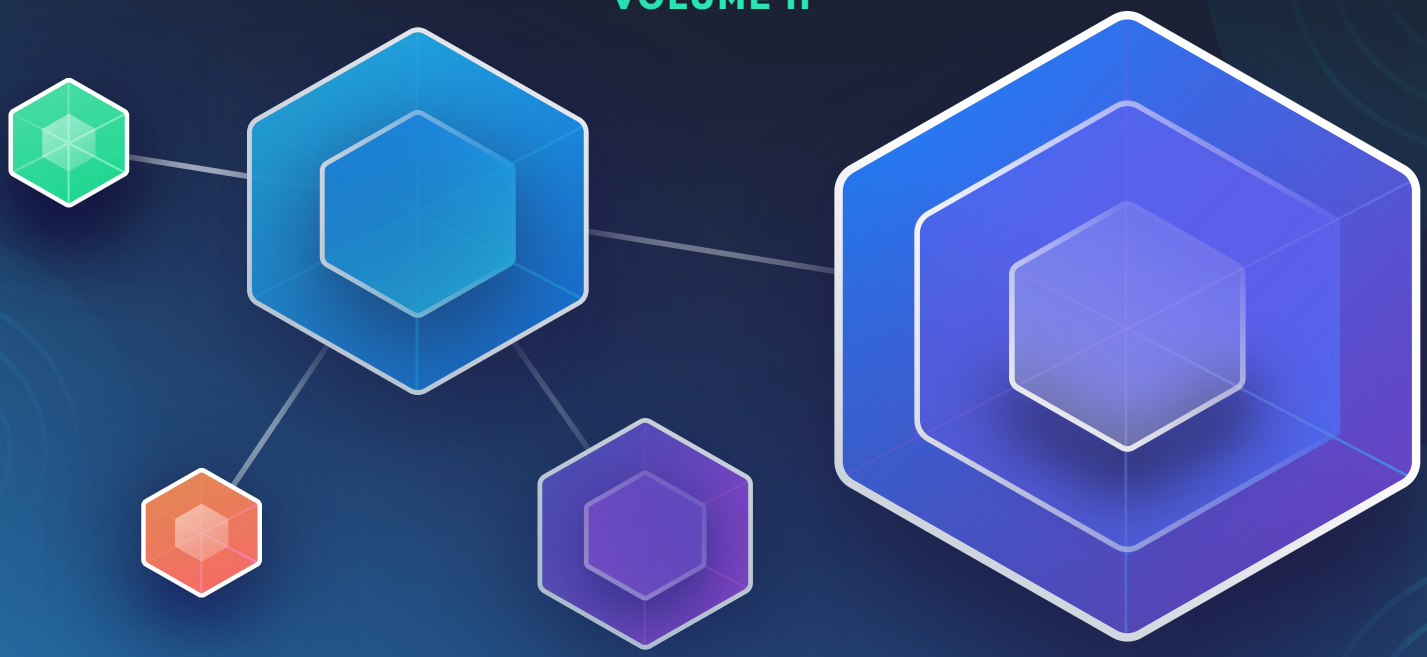
DZone
A DEVADA MEDIA PROPERTY

THE 2018 DZONE GUIDE TO

Microservices

SPEED, AGILITY, AND FLEXIBILITY

VOLUME II



RESEARCH PARTNER SPOTLIGHT

axway

Key Research Findings

BY JORDAN BAKER - CONTENT COORDINATOR, DEVADA

DEMOGRAPHICS

For the 2018 DZone Guide to Microservices, we surveyed developers, architects, and technologists from across the software and IT industry. We received 682 responses with a 79% completion rate. Based on these numbers, we've calculated the margin of error for this survey to be 2%. Below is a brief look into the demographics of our survey takers.

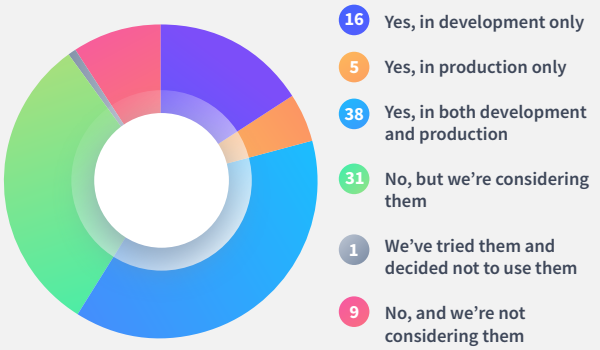
- 43% live in Europe, 23% in the USA, and 10% in South Central Asia.
- 43% work for companies headquartered in Europe and 32% work for US-based companies.
- Most survey takers work for enterprise-sized organizations.
  - 28% work for organizations sized 100-999 employees.
  - 18% work for organizations sized 1,000-9,999 employees.
  - 18% work for organizations sized 10,000+ employees.
- Respondents were split into three main job categories:
  - 37% work as developers/engineers.
  - 25% serve as developer team leads.
  - 20% work as architects.
- 82% work on developing web applications/services, 46% develop enterprise business applications, and 14% develop high-risk software.
- There were five main programming language ecosystems reported:
  - 85% Java
  - 70% JavaScript (client-side)
  - 38% Node.js
  - 34% Python
  - 33% C#
- The average respondent has 17 years of experience in the software industry.

THE WHO, WHEN, AND WHY OF MICROSERVICES

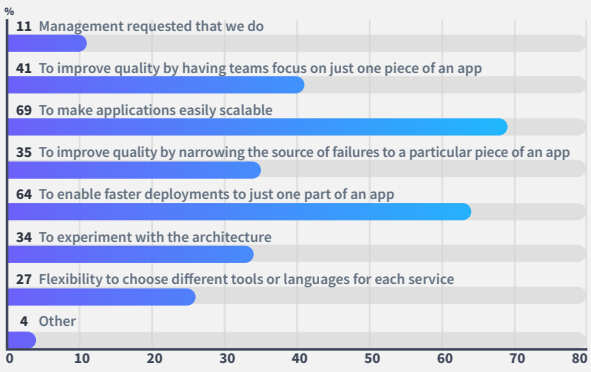
59% of respondents reported using microservices in some capacity, though where they implemented microservices in the SDLC proved somewhat variant. 38% use microservices in both development and production, while 16% use them in development only, and just 5% use microservices in production only. If we compare these numbers to the two most prominent types of developers in this survey (web app and enterprise business app), we see that using microservices in both development and production proved more popular among web developers. 43% of web app developers (versus 36% of enterprise business app developers) reported using microservices in both dev and prod.

Comparing the data regarding where in the SDLC respondents use microservices to the data regarding language ecosystems used by respondents, web development technologies come to the fore. 50% of respondents who work with the Node.js ecosystem reported using microservices in both development and production; this proved the highest percentage of any language ecosystem. Following Node.js, 47% of those who work with the Python ecosystem use microservices in both dev and prod, while 43% of client-side JavaScript developers and 42% of Java developers use microservices architecture in both environments.

ARE YOU CURRENTLY USING A MICROSERVICES ARCHITECTURE FOR ANY OF YOUR APPLICATIONS?



WHY ARE YOU USING MICROSERVICES?



Now that we know who is using microservices, and where in the SDLC microservices are prominent, let's examine why developers use this architectural pattern. At 68%, microservices' ability to make applications more easily scalable proved the most popular answer to this question among respondents. In a close second, 64% of survey takers told us that they use microservices to enable faster deployments to just one part of an application. These two uses of microservice architectures were by far the most popular, with a statistical differential of 23% separating enabling faster deployment from the third most popular use case, improving quality by having teams focus on just one piece of an app. The other notable uses, each chosen by about a third of respondents, were to improve quality by narrowing down the source of failures to a particular piece of an app (35%) and to experiment with the architecture (34%).

Interestingly, if we compare this data to the type of programming language ecosystems respondents use, we find that each language ecosystem lends itself to a different benefit of microservices. Among those working in the Java ecosystem, the most popular use case for microservices (chosen by 86%) was experimenting with architecture. For those working in the JavaScript ecosystem, improving by having teams focus on just one piece of an app (chosen by 72%) came out as the most popular option. For Node.js (45%) and Python (28%) ecosystem developers, making applications easily scalable won out.

Due to the benefits delineated above, 68% of respondents told us they feel microservices architecture has made their job easier, and 72% said they think the excitement around microservices is indeed warranted.

While a large majority of respondents enjoy working with microservices, we did have a small faction (99 respondents) who reported to be uninterested in microservices. The most notable reason being a lack of applicable use cases (58%). Other responses included a lack of knowledge on the subject (34%) and a lack of training (25%). All three of these critiques exhibit a rather interesting year-over-year pattern. While microservices detractors decreased from 125 respondents in 2017 to the 99 in 2018 noted earlier, the percentage of those who feel microservices has a lack of applicable use cases among this group went up by 19%. But also with the passing of a year, the percentage who claimed a lack of knowledge on the subject fell by 4%.

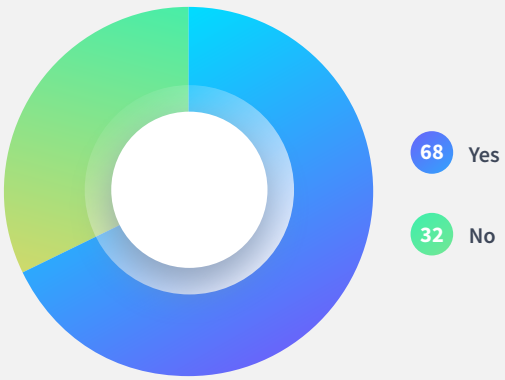
DEVELOPING MICROSERVICES

Among our general survey population, 70% (434 respondents) reported implementing DevOps processes such as continuous delivery. Of those 434 respondents, 39% told us they use microservice architecture in both development and production, 16% use microservices in development only, and another 6% use microservices in production only. Adding these all up, 61% of the 434 respondents who work with DevOps processes use microservices in some capacity when developing applications. The mean number of applications respondents are currently running with a microservices architecture came out to five; the highest number of apps reported was 50 and the lowest 0. And, of those respondents who have refactored legacy apps to take advantage of a microservice architecture, the mean number of refactored apps per respondent came out to two, with 15 being the highest number of applications reported.

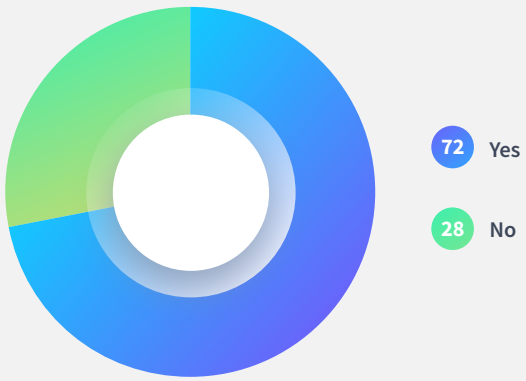
When asked for the language they feel best supports this type of microservices-based development, an overwhelming majority, coming in at 82% of respondents, said Java. The second most popular option was the JavaScript-based runtime environment, Node.js, with a 40% adoption rater among survey takers. And, with a 31% adoption rate, client-side JavaScript and Python also proved rather well-liked. If we compare the adoption rate of these languages to our data on the types of applications that respondents develop (and how respondents choose to secure their microservices), we get some intriguing results.

Among both web application (83%) and enterprise business application (85%) developers, Java proved the most popular language. Python, similarly, was statistically stable between these two user groups, with a 32% adoption rate among web app developers and a 31% use rate among enterprise business devs. When we get to the data on client-side JavaScript and Node.js, however, interesting fluctuations appear. While 45% of web app developers reported using Node.js, only 34% reported

HAS USING MICROSERVICES AS ARCHITECTURE MADE YOUR JOB EASIER?



DO YOU THINK THE EXCITEMENT AROUND MICROSERVICES IS WARRANTED?



using client-side JavaScript for their microservices; among enterprise business devs, 42% reported using Node.js, while only 28% claimed to use JavaScript on the client-side. This is interesting to note, as one of the main advantages of Node.js is the ability to code both server-side and client-side applications in the JavaScript language. And yet, we see far more respondents interested in using Node.js for their microservices-based development than client-side JavaScript.

When we asked respondents how they secure their microservices, 47% said JSON web tokens, 43% reported using OAuth 2, and 28% told us they implement user authentication. When we compare these numbers to the data on the top microservices-friendly languages, we find that JSON web tokens proved more popular among these respondents than among the general survey population. 55% of those who use Node.js for microservices development reported using JSON web tokens, 52% who use JavaScript use JSON web tokens, and 49% who use either Java or Python use JSON web tokens to secure their microservices. User authentication, too, proved more popular among users of these four languages than the general population, while OAuth 2's adoption rates witnessed far less fluctuation.

Despite the popularity of microservices, this architectural pattern comes with its own set of challenges. 58% of respondents reported that monitoring can present an issue when building apps with microservices. Fascinatingly, the second most oft reported challenge of building apps with microservices was changing culture to be open to microservices. 40% of respondents told us cultural shift presents an issue. Though other, more technical, challenges were reported — like changing API contracts (34%) and communicating between microservices (32%) — it appears that organizational structure has become a bigger roadblock to microservice adoption.

TOOLS FOR BUILDING MICROSERVICES

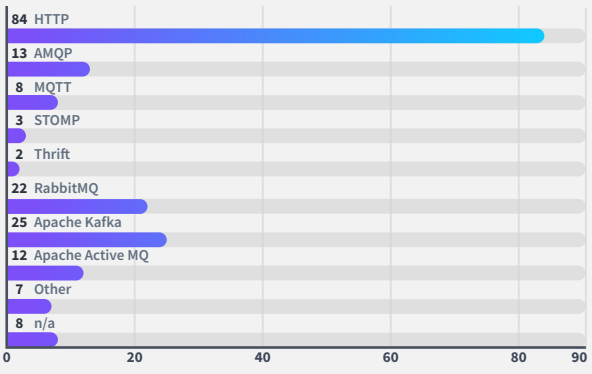
Given that 82% of respondents said they felt Java was one of the languages that best supports microservices, it comes as no surprise that Spring Boot (57%) and Java EE (22%) were the two most used frameworks/tools reported for building microservices. The platforms used to manage microservices once built, however, had a more even spread. 32% of those who microservices management platforms told us they use Istio, 18% use Kong, 17% use Conduit, and 13% used Linkerd.

Looking at the data for communication protocols, three main choices emerged: HTTP (84%); Apache Kafka (25%); RabbitMQ (22%). If we compare this data to the four largest programming language ecosystems delineated in the Demographics section (Java, JavaScript, Python, and Node.js), we find that HTTP remains the first choice for communication protocols among a large majority of respondents. Here's a breakdown of HTTP users per ecosystem: 84% Java; 86% Node.js.; 85% JavaScript; 84% Python. Apache Kafka proved more popular among Python ecosystem developers than anyone else, with a 33% adoption rate. Similarly, RabbitMQ saw a 30% adoption rate among Node.js ecosystem developers, an 8% increase over the general survey population.

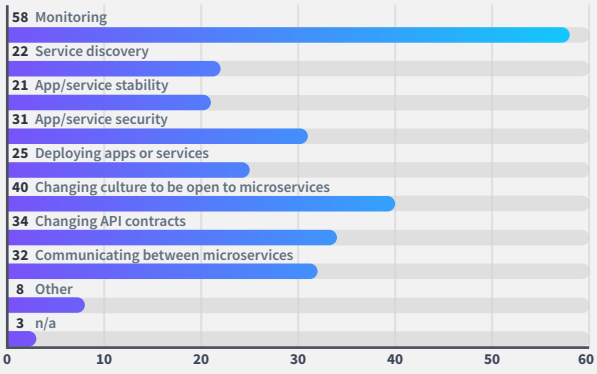
For the environments used to build, host, and deploy their microservices-based applications, over half of respondents (70%) use containers. Container technology factored in heavily in development, with 27% of respondents using containers only in the development stage and 35% using containers in both development and production. Among this group of survey takers doing microservice development in containerized environments, 51% use Kubernetes as their container orchestration tool. To delve deeper into the topics of containerization and container orchestration, head over to the *DZone Guide to Containers: Development and Management*.

Even with all these statistics, developers still appear split on the usefulness of tools and frameworks for microservices development. 60% of respondents told us they feel that tools and frameworks have provided sufficient best practices for working with microservices, with 40% feeling there is still work to be done on this front.

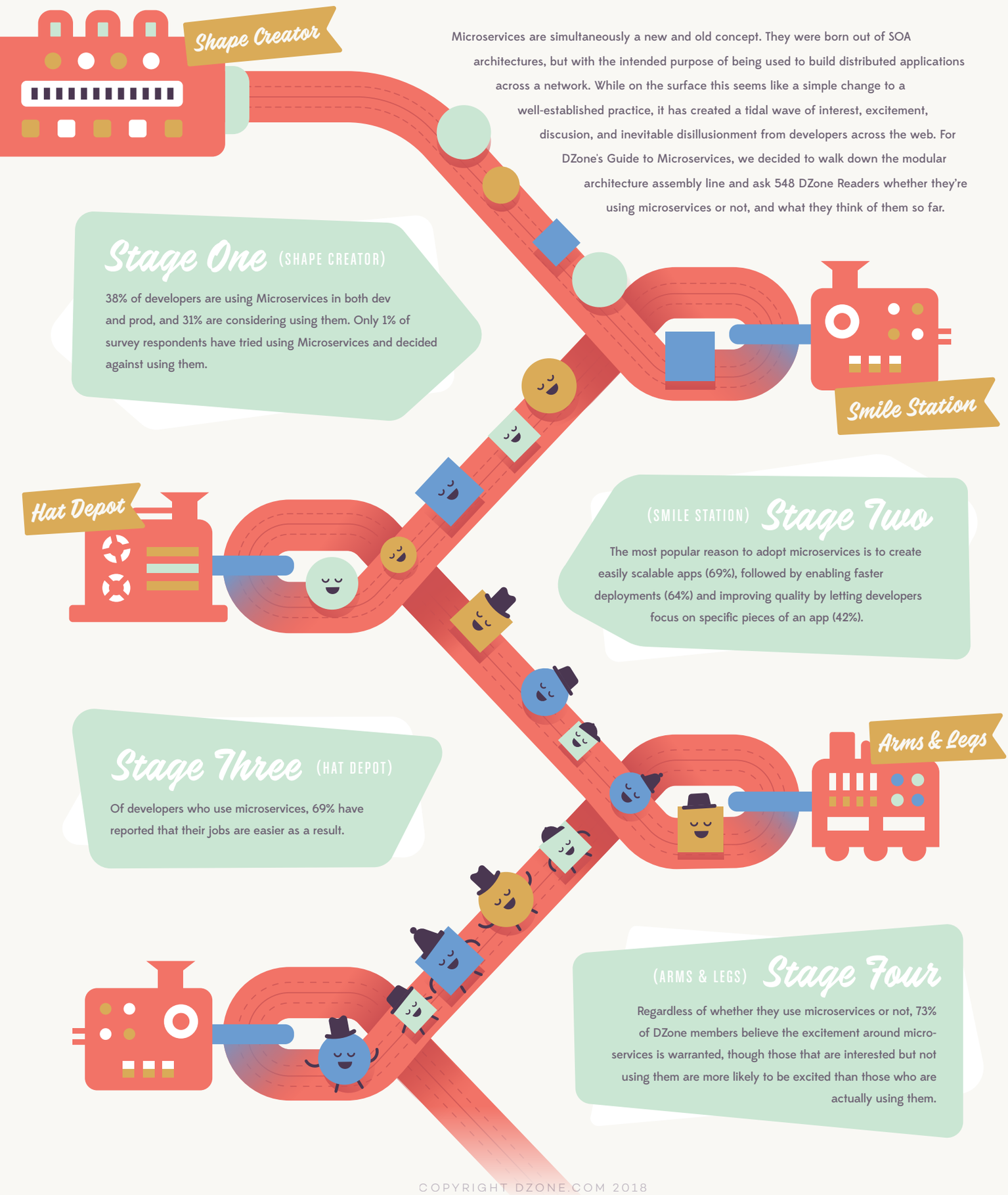
WHAT PROTOCOLS DO YOU USE TO COMMUNICATE BETWEEN SERVICES?



WHAT CHALLENGES DO YOU FACE WHEN BUILDING APPS WITH MICROSERVICES?



Big Things COME IN MICROSERVICES



Microservices are simultaneously a new and old concept. They were born out of SOA architectures, but with the intended purpose of being used to build distributed applications across a network. While on the surface this seems like a simple change to a well-established practice, it has created a tidal wave of interest, excitement, discusion, and inevitable disillusionment from developers across the web. For DZone's Guide to Microservices, we decided to walk down the modular architecture assembly line and ask 548 DZone Readers whether they're using microservices or not, and what they think of them so far.

Stage One (SHAPE CREATOR)

38% of developers are using Microservices in both dev and prod, and 31% are considering using them. Only 1% of survey respondents have tried using Microservices and decided against using them.

(SMILE STATION) Stage Two

The most popular reason to adopt microservices is to create easily scalable apps (69%), followed by enabling faster deployments (64%) and improving quality by letting developers focus on specific pieces of an app (42%).

Stage Three (HAT DEPOT)

Of developers who use microservices, 69% have reported that their jobs are easier as a result.

(ARMS & LEGS) Stage Four

Regardless of whether they use microservices or not, 73% of DZone members believe the excitement around microservices is warranted, though those that are interested but not using them are more likely to be excited than those who are actually using them.

diving deeper

INTO **MICROSERVICES**

twitter

 **@jessfraz**

 **@rhein_wein**

 **@b0rk**

 **@crichardson**

 **@simona_cotin**

 **@jldeen**

 **@christianposta**

 **@michelebusta**

 **@martinfowler**

 **@ThoHeller**

videos

What Are Microservices?

Learn what microservices are, why people are moving to microservices, and how you can create effective microservices.

Mastering Chaos – A Netflix Guide to Microservices

Get insight into how Netflix does microservices, what the anatomy of a microservice is, what cultural methods can help achieve microservice mastery, and more.

Introduction to Microservices, Docker, and Kubernetes

Learn about using microservices with two of the most popular container platforms out there.

zones

Microservices dzone.com/microservices

The Microservices Zone will take you through breaking down the monolith step-by-step and designing microservices architecture from scratch. It covers everything from scalability to patterns and anti-patterns. It digs deeper than just containers to give you practical applications and business use cases.

Integration dzone.com/integration

The Integration Zone focuses on communication architectures, message brokers, enterprise applications, ESBs, integration protocols, web services, service-oriented architecture (SOA), message-oriented middleware (MOM), and API management.

Cloud dzone.com/cloud

The Cloud Zone covers the host of providers and utilities that make cloud computing possible and push the limits (and savings) with which we can deploy, store, and host applications in a flexible, elastic manner. The Cloud Zone focuses on PaaS, infrastructures, security, scalability, and hosting servers.

refcardz

Getting Started With Spring Boot and Microservices

This Refcard will show you how to incorporate Spring Boot and Hazelcast IMDG into a microservices platform, how to enhance the benefits of the microservices landscape, and how to alleviate the drawbacks of utilizing this method.

Patterns of Modular Architecture

Covers 18 modularity patterns to help developers incorporate modular design thinking into development initiatives.

Getting Started With Microservices

In this updated Refcard, you will learn why microservices are becoming the cornerstone of modern architecture, how to begin refactoring your monolithic application, and common patterns to help you get started.

books

Building Microservices: Designing Fine-Grained Systems

Get examples and practical advice to help you build, manage, and evolve your microservice architectures.

Spring Microservices in Action

Learn how to build microservice-based applications using Java and the Spring platform.

The Tao of Microservices

Get yourself on the path to microservices with a conceptual view of microservice design, core concepts, and microservices applications.

Modernizing the Monolith: Accelerate Your Microservices Projects With Axway

Modernizing IT is about breaking down the monoliths of the past so the business can innovate faster and stay competitive and relevant in the future. Implementing microservices efficiently is essential to getting it done.

To build microservices that do one thing and do it well, you'll need to focus on making sure they are independently scalable, quickly deployable, and reliably consistent. Here are three key factors to keep in mind.

1. **Orchestration and mediation deliver scalability and high availability**
   In a microservice-based approach, each microservice owns its model and data so it will be autonomous from a development and deployment point of view. These kinds of systems are complex to scale out and manage, so you absolutely need a solution that lets you quickly orchestrate and mediate APIs if you want to have a production-ready application to target new channels or devices.

2. **Service mesh enables safe and reliable fast inter-service communication**
   To simplify service-to-service communication, use a configurable service mesh for service discovery, load balancing, encryption, authentication and authorization, support for the circuit breaker pattern, and other capabilities.

   By creating a dynamic infrastructure layer for communication at varying degrees of granularity, you can unlock the data trapped in your organization and mesh it up to deliver new business value.

3. **An API-first strategy offers speed and flexibility**
   Don't be afraid to start designing an API before you start implementing the microservice that will consume it. This approach accelerates projects by letting you validate your API design before investing too much in writing the actual microservice.

WRITTEN BY SHRAVANTHI REDDY
DIRECTOR, API AND APP DEVELOPMENT SOLUTIONS, AXWAY

PARTNER SPOTLIGHT

Axway AMPLIFY™ API Builder

Build, assemble, and deploy APIs and microservices faster with a low-code/no-code approach based on node.js/express

PRODUCT
Building microservices and APIs

OPEN SOURCE?
No

NEWEST RELEASE
Version 4.0

PRODUCT OVERVIEW

API Builder provides everything you need to rapidly create microservices or APIs accessing enterprise resources on-premises or in the cloud, quickly orchestrate and mediate them to target new channels or devices, and deploy them to your own private cloud no matter where it is running.

Use API Builder to:

- Design your APIs using a model-first approach or API-first approach
- Add value to existing APIs or create new APIs
- Read and write data to and from an external data source (such as MySQL, MongoDB, Oracle 12g DB, or in-server memory) using a library of free, open source, and enterprise-grade connectors
- Bake docker containers to deploy in any containerized platform

STRENGTH

- Simplifies orchestration and mediation with visual creation of custom reusable nodes
- Connects faster to enterprise and cloud services to expose their value
- Replaces monolithic deployments with a containerized environment
- Increases speed and frequency with an independently deployable solution
- Relieves the pain of scaling for each API

WEBSITE axway.com | **TWITTER** @axway | **BLOG** apifriends.com

axway

axway IMAGINE SUMMIT 2019

Accelerating the future. Together.

March 4-6 | Orlando, Florida
21-22 March | Chantilly, France

Modernize the monolith

Break up applications into microservices to innovate faster

- Quickly build microservices and APIs that drive business value
- Mobilize orchestrations as independently scalable microservices
- Target new channels and devices with easy and secure access to enterprise resources

AMPLIFY™ API Builder
TRY IT FOR FREE

axway.com/api-builder-free-trial