Top Tools for Cost-Effective Web Development

Learn how to build your next web product with minimal effort and optimal return
Introduction

How Web Products Are Made

Everything Starts With DevOps

What's DevOps

Best DevOps Technologies

Application Containerization: Docker
Automation Technology: Jenkins
Configuration Management: Ansible
Data Collecting: Logstash
Data Processing: Graylog or Kibana
Package Management Automation: Artifactory or Nexus

Behind the Curtains: Building the Back End of Your Product

Server Language: Java or JavaScript
Java Framework Of Our Choice: Spring Boot
Ways to Build Reactive Java Apps: Vert.x, Ratpack, Reactor and Akka
Object/Relational Mapping: Hibernate and Spring Data
Performance Acceleration: Varnish
Microservices’ Front Door: Netflix Zuul

Ensuring Smooth Interaction With Best Front-end Technologies

JavaScript's Swiss Army Knife: Angular 2
The MVC Veteran: Backbone
Load it Heavily: React + Flux
Go Full Stack: Meteor
Implementing Design: Bootstrap and Materialise

Bottom Line
Introduction

In today’s world of advanced software technologies, a minimum viable version of any new web solution is released within 2-4 months after the development begins.

But we know how hard it is to choose the technology for your product development, get the right balance between popular and proven tools, and to see the difference between the best and the optimal application architecture.

To avoid huge overhead both in expenses and efforts required for developing and updating an unoptimized product, read this eBook and learn:

- how modern web and mobile products are made nowadays
- why it’s so critical to decide which software architecture and product roadmap first
- what’s DevOps; back-end and front-end of the product, and how to approach them correctly
- things you should know before proceeding to choose the technology stack
- what the most cost-effective technologies for building highly reliable enterprise and user-oriented web services and products are in 2016-2017

Let’s start!
Everything starts with software architecture. Limited to a set budget, you and your development team should first decide how to make a robust and scalable solution.

The business model of developing an app.
Once you've done it, the development team comes up with a **product roadmap outlining the steps and stages required to be completed**, as well as the estimates for the project.

Such planning is crucial for the complete development process. Without a defined architecture, developers might make fatally wrong choices, come up with unexpected functionality, and finally, release your product much later than you expected; not mentioning the money you will have spent unwisely.

**To get a solid product, you must ensure the right architectural approach behind its DevOps, back-end, and front-end parts.**

Let's talk about them in more detail.
Everything Starts With DevOps

What’s DevOps

In short, DevOps is a product deployment practice where developers (‘Dev’) and system engineers/operators (‘Ops’) collaborate to automate the process of software delivery and infrastructure changes.

The basic idea behind it is applying the Agile methodologies not only to product development, but to its releases and updates as well. Such an approach was eventually developed into widely used DevOps practices named Continuous Delivery.

Product Delivery Cycle with Devops

- The team plans the next product version
- The team implements the required changes
- The developers submit many code updates and deliver them via continuous integration practice
- Then the build is deployed to production
- The updated product starts operating
- The team gathers the feedback about the updated product
They allow automation of update delivery to a production environment and ensure product stability at all times through automated testing. As a result, it becomes possible to update your web application with a single button once you’re ready to go.

Eventually, developers can integrate the written code into a shared repository up to several times a day. Then such code will be automatically tested and verified to help find and address any possible bugs and problems as early as possible. Such an approach is also known as **Continuous Integration**.

As a result, **DevOps engineers bring the following advantages to the product development process:**

- **Continuous Integration**
- **Risk-reduction**
- **Continuous Delivery**
- **Easier scalability**
- **Technical support**
- **Performance monitoring**
Best DevOps Technologies

In our practice, we turned to the next DevOps technology stack to ensure the most cost-effective product development for our clients:
Application Containerization: Docker

Instead of putting an app into a resource-consuming virtual machine, Docker allows placing it into a small isolated container with everything you may need to run it: system tools, libraries, runtime environment, etc. So the same hardware can run more apps than before.

For developers the Docker containers give instant application portability, are easy to deploy in a cloud, and can be incorporated into other DevOps apps, such as Ansible or Puppet.

While Docker is not the first and only offering on the market (now there’s Rocket and Canonical’s LXD, for instance), it has seen major popularity growth lately, with the usage numbers almost doubling within the first few months of 2016.

Automation Technology: Jenkins

Jenkins is the core tool allowing automated testing of the submitted builds. Being extremely flexible, it can be used as a continuous integration server or a continuous delivery hub for relatively small to medium projects (in rare cases of huge projects it is worth switching to Drone).

**BENEFITS:**

› thanks to many plugins you can extend Jenkins to make it do anything you want

› It can be distributed across multiple servers and platforms

› it is flexible, written in Java and free, which made it so popular among the majority of developers all over the world
Configuration Management: Ansible

When you have many servers and applications to manage, version-control, and test, Red Hat’s Ansible helps you take care of all that. Its main advantage is it can be used in any repeatable process, whether it’s application deployment or server infrastructure provisioning.

Ansible connects to the nodes and pushes simple Python scripts called Ansible Modules to them via SSH. Once they are finished executing, these modules are then deleted. Such an approach makes Ansible incredibly simple and not requiring any additional infrastructure (like daemons or servers).

It is free, easy to learn and use, and can be easily version-controlled. Ansible turns out to be the most cost-effective option on the market for its type.

Data Collecting: Logstash

Logstash is a tool helping to collect and access data from different sources for further processing. And since data usually comes in different formats, Logstash is also able to normalize and help you make the most of it.

As a log and event collector, Logstash unites all data into a single stream and then forwards it according to algorithmic if-then statements. Such an approach is easier to comprehend, especially for procedural programmers.

Logstash is also part of the hugely popular Elastic stack, allows extensibility through its API, and shows high performance by being able to process over 10,000 events per second; which makes it one of the most appealing tools for event and log management.
Data Processing: Graylog or Kibana

Once the required data is normalized with Logstash, it must be processed and visualized in order to see trends and get analytics needed to move on and make important business and technical decisions. For this circumstance, we recommend using Kibana or Graylog.

Kibana is another part of the Elastic stack providing an intuitive interface and deep analytics and visualization possibilities, easy to use for even non-dev users. A comprehensive and customizable log analytics dashboard helps to quickly find, filter, and analyze any amount of data, which is why it is used by companies like Netflix and Linkedin.

Graylog has a simpler interface, but includes alert mechanisms and allows management of user access to the defined data streams, which turns out to be essential in some cases. So which data processing tool is the best depends on the peculiarities of your project. (Besides, you can always install and use both of them.)
Package Management Automation: Artifactory or Nexus

However unique your web product is, it most definitely includes functionality that was once implemented, tested, and checked by other developers. And each day the offered number of such plugins, extensions, libraries, and other third-party elements, either open-source or proprietary, keeps growing.

Using them helps to cut the development time, but the more you use them in your project, the more efforts you must exert to manage and keep an eye on these binaries, their up-to-dateness, and compatibility. This is where Artifactory and Nexus come in handy.

In short, they work as your local proxy proving access to the components your build tools require. However, Artifactory is much more functional: it can cache the needed third party elements (called ‘artifacts’) locally; the whole process of artifact management can be automated via a RESTful API; indexing and search capabilities are also much wider.

On the other hand, Nexus is easier to install and set up, and the more recent 3.0 version comes with a fresh look and a new integration API, which makes it a great option for smaller projects.
Behind the Curtains: Building the Back End of Your Product

The back end is the foundation of your web product where its core functionality is defined and implemented. It is a server (or a bunch of servers) hosting and running your application and database.

Back end developers are responsible for all the things you cannot see when interacting with the product through the user interface:

- how the data entered is stored by the application
- how users are registered and authenticated
- how the application communicates with third-party services through the API
- how the high level of performance is achieved
- how the offered security level is reached
- how the product’s business logic is implemented and much more.

Solid web applications require 60-70% of development efforts to be spent on implementing its back-ends, which is why choosing the right technology can have a dramatic effect on your product’s costs.
Server Language: Java or JavaScript

The most important technology you should choose first is the server language. It will be used to implement most of your product’s core functionality and you will be unable to quickly switch to another language if required.

While there’s a big choice of server languages, when it comes to proven reliability and cost-effectiveness, most companies choose Java or JavaScript.

The main difference between them is that Java is a full-feature general-purpose OOP language, while JavaScript is a scripting language primarily created to bring interactivity to web interfaces.

Java has been considered the best option for big or enterprise applications requiring long-term development efforts. JavaScript, which can run on the server side with Node.js, usually is the option to consider for smaller projects; where it turns out to require ~3x less development effort than Java.

While both approaches are cost-effective, and the final choice depends mostly on your product requirements and developers’ expertise; nobody writes any code in Java or JavaScript from scratch today. Instead, developers use frameworks.
A framework is often a skeleton of an app providing developers with many generic elements required in most apps. Thus, using frameworks increases the development speed and dramatically decreases the effort required to achieve the necessary result.

Since JavaScript is used in front-end as well, we'll get back to JS frameworks later, and focus on Java technologies now.

**Java Framework Of Our Choice: Spring Boot**

Spring is one of the most popular Java frameworks coming with many sister projects to address the modern needs of businesses. It follows the very popular MVC architectural pattern, handles transactions, takes care of dependency injections, and more.

In fact, some might say it is overloaded with possibilities, which is why it requires deep expertise and concerted effort to set it up and make it run. To cope with these circumstances, Spring Boot was introduced.

As a suite of already configured tools, it allows developers to launch a new Spring web app with a minimum of written code. No version collisions, no need for XML, production support, and many more features make Spring Boot an essential tool for any Java-based project.
Ways to Build Reactive Java Apps: Vert.x, Ratpack, Reactor, and Akka

If your app is going to be reactive (i.e. event-driven, responsive to user’s behavior, resilient, and scalable), Java on its own won’t be enough here as it is not a “reactive language”. Although it is expected to support reactive models in version 9 (which is to be released in the middle of 2017), at the moment you'll need to use a Reactive layer on top of the JDK.

The cost-effective tool we had the most experience working with is Vert.x, a server framework built in Java. In its essence, it is very similar to Node.js: both of them provide an event-based programming model, offer asynchronous API, and start single-threaded lightweight servers instead of containers.

Vert.x sports a concurrency model allowing creation of multiple threads, aims at reusability with its module system, and provides an event bus to make different programs communicate with each other effectively.

Although it is written in Java, you can use this framework to write applications in any language of your choice. Such design allows it to be much more flexible and show performance multiple times better than that demonstrated by Node.js code, which is why many developers eventually turn to Vert.x.
Other, more up-to-date options leverage Reactive Streams, a handful of Java interfaces expressing the basic building blocks of Subscriber and Publisher, and thus forming a language that is common for interoperable libraries:

1. **Ratpack.** Being a set of libraries made of Netty, Reactive Streams, Java 8, and Google Guava, it allows autoconfiguration for embedding it into Spring applications. Ratpack brings up and listens to an HTTP endpoint instead of using the embedded servers provided by Spring Boot.

2. **Project Reactor.** This Java framework is an implementation of the Reactive Streams Specification and was initially developed by the team that made Spring (recommended above). Its IO modules provide wrappers around Netty, Aeron, or any other alternative low-level network run-time.

3. **Akka + Lagom/Play.** Also having Reactive Streams built in, Akka is a toolkit for building apps using the Actor pattern in Java. Lightbend, the company behind this open-source tool, also offers Lagom, a defined framework allowing build of distributed systems with microservice architecture; and Play, a high velocity, stateless and non-blocking framework built on top of Akka that consumes minimum hardware resources for delivering highly scalable applications.

Akka + Lagom/Play is great for huge distributed systems. Ratpack is lightweight, does not limit you with libraries and tools you can use for composing your app, and will be a nice fit for most fast and evolvable web solutions. Project Reactor is a bit more sophisticated, defined, and advanced than Ratpack, and is used for building applications with extremely high throughput demand and low tolerance for latency.
Object / Relational Mapping:
Hibernate and Spring Data

ORM is a programming approach allowing developers to avoid writing tedious SQL statements whenever applicable, focus on implementing the product's business logic, and generally make the code more readable and understandable. ORM frameworks make it possible to be independent of the chosen RDBMS and its language.

For the Java programming language, use Hibernate or Spring Data.

Hibernate is the ORM framework that enables data conversion between the DB and Java via its own Annotations. It's a free and open source Java Persistence API (JPA) implementation that has a solid community of developers and offers many ways to develop apps fast (database independency, caching, type and parameter handling, etc.). Using it in your project in addition to other technologies, when applicable, may significantly affect the development speed of your team.

Spring Data JPA is a JPA data access abstraction. Currently it's a more advanced ORM tool, allowing skipping of a lot of manual work for RDBMS or other data repositories. It simplifies access to persistence stores by providing generic interfaces for them, and, unlike with Hibernate, you also get automatic handling of Pageable queries, base classes for standard CRUD repositories, and more.
Performance Acceleration: Varnish

In case your product is going to be highly dynamic and content-heavy, you’ll need an open source HTTP accelerator like Varnish.

As an intermediary between Apache servers and web requests, it helps to reduce the load on the web server while increasing the website performance. In its essence, Varnish works as an HTTP proxy that stores the cache of previous HTTP responses for future use. To speed things up even more, it stores its cache in memory, not on disk. Along with other optimizations, it allows processing loads of requests at the blink of an eye.

Microservices’ Front Door: Netflix Zuul

When you’re developing a web solution based on the microservice architecture, it might be a challenge to present your system consumers with a single interface where many app parts are interacting with each other. To address this issue, Netflix offers its open-source Zuul proxy server.

Being an edge service, it plays the role of a front door that allows any user interface to consume services from multiple hosts. Zuul can also be used for routing rule management and load balancing in your system.

Recently Netflix announced the release of Zuul 2, which will be asynchronous and non-blocking.
Ensuring Smooth Interaction With Best Front-end Technologies

The front end part of the web application is everything the user is expected to see and operate with when interacting with your product.

Now-a-days the front end functionality of many web services is extremely broad: it manages and shows massive amounts of various data, and allows animations and media-rich interactions.

Add here the front end design, requiring HTML/CSS developers to implement the look and feel of your service exactly as the designer has imagined it, and make it compatible and responsive for hundreds of mobile devices, browsers, and Internet-enabled devices, and you’ll see why front-end development is now treated so seriously.

To implement any kind of interactivity and animation on the front end of a web product, you’ll need JavaScript. But writing any functionality in pure JavaScript will require fantastic efforts, which is why developers have come up with many frameworks and libraries helping to significantly speed up the process in different ways.
JavaScript's Swiss Army Knife: Angular 2

Angular 2 offers a huge community and is the choice of many development teams aiming to speed up the product delivery process as quickly as possible.

However, use Angular only in mid-scale projects, as its performance starts to drop when the project becomes heavier. More severe problems may appear as well on huge projects.

The MVC Veteran: Backbone

Backbone has been on the market for a while, and some might say it has better alternatives. But it's still an excellent and tried framework or, rather, an MVC structure allowing developers to build web solutions with extremely high performance.

If combined with Marionette and Underscore, it's a powerful tool with a great selection of features helping to simplify the development process. Another possible approach is to use exoskeleton.js to turn your Backbone structure into a surprisingly lightweight and fast framework.

Backbone also turns out to be a smart choice if you're planning to deliver a supplementary mobile application to your service, such as using it with Apache’s Cordova, which will allow development of an app that will be almost as fast as native.
Load it Heavily: React + Flux

While React is not a framework, but a library simplifying the developer's management of DOM elements with a virtual structure, combining it with the Flux architecture (both released by Facebook) makes it the best option for large-scale projects with a lot of custom functionality written from scratch.

In fact, using React + Flux is also our most used approach for developing enterprise projects due to its ability to ensure the requirements for reliability and uptime.

Go Full Stack: Meteor

Meteor is a unique framework written using Node.js that allows developing both the back end and the front end of the app in JavaScript. While due to its limitations it is suitable mostly for small scale projects, it's a great option for front-end developers who don't want to dive deeply into understanding the server side, but need to implement additional back-end functionality to the site's interface.
Implementing Design: Bootstrap and Materialise

To describe the presentation of a web product in code, we use CSS and HTML. However, developers may work more effectively by using such frameworks as Twitter’s Bootstrap and Materialize.

Bootstrap contains HTML-and CSS-based templates for designing typography, navigation, buttons, forms, and other interface elements. You can also extend it with jQuery plugins.

Materialize is a similar framework that was introduced somewhat later than Bootstrap. So its design templates are based on a more modern Material design and support only more up-to-date browsers.

While Bootstrap provides you with a grid that you can fill with elements and pre-configured CSS classes that you can override, Materialize turns out to be more particular on how the UI elements must look, and thus is more constraining. Still, both are great free tools to speed up front-end development and save effort on routine tasks.
Bottom Line

Rapid development of the Web makes modern markets extremely fast-paced and dynamic. To succeed with a new web project, you should not only have a proven business idea and a great team of experts, but also pick the most cost-effective tools and make the best architecture choice. It will help you quickly release any web product without sacrificing its quality, reliability, and most importantly, its further development speed.

For almost ten years we have been helping enterprise companies to develop and maintain web-based solutions and mastered a modern technology stack with the best potential and perspectives. We define cost-effective reliability and security as the №1 priority, and our experience helps us to evaluate all possible technology options for project realization.

If you’re looking for a team of developers to take the best advantage of modern technologies and deliver your next web product in the most cost-effective manner, contact us. With our 14 years of experience and 135 experts, we will surely find the way to help you succeed.