



**Politecnico  
di Torino**

**Politecnico di Torino**

**Master Degree in Communication and Computer Networks Engineering**

Master's Degree Thesis

# **Integrating DevOps checklist and pilot sample project**

**Supervisor:**

**Prof. Maurizio Morisio**

**Candidate:**

**Klementin Hasani**

**Co-Supervisors:**

**Daniele Sabetta**

**Matteo Ricardo Bonfanti**

OCTOBER 2022

## Table of Contents

Acknowledgment	vi
Chapter 1	1
1. Introduction	1
1.1 Thesis Objective	1
Chapter 2	3
2. DevOps	3
2.1 How DevOps Developed	3
2.2 The nine Key Stages	5
2.3 DevOps Pipeline	6
2.3.1 Components of a DevOps pipeline	6
2.3.2 Solutions for DevOps practices	7
2.4 Automation in DevOps	7
2.4.1 Software for DevOps automation	8
2.5 Tools Ecosystem	9
2.6 Companies Worldwide	9
Chapter 3	15
3. Azure DevOps	15
3.1 Azure Boards	15
3.2 Azure Pipelines	16
3.3 Azure Repos	17
3.4 Azure Test Plans	17
3.5 Azure Artifacts	18
3.6 Azure DevTest Labs	18
3.7 Application Insights	18
3.8 Application Life Cycle with Azure	20
Chapter 4	22
4. Agile Manifesto	22
4.1 The 4 Agile Values	22

4.1.1	<i>The 12 Agile Principles</i>	24
4.2.	<i>Scrum</i>	24
4.3.	<i>Work Item Process</i>	27
<b>Chapter 5</b>		<b>29</b>
5.	<i>Integrated Infrastructure</i>	29
5.1	<i>The key to IT Success</i>	29
5.2	<i>Achieving Infrastructure Integration</i>	29
<b>Chapter 6</b>		<b>30</b>
6.	<i>Cloud adoption</i>	30
6.1	<i>Kubernetes</i>	33
6.2	<i>Kubernetes &amp; Docker</i>	36
<b>Chapter 7</b>		<b>37</b>
7.	<i>Git</i>	37
7.1	<i>Git Flow</i>	38
7.2	<i>Trunk based development</i>	39
7.3	<i>GitHub, GitLab, One Flow</i>	40
7.4	<i>Top Git Hosting Services for 2022</i>	40
<b>Chapter 8</b>		<b>43</b>
8.	<i>To DevOps OR NOT to DevOps?</i>	43
<b>Chapter 9</b>		<b>44</b>
9.	<i>Introduction</i>	44
9.1	<i>Building process</i>	45
9.1.1	<i>Qualification</i>	45
9.2	<i>Shopping Project Overview</i>	48
9.2.1	<i>Microservices 1 – Shopping MVC Client Application</i>	49
9.2.2	<i>Microservices 2 – Shopping API Application</i>	53
9.2.3	<i>Microservices 3 – MongoDB Database</i>	55
9.2.4	<i>Azure Kubernetes Services deployment</i>	65

## List of Figures

Figure 1: DevOps Lifestyle	1
Figure 2: DevOps	3
Figure 3: “10+ Deploys per Day: Dev and Ops Cooperation at Flickr” presentation & John Allspaw (right person) and Paul Hammond (left person)	4
Figure 4: DevOps Stages	6
Figure 5: DevOps Tools Ecosystem 2021	9
Figure 6: HP platform	11
Figure 7: Etsy’s strategy	11
Figure 8: Benefits of Netflix after cloud migration	13
Figure 9: Benefits of Hackerone	14
Figure 10: Azure DevOps	15
Figure 11: Final Step of Pipeline creation	16
Figure 12: Overview of the selected application	19
Figure 13: Application lifecycle management	20
Figure 14: Agile	22
Figure 15: Agile Principles	24
Figure 16: Scrum Methodology	25
Figure 17: Work Item Process	28
Figure 18: Cloud & Devops	30
Figure 19: Cloud Computing	31
Figure 20: Biggest Cloud Computing Provider	32
Figure 21: Serverless Computing Architecture	33
Figure 22: Kubernetes	33
Figure 23: Kubernetes 6 Levels	34
Figure 24: Kubernetes Cluster	35
Figure 25: Docker Architecture	37
Figure 26: Combining Docker and Kubernetes	37
Figure 27: Git	38
Figure 28: Top Git Hosting 2022	41
Figure 29: DevOps & NoOps	43
Figure 30: Orbyta logo	45
Figure 31: Orbyta List Projects	46
Figure 32: Project creation	46
Figure 33: Web Application Structure	48
Figure 34: Shopping project structure	49
Figure 35: GitHub repository	50

Figure 36: Solution Explore	50
Figure 37: Dockerfile code	51
Figure 38: DockerHub push	52
Figure 39: shoppingapp-web service	53
Figure 40: Update github	53
Figure 41: Shopping.API Swagger	54
Figure 42: Shopping.Client Website	54
Figure 43: MongoDB.Driver	55
Figure 44: MongoDB added	56
Figure 45: docker-compose-override.yml & docker-compose.yml	57
Figure 46: PowerShell – docker ps & docker images	58
Figure 47: PowerShell – docker images	60
Figure 48: shoppingapi repositories – push successfully	61
Figure 49: shoppingclient repositories – push successfully	61
Figure 50: PowerShell – kubectl get pod	64
Figure 51: Step of the process from local to cloud	65
Figure 52: Structure of ACR DevOps Pipeline	65
Figure 53: Creation of resource group - command line	66
Figure 54: Azure Container Registry	66
Figure 55: tag image containers	67
Figure 56: ACR repository	67
Figure 57: Azure Kubernetes Service creation – command line	68
Figure 58: Pull-Secret image	68
Figure 59: shopping.client & shopping.api deployment section	69
Figure 60: : shopping.api & shopping.client service section	69
Figure 61: : Kubernetes resource: PODs, Services, Deployments, Replica Sets	70
Figure 62: shopping.client webpage – External IP	70
Figure 63: shopping.client webpage – Update V2	71
Figure 64: Automate scenario process	72
Figure 65: Shopping API - pipeline	75
Figure 66: Shopping Client – pipeline	78
Figure 67: Services on AKS	78
Figure 68: Shopping Client webpage – The last update	79

## List of Tables

<i>Table 1: Six Scrum principles</i>	26
<i>Table 2: Scrum Guide</i>	27
<i>Table 3: Kubernetes Components</i>	35
<i>Table 4: The best host Git Repository</i>	41
<i>Table 5: Fill up the project</i>	47
<i>Table 6: Configure Pipeline shoppingapi</i>	74
<i>Table 7: Configure Pipeline shoppingclient</i>	77

## Acknowledgment

I would like to acknowledge and give my warmest thanks to my supervisor **Prof.** Maurizio Morisio who made this work possible. His advice carried me through the final stage of writing my project. I would also like to thank my co-supervisors Daniele Sabetta and Matteo Bonfati for their precious guidance and support throughout the whole process. I have benefited greatly from their incredible expertise and experiences, and I am confident that what I have learned from them at this time will be very helpful to me in my future career.

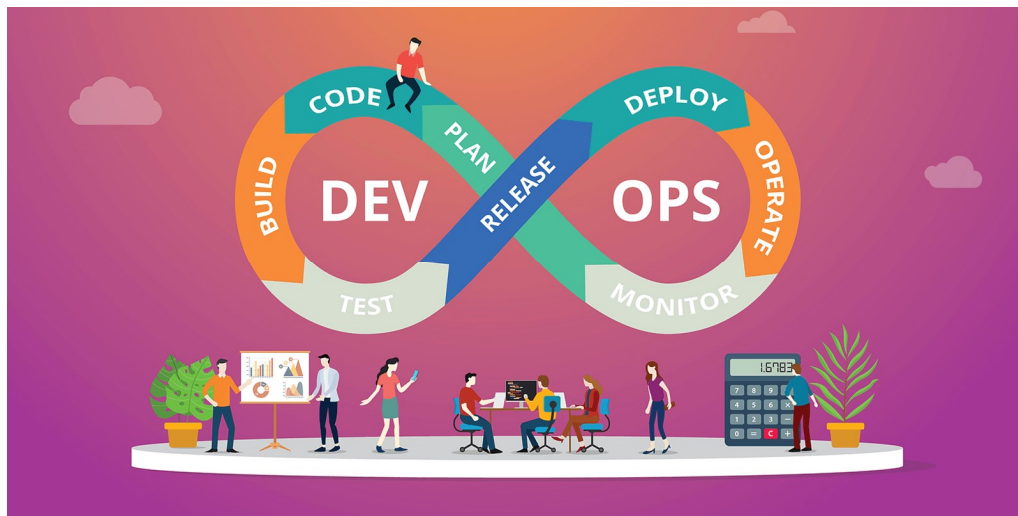
My sincere gratitude also goes to all of the other Orbyta company employees for their outstanding contributions and cooperation over the length of the six months.

Last but not least, I want to express my gratitude to my parents and my girlfriend for their unwavering love and support during every phase of my Master's degree pursuit, from the beginning to the final.

# Chapter 1

## 1. Introduction

DevOps is a trending topic that is popular for increasing company productivity, no matter your industry. Every day more companies work to bring this disruptive model to their organizations. DevOps has one primary goal to achieve: continuous integration to continuous delivery.



*Figure 1: DevOps Lifestyle*

So, the development and operations processes become faster and more resource friendly. Companies can save money while producing more high-quality software products for customer consumption or internal use. Ultimately, a DevOps engineer brings a software solution from inception to completion by seeing the big picture and helping everyone involved in the project work together.

### 1.1 Thesis Objective

This thesis was developed at **ORBYTA Tech** company.

**ORBYTA Tech** is the **ORBYTA** group company that is specialized in IT consulting in the software applications and IT systems field. With a staff of about 200 technicians and experts in IT, it realizes highly complex projects with the most modern technologies and exploits the most innovative methodologies. The company fully manages the life cycle of software products, from the design, implementation, delivery, integration and application maintenance phases of complex software and hardware systems. Thanks to the experience and the vast know-how of the business units,



the company offers a 360 ° consultancy from UX / UI to database development, with everything you can meet in between.

In the context of current technological trends, ORBYTA is undergoing a process of continuous evolution and adoption of the most modern means to pursue a high level of automation (*hyper-automation*) of processes, both internal and at the service of customers.

**Hyper automation** is an approach with which business activities identify, control and automate processes, exploiting cutting-edge and intelligent technologies. It is used to manage a large number of processes with speed and is essential to orchestrate different resources, tools and platforms. The aim is to drastically reduce operating costs and effectively lead companies towards a conscious digitalization.

Similarly, in software development, automation has always been the key driving factor. There have always been efforts to automate mundane and repetitive tasks across SDLC. It has both a human and business perspectives to it. Smart IT engineers do not find it intriguing to spend hours doing a manual code review or test an application manually, rather they would find interesting ways to automate these monotonous and routine tasks. Even from business perspective, it improves the efficiency and hence speeds to market.

As technologies advanced and the pace of DevOps adoption picked up, it started becoming evident that it's not possible for one technology to automate a complete process that a single human can do. There are multiple disparate system interfaces and human interventions that are needed. Thus, the idea to combine different tools that will enable automating tasks which couldn't otherwise be automated earlier came into play. Hyper automation became mainstream after it got its mention in Gartner Top 10 Strategic Technology Trends starting from 2020.

At *ORBYTA Tech*, **DevOps** must become an approach to culture, automation, and platform design designed to deliver greater value and responsiveness to your business through efficient, high-quality service delivery. The methodology put in place brings together members of the operations and development teams into a single distributed team. This allows you to take ideas and projects from development to production more quickly and efficiently. The methodology involves more frequent code changes and more dynamic use of infrastructure than traditional manual management strategies.

The aim of the thesis is to describe the DevOps software process development and management in place and to create a shared standard to spread DevOps knowledge and best-practices for all ORBYTA Tech business units.

In the following sections we will explain the basic concepts of DevOps, including how we got here, best practices, key topologies, and the most common benefits of a DevOps environment and how *ORBYTA Tech* is implementing the methodology in its process and within the organization.

## Chapter 2

### 2. DevOps



***Figure 2: DevOps***

DevOps is a new philosophy that can help software organizations innovate faster and be more responsive to business needs. It promotes collaboration between developers and operations, which improves quality of software deployments and more frequent software releases.

Adopting the DevOps philosophy requires a new mindset, new tools and new skills. Collaboration is a tenet of DevOps practice and philosophy. DevOps is not a software or programming language. It is also about collaboration between DevOps and other parts of the business.

On DevOps, there are three overarching stages executed in a logical order:

- Build
- Test
- Deploy

From the code that is build, we go to test it and if everything goes well, deploy it. It is important to understand the lifecycle stages, which will create efficiency, reliability, speed, and agility.

#### 2.1 How DevOps Developed

Before the year 2000, most IT industries adopted the classical waterfall model, a linear approach for software development.

- Developers had to spend a lot of time developing and integrating heavy pieces of code.
- QA engineers and operations teams, who worked in silos, spent more time testing the code.

And at the end of day, the result was a large, sometimes years-long gap between software releases, with frequent bug fixes and software patches deployed between each release. With the establishment of the Agile software methodology, IT industries moved on to developing software iteratively and frequently released them into production. Continuous Integration and Continuous Delivery are among the major techniques adapted in this model for the rapid delivery of software. DevOps consequently promoted the smooth collaboration between development and operations teams at each step of cycle. So, we can safely say that DevOps has its roots in the Agile methodology. The concept of DevOps emerged out during an Agile conference held in Torino, Canada. A man by the name of Andrew Shafer tried to put together a meetup session entitled “Agile Infrastructure.” When Patrick showed up for the session, he was the only one there. Andrew had received so much negative feedback from his posting that not even he showed up to his own session. However, Patrick was so excited to learn of a like-minded person that he hunted him down at the conference and that talk in the hallway. They formed a discussion group for other people to post their ideas for how to solve this divide between development and operations later that year.

Initially, the interest was pretty tame and not a whole lot came of it. In June of 2009, John Allspaw and Paul Hammond gave a talk “10+ Deploys a Day: Dev and Ops Cooperation at Flickr.”



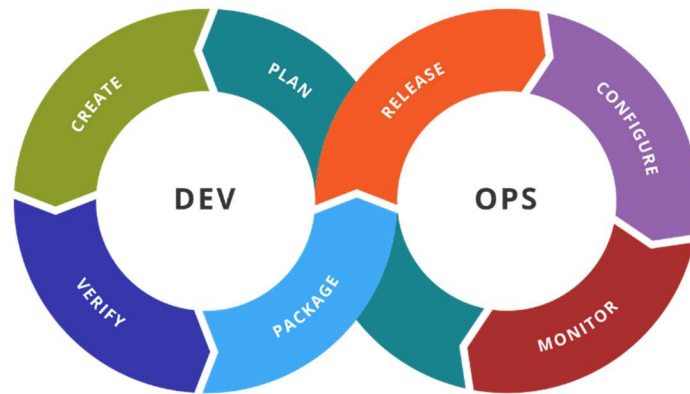
***Figure 3: “10+ Deploys per Day: Dev and Ops Cooperation at Flickr” presentation & John Allspaw (right person) and Paul Hammond (left person)***

Our friend, Patrick happened to watch the streaming video of that presentation, and it instantly resonated with him. He realized this was exactly the solution for which he had been looking. He put out a call to have a gathering of developers and system administrators to get together and

discuss the best ways to start bridging the gap between the two disparate fields. In October 2009, the event DevOps Days named by him, garnered a fair amount of attention from experts in both fields and sparked lively debates over Twitter. It was not long before some of the smaller tech enterprises were attempting to put together DevOps practices as well as tools built to aid these newly forming teams. Better communication and understanding would also help teams to recognize the priorities of each other. And all of these benefits would mean growing productivity and high-speed delivery. DevOps is that one logical change the IT industry needed badly.

## 2.2 The nine Key Stages

- a) **Plan** is the stage of DevOps that create a product roadmap. This will help the team organize resources and make priorities for the upcoming stages.
- b) **Create** is the first stage where you start to code, run tests (CI) and deploys a new version of the application (CD). This is one of the keys to improve velocity because multiple developers can act at the same code base.
- c) **Verify** is the stage focused on code quality, security testing, parallel execution and automation. The possibilities that developers find and fix errors while they are developing has proven to be more cost-effective and efficient.
- d) **Package** stage happens to be after code has been designed and tested. It stores the software in a state where it can be reused later.
- e) **Release** is the moment when DevOps deploys the software to end users.
- f) **Configure** is the stage where DevOps manages infrastructure and software platforms. Automated configuration management is created to take care of these complex environments, networks and storage systems.
- g) **Monitor** is an important stage as the DevOps sees the impact of the software on infrastructure and users. It also provides data to respond to incidents which this increases security, agility and reliability.
- h) **Protect** is about securing your applications and the infrastructure that software is running from interferences.
- i) **Manage** stage close the loop and is about feedback and control across your end to end software development lifecycle.



**Figure 4: DevOps Stages**

The benefit of a united platform is the ability to manage and control the entire software development lifecycle from one place. The security is part of every stage of the process.

### 2.3 DevOps Pipeline

A DevOps Pipeline is a set of automated processes and tools that allows developers and operations professionals to work cohesively to build and deploy code to a production environment. Since there is not one standard DevOps pipeline, an organization's design and implementation of a DevOps pipeline depends on its technology stack, a DevOps engineer's level of experience, budget, and more. A DevOps engineer should have a wide-ranging knowledge of both development and operations, including coding, infrastructure management, system administration, and DevOps toolchains.

#### 2.3.1 Components of a DevOps pipeline

**C integration/C delivery** (CI is the step that enables iteration by committing changes to a shared source code. It is all about efficiency. / The release of newer or modified code into production is automated by CD.)

**Continuous testing** (incorporates automated, prescheduled, continued code tests as application code is being written or updated Version Control system allows developers to record changes in the files and share them.)

**Agile planning development** (Like version-control mechanisms. Organizes work in short iterations to increase the number of releases. This allows for flexibility and pivots once the ideas are tested on an early product increment. Engineers commit code in small chunks multiple times a day for it to be easily tested.)

**Infrastructure as code** (Allows operations teams to monitor environment configurations, track changes, and simplify the rollback of configurations. Kubernetes/ open shift /docker)

**Configuration management** (Define the state of each system. CM is most described as the automation, management and maintenance of configurations at each state.)

**Continuous monitoring** (monitoring both the code in OPS and the underlying infrastructure. Issues makes its way to back to development.)

### 2.3.2 Solutions for DevOps practices

- CI/CD: Define a pipeline and manage releases with multiple environments with Azure Pipelines. Automate with GitHub Actions. Extend or simplify CI/CD with Jenkins plug-ins. Target any service including Azure Kubernetes Service on Azure. Create fast and repeatable deployments with Spinnaker.
- Agile: Manage projects with GitHub and use Azure Boards to define, assign, track work items or manage backlogs. Also get advanced analytics and reporting.
- Version Control: Manage git repositories, share and collaborate with GitHub.
- IaC: Define cloud resources with Azure Blueprints and use open-source tools such as HashiCorp Terraform and Ansible.
- Configuration management: Manage resource configuration with Ansible, Chef, Puppet and Azure Automation.
- Monitoring: Monitor infrastructure health and integrate into existing dashboards in Grafana, Kibana or Azure portal with Azure Monitor. Use built in container monitoring for AKS.

The above practices allow team members to create a DevOps environment based on a collaborative structure. A DevOps environment leads to an increase in visibility and a decrease in the risk of uncertainty. As a result, the increase in visibility not only improves communication between teams, but also increases time-to-market. A DevOps environment not only increases visibility and reduces uncertainty, but also improves the process of detecting and addressing errors, bugs and other issues. There are no more divisions or barriers; instead, each issue, bug and business requirement is everyone's responsibility. This environment can also help reduce bottlenecks and eliminate waste in the development process. It is estimated that by 2019, 1 in 3 organizations could have a DevOps practice. This means that, today many organizations are in the process of building one.

### 2.4 Automation in DevOps

With the rapid growth of technology sector software, development teams are under constants pressure to meet the increased customer expectations for business applications. DevOps and automation are two key components that help organizations streamline the development process. Automation, which cuts time and money spent on repetitive tasks and eliminates human errors, streamlines the whole DevOps process. DevOps teams are increasingly looking to

automate processes throughout the development and deployment lifecycle. This promotes speed and increases deliveries, and deployments. There is no need to worry about the importance of human because in fact automation minimizes dependency on humans from managing a lot of tasks. This automation results in several key improvements:

- Removes manual errors
- Dependency removed
- Latency removed
- Increases number of deliveries
- Reduces the lead time
- Provides faster feedback
- Team members are empowered
- Enables speed, reliability and consistency

#### **2.4.1 Software for DevOps automation**

Plenty of software options are available. Both open-source and licensed tools support end-to-end automation of a DevOps pipeline. Among them, CI/CD tools are the most common type of tools.

Puppet and Chef are solid cross-platform configuration management tools. These tools deal with infrastructure management, automating the configuration, deployment, and management of infrastructure.

Jenkins, TeamCity, and Bamboo are CI/CD software that automates tasks starting from development pipeline to deployment.

Specialized software and tools focus on a single function that is a crucial part of the DevOps pipeline, for example:

- Source code management: Git, CVS, Subversion.
- Infrastructure provisioning: Ansible, Terraform, Vagrant.
- Application monitoring: SonarQube, Nagios.
- Containerized applications: Docker, Kubernetes.
- Log management: Splunk, Datadog, SolarWinds Log Analyzer.
- Security monitoring: Snort, Splunk.

You can combine all these tools to create a comprehensive automated DevOps lifecycle.

## 2.5 Tools Ecosystem

# DevOps Tools Ecosystem 2021

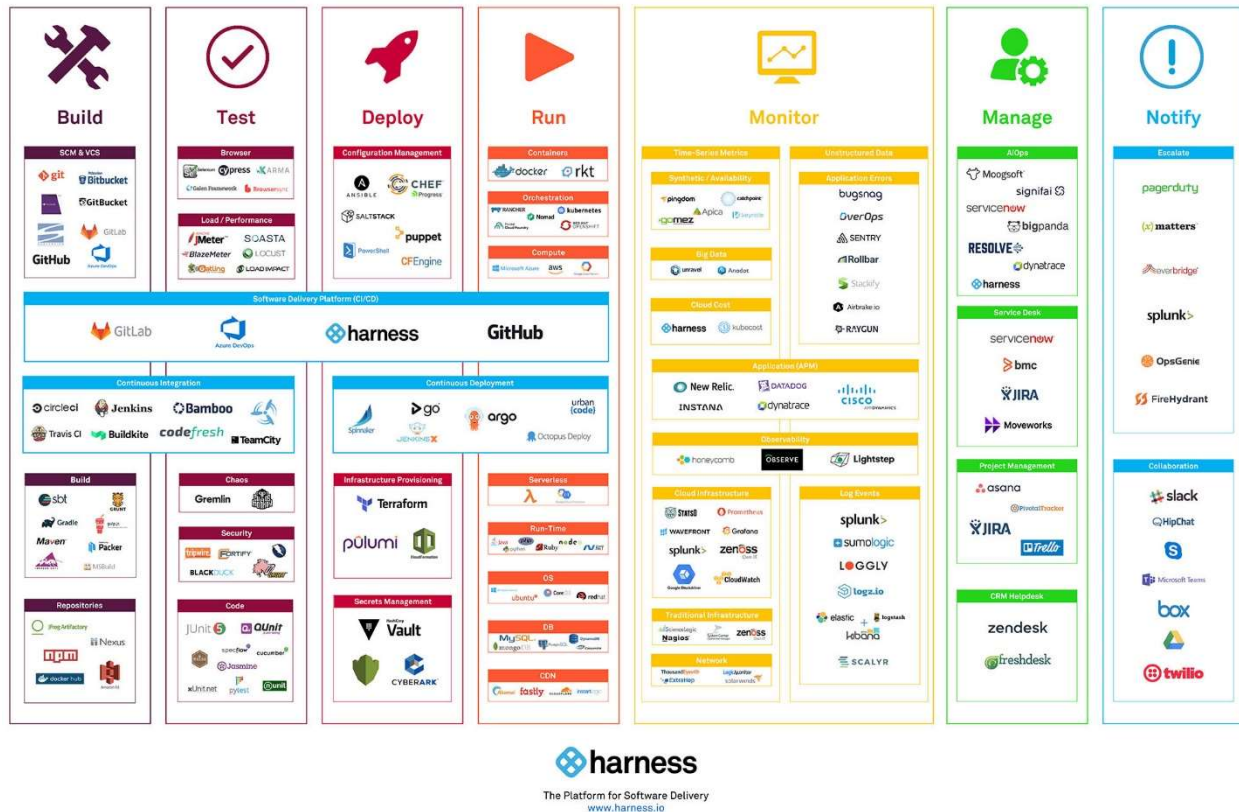


Figure 5: DevOps Tools Ecosystem 2021

## 2.6 Companies Worldwide

Companies using DevOps are undergoing a serious culture shift. The following list of Companies using DevOps have seen triumph:



- United Airlines, Inc. is a major American airline. This company changed its traditional method of testing to **continuous testing** using DevOps which helped the company to save \$500,000. It also increased its coverage of code by 85%.





- Facebook helped change the way we think about software development. Many of the tenets it adopted early on, including code ownership, incremental changes, automation, and continuous improvement, were DevOps in all but name. Its approach has matured over the years, and it recently migrated its entire infrastructure and back-end IT to the Chef configuration management platform. Facebook's accelerated development lifecycle to reshape consumers expectations of software. Its recently announced bi-weekly app updates effectively served notice that constant, rapid refreshes for mobile apps are the new normal, and any company that cannot keep up risks getting left behind.



- HP company faced a problem regarding testing their software. Here bugs are detected using manual testing after six weeks of writing code and if there are any bugs found it would take one week to fix them. So to overcome this issue they integrated **the Continuous Integration and Continuous Deployment pipeline**. As you can see in the given figure below, their first step was to create a common platform to support all the products and models. This was called continuous integration which eliminated toil caused by the integration of different code branches.

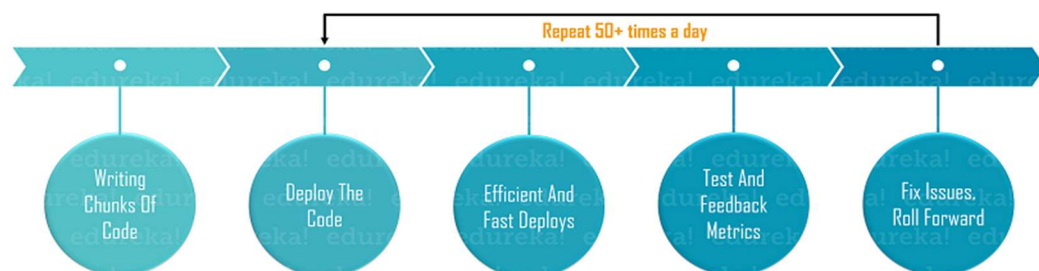


**Figure 6: HP platform**

They also built a set of automated unit tests running against the trunk, which reduced the 6 weeks manual test time, thereby improving product quality and inducing faster feedback. Within these automated tests, HP uses a tool called “Stopped the line” that alarms the developer when code breaks any of the unit tests or builds. *These DevOps practices-led around 100 to 150 code commits and 75,000 to 1,00,000 lines of code changes in a single day.*

## Etsy

- Etsy is one of the earliest Companies Using DevOps. It is an American e-commerce website focused on selling handmade and vintage supplies. Initially, Etsy struggled with the development of their organization because they adopted monolithic architecture. Their deployment rate was about two times a week resulting in the isolation of departments, so it had to find a way out of this traditional system. The new Chief Technology Officer brought in a team to adopt DevOps practices. The CI/CD pipeline helped in deploying services about 50 to 100 times a day. In code deployment, Etsy uses Deployinator a tool that offers one click-deployments. They also use Amazon Web Services to perform their DevOps operations.



**Figure 7: Etsy's strategy**

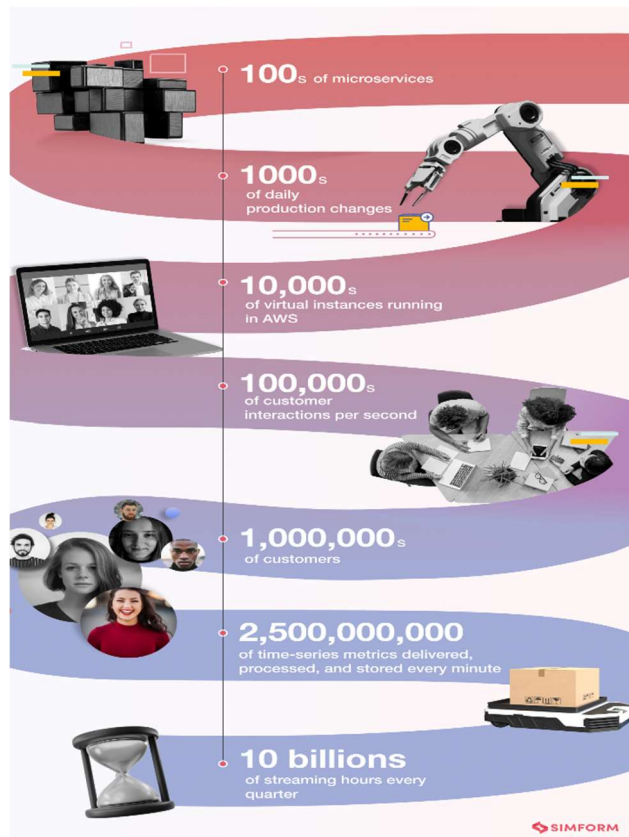
The entire testing phase can thus be automated with the help of a very famous Continuous Integration tool called Jenkins. Jenkins now executes more than 14,000 test suits per day.



- Back when Amazon was still run-on dedicated servers, it was a constant challenge to predict how much equipment to buy to meet traffic demands and pad estimates to accommodate for unforeseen traffic spikes. As a result, about 40 percent of Amazon's server capacity was wasted. Once the online retailer moved to the Amazon Web Services cloud, it allowed engineers to scale capacity up or down incrementally. But it also spurred a transition to a continuous deployment process that allows any developer to deploy their own code to whichever servers they need, whenever they want. Within a year of Amazon's move to AWS, engineers were deploying code every 11.7 seconds, on average. The agile approach also reduced the number and duration of outages.

## NETFLIX

- When Netflix evolved its business model from shipping DVDs to streaming video over the web, it waded into uncharted waters. There were not any commercial tools available to keep the company's massive cloud infrastructure running smoothly, so it turned to opensource solutions. After many suggestions from developers, it created the Simian Army, a suite of automated tools that stress test Netflix's infrastructure and allow the company to proactively identify and resolve vulnerabilities before they impact customers. It prompted Netflix to move to the cloud and give their infrastructure a complete makeover. Netflix chose AWS as its cloud partner and took nearly seven years to complete its cloud migration. They decided to rewrite the entire application in the cloud to become truly cloud-native, which fundamentally changed the way the company operated. As a significant part of their transformation, Netflix converted its monolithic, data center-based Java application into cloud-based Java microservices architecture. As a result, it helped Netflix accelerate innovation and stumble upon the DevOps culture. After completing their cloud migration to AWS by 2016:



**Figure 8: Benefits of Netflix after cloud migration**



- Adobe's DevOps transformation began five years ago when the company moved from packaged software to a cloud services model and was suddenly faced with making a continuous series of software updates rather than big, semi-annual releases. Adobe uses Cloud Munch's end-to-end DevOps platform to automate and manage its deployments. Because it integrates with a variety of software, developers can continue to use their preferred tools, and its multi-project view allows them to see how a change to any one Adobe product affects the others. The move has enabled faster delivery and better product management.



- In 2015, Adidas released the first of its Yeezy sneakers, which were designed in collaboration with Kanye West. Demand for the original shoe was enormous and was great for business, but it presented a huge challenge for IT. The site crashed when new Yeezy were announced. Inside the company, developers moaned that they were helpless to fix things and that it could take up to a week to get a simple virtual machine spun up. Adidas underwent a massive transformation that embraced cloud-native architecture, Kubernetes, and DevOps and that involved a wholesale cultural shift. Today, you can find [Adidas' DevOps Maturity Framework](#) on GitHub.

## hackerone

- This company gives organizations access to the largest community of hackers on the planet, with a presence in more than 70 global locations. They adopted GitLab to eliminate disparate toolchains and shift security left. Hacker One achieves 5x faster deployments with GitLab's integrated security. The benefits of company:



***Figure 9: Benefits of Hackerone***

Also: Cost Saving, Improved trust in software, single source of truth and less context switching.

## Chapter 3

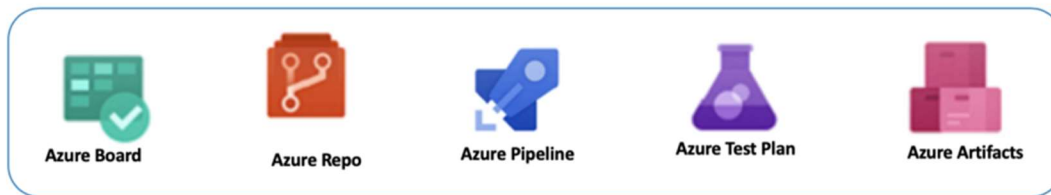
### 3. Azure DevOps



*Figure 10: Azure DevOps*

**Azure DevOps** is a Software as a service platform from Microsoft that provides an end-to-end DevOps toolchain for developing and deploying software. It comprises a range of services covering the full development life cycle.

At the time of writing these are:



*Figure 11: Azure services*

#### 3.1 Azure Boards

Azure Boards is a service for managing the work for your software projects. It provides a variety of choices for planning and managing work. Example:

- Each project comes with a pre-configured Kanban board perfect for managing the flow of your work. Boards allow you to add the columns you need for each team and project, support swim lanes, card customization, conditional formatting, filtering and even WIP limits.
- Backlogs help you keep things in order of priority, and to understand the relationship between your work. Drag and drop items to adjust the order, or quickly assign work to an upcoming sprint.

- Sprints give you the ability to create increments of work for your team to accomplish together. Each sprint comes equipped with a backlog, task board, burndown chart, and capacity planning view to help you and your team deliver your work on time.
- Dashboards: Azure Boards comes complete with a rich canvas for creating dashboards. Add widgets as needed to track progress and direction.
- Queries let you tailor exactly what you are tracking, creating easy to monitor KPIs. It is one of the most powerful features of Azure Boards.

### 3.2 Azure Pipelines

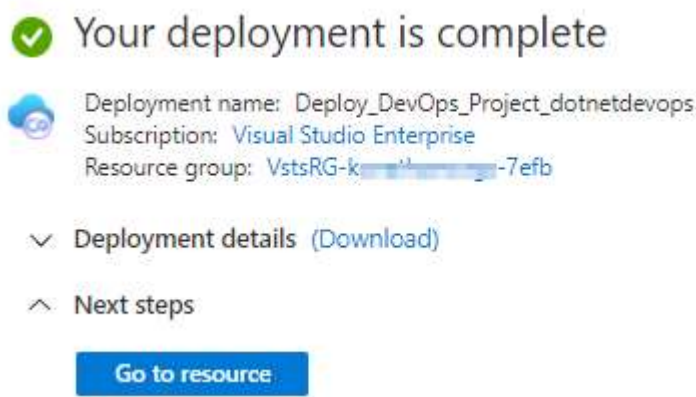
Azure Pipelines is an automated set of processes that helps developers to compile, build and deploy codes on other computation platforms. The goal of this pipeline is that there is no manual intervention, all the changes are automatically executed in the project. A pipeline is normally broken down into categories:

- a. Source Control
  - b. Build Tools
  - c. Package creation
  - d. Configuration management
  - e. Monitoring
- **Steps of Creating a Build Pipeline**

Creating a team project with sample .NET code repository.

Created build and release pipelines to compile, test and deploy the application. A release pipeline is a process by which we take committed code into production.

Created Azure Web App and Azure SQL database in Azure.



***Figure 11: Final Step of Pipeline creation***

### 3.3 Azure Repos

Azure Repos is a set of version control tools that you can use to manage your code. Azure Repos provides two types of version control: Git and TFVC. Free for first 5 users with unlimited private repositories. The concepts of Azure Repos:

- **Repository** is the location for our code.
- **Branch** is a lightweight reference that keeps a history of commits and provides a way to isolate changes for a feature/bug fix from our main branch.
- **Branch policies** is an essential part of the Git workflow. Protect the critical branches in our development.
- **Pull and Clone:** Create a local copy of an existing Git repo by cloning it.
- **Push and Commit:** We can share these changes (commit) to the remote repository by pushing.
- **Fork:** It is a complete copy of repository, including all file commits, and branches.
- **Git:** It is a distributed version control system.
- **Notification:** Using notification, we will receive an email whenever any changes occur to work items, code reviews, source control files, pull requests and builds.
- **Projects:** A project provides a place where a group of people can plan, track progress, and collaborate on building software solutions.
- **Teams:** A team corresponds to a selected set of project members.

### 3.4 Azure Test Plans

Azure Test Plans is a service which provides a browser-based test management solution for exploratory, planned manual, and user acceptance testing. It also provides a browser extension for exploratory testing and gathering feedback from stakeholders. Now Azure DevOps Test Plans can be used for both Automated and Manual testing. In modern software development processes, everybody in the team contributes to or own quality. Let's take a look:

- **Testing is integral to DevOps and Agile teams:** Based on tests of user stories, features, or scenarios that are managed on a Kanban board as in Azure Boards. A team can leverage manual testing right from within their Kanban board. This provides end-to-end traceability. Developers and testers can use this capability to maximize quality within their teams.
- **Quality is a team sport through exploratory testing:** Exploratory testing is an approach to software testing that is described as simultaneous learning, test design and test execution. The Test and Feedback extension enables exploratory



testing techniques in Azure Test Plans. It helps you to spend more time finding issues, and less time filling them.

- **Planned manual testing for larger teams:** it lets you organize tests into test plans and test suites. Test suites can be dynamic to help you understand the quality of associated requirements under development, or static to help you cover regression tests. Testers execute tests assigned to them using a runner to test your app(s), in a browser or as a client on your desktop. To track overall process and outcomes, leverage lightweight charts, which can be pinned to your dashboard for easy monitoring.

### 3.5 Azure Artifacts

Azure Artifacts is a package management solution integrated into Azure DevOps that allows you to create and share Maven, npm, and NuGet packages via feeds that can be both public and private to an organization with teams of any size. It enables developers to consume and publish different types of packages to Artifacts feeds and public registries. An alternative to build your own solution path would be to leverage a SaaS solution such as Azure Artifacts which is included in Azure DevOps.

### 3.6 Azure DevTest Labs

Azure DevTest Labs is a service that Microsoft Azure provides functionally for managing environments that contain Azure Virtual Machines. It enables you to manage costs by setting things like maximum number of VMs per lab and per user, allowed VM sizes and VM auto-startup and auto-shutdown times. The main reason for a developer to use DevTest Labs is that it provides self-service. You can create the VMs that you need, provision and de-provision them and have everything ready to develop and perform tests on. No more waiting on IT operations.

### 3.7 Application Insights

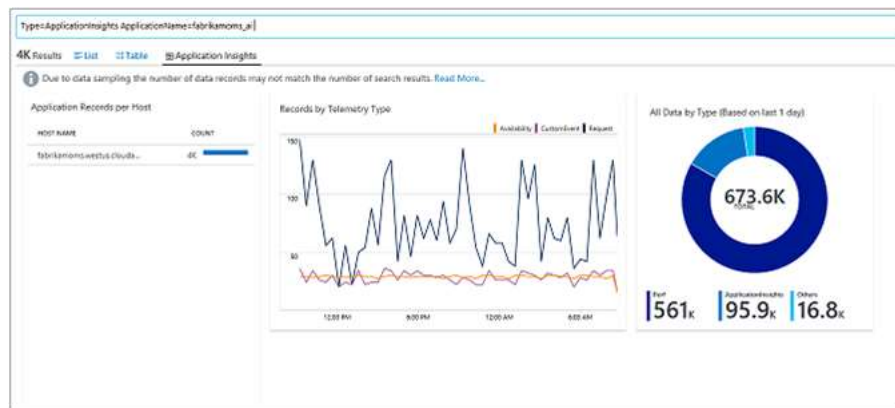
Application Insights is an application performance management service for web applications that enables you to do all the monitoring of your website performance in Azure. Developers and DevOps can use Application Insights to:

- Help diagnose issues by using powerful analytics tool.
- Automatically detect performance anomalies.
- See what users actually do with apps.

- Help continuously improve app performance and usability.

How Application Insights works is you insert a small package to your application and set up the Application Insights resource with Azure, thus sending the telemetry data to Azure to collect information. The web app is monitored and it sends data to the Insight portal. You can pull in your host environmental data, allowing you to look at performance logs, Azure diagnostics and container logs, giving you a full look at what is going on inside the application, as well as in the environment where it lives. You can set up periodic web test that will allow you to send requests to the web server to ensure that it is responding properly and that the website is working the way it is supposed to.

Some of things you can track or collect are:

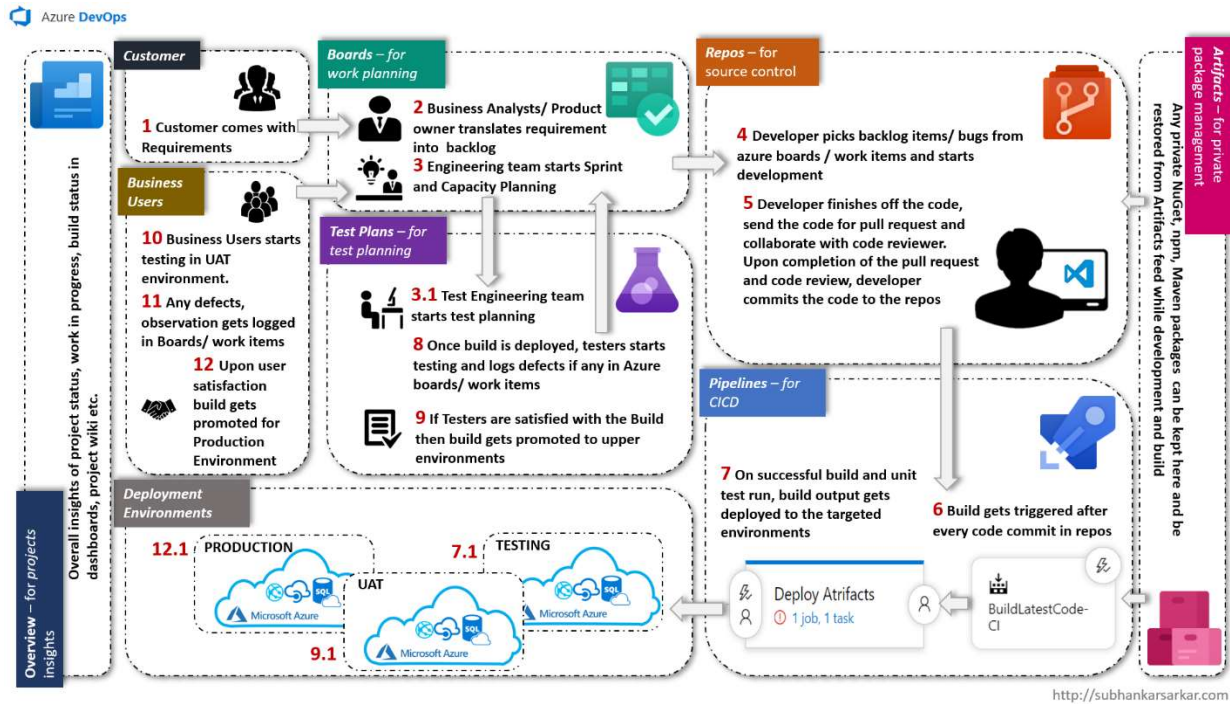


**Figure 12: Overview of the selected application**

- Performance and host diagnostics, which gives you a complete picture of what is happening in your application.
- Trace logs for correlating trace events with requests to help you get a deeper insight into the data and dig deeper into the diagnostics to improve performance.
- You check the most popular webpages in your application, at what times of day and where users are.
- Also check the exactly time and the source position that traffic is coming.
- Exceptions for both server and browser information, as well as page views and load performance from the end user side.

Response times and failure rates to find out if there is an external service that is causing performance issues on your app.

### 3.8 Application Life Cycle with Azure



**Figure 13: Application lifecycle management**

Application lifecycle management is the product lifecycle management of computer programs. It encompasses requirements management, software architecture, computer programming, software testing, software maintenance change management, continuous integration, project management, and release management. In the beginning of the project or product it starts with an idea or some business requirement and after going through multiple steps it takes shape of a usable product. To be more specific, it goes on like this:

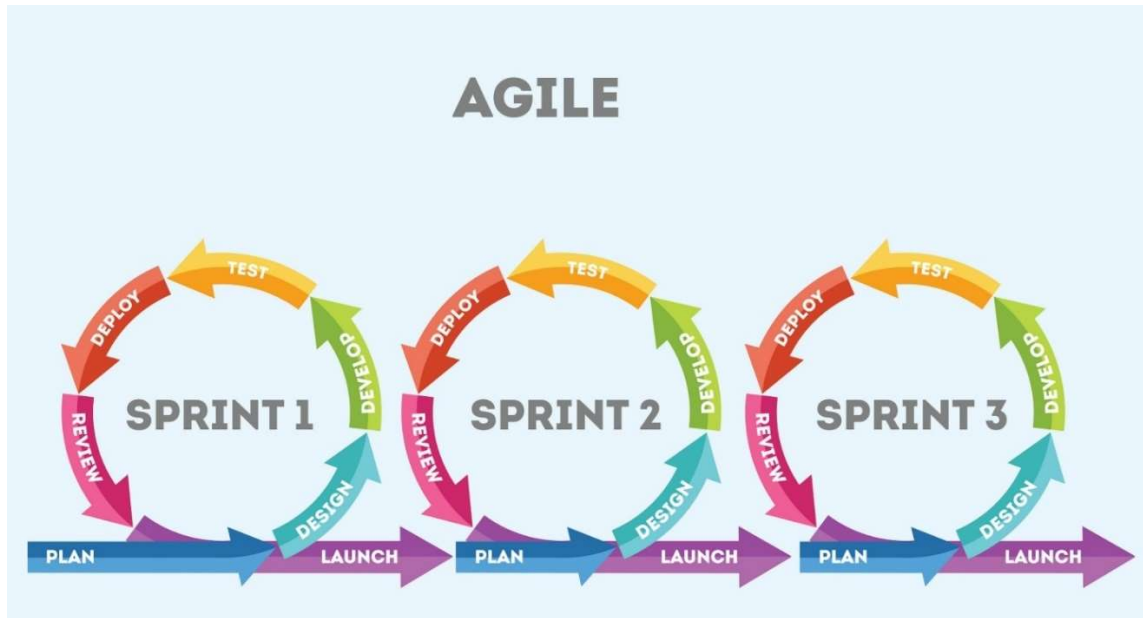
1. Customer comes with requirements.
2. Business analysts/ Product owner analyses the requirement and starts documenting them in the *Azure Boards*. They start creating Epic, Feature and user stories.
3. Engineering team then starts estimating the work and plan those user stories for sprint. In parallel test engineers start for Test planning in *Test Plans* based on the acceptance criteria of the user stories.
4. Once sprint kicked in – developers start picking user stories from the backlogs created at step 2 and starts implementing them.

5. Once developers are done with the development they create a pull request and sends for code review. Upon satisfaction of the code reviewer, developer commits the code in the *Azure Repos*.
6. A build gets triggered in the *Build Pipeline* as soon as the code gets committed in the *Azure Repos* using continuous integration (CI). A CI build gets the latest code from the repos, builds them in the build agents, restores any private packages from the feed setup in the *Azure Artifacts*. It runs unit tests, generates code coverage result and finally generates the build output for deployment. If any one of the steps configured in the build pipeline fails then whole commit gets rejected and developers gets a build failure notification. Developer fixes the build and commits one more time and CI build triggers automatically.
7. Build output needs to be moved to the deployment environment. In a *Release Pipeline* source of the build artifacts is the output of the CI build of previous step 6, destination will be the servers where these artifacts need to be deployed. In this example it is deploying the artifacts to the Azure Cloud App Services and Azure SQL databases for a TESTING environment (step 7.1). A gate or approval mechanism can be set before the deployment happens, this gives control to the approvers to pick and choose which build to be deploy.
8. Now that build is deployed in TESTING environment, test engineers start testing and verifying the build quality. They run a set of manual, automated and load tests as planned during the planning phase (step 3.1) in *Azure Test Plans*. If they observe any defects then they log then in *Azure Boards* backlog and creates a new bug/issue/observation. Developer picks these defects once these are planned and fed back to the sprint, fixes the defects and commits to the *Azure Repos* which eventually triggers CI/CD in Azure Pipelines.
9. If test engineers are satisfied with the build quality and functionality delivered, then this build gets promoted to the upper environment (step 9.1). In this example it is UAT environment for business users to test.
10. Business users now gets a new build in UAT environment with all the features planned for the current sprint. Business user starts verifying the build and functionality in UAT environment.
11. If business users find any defect then they log them into the *Azure Boards* backlogs as bug/observation. Development and build cycle continue once these defects are planned and fed back to the sprint.
12. If business users are satisfied with the functionality and the build quality then the build gets promoted to the PRODUCTION environment (step 12.1). Once this is released to production, end users start using the system. If any issue

occurs in the production then it again gets logged in the *Azure Boards* backlog as an Incident/ Bug/ Issue.

## Chapter 4

### 4. Agile Manifesto



**Figure 14: Agile**

Agile is an iterative product-development methodology. Teams work in brief, incremental "sprints", frequently regrouping to review the work and make changes. The Agile method encourages frequent feedback. Team split out long-term plans into discrete phases for execution. In February 2001, 17 software development practitioners published the Agile Manifesto. It is a brief document built on 4 values and 12 principles for Agile software development.

#### 4.1. The 4 Agile Values

##### 1. Individuals and interactions over processes and tools:

It is the team you work with and the way you work together more valuable than the processes they follow or the tools. You should put a smart, motivated team.

##### 2. Working Software Over Comprehensive Documentation

Under the agile philosophy, getting software in the hands of customers is the highest priority. After all, we should get feedback from real users, to improve on the best way our product.

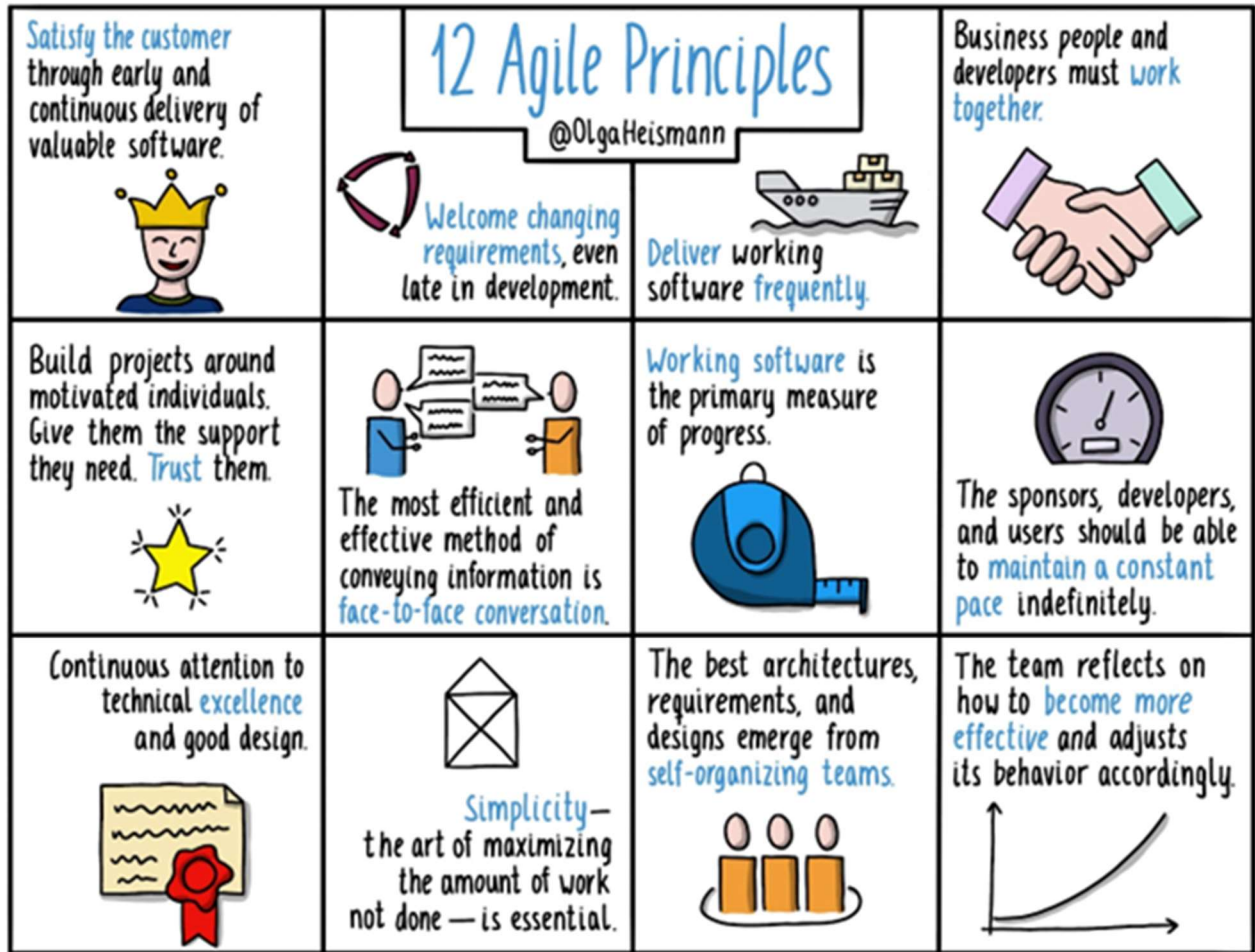
### 3. Customer collaboration over contract negotiation

Under the agile philosophy, customer collaboration begins early in the development process and happens frequently throughout. When you talk to customers often and build feedback into your development process, you reduce risk and eliminate guesswork. No doubt that the contracts will always be important but the real communication with customers is vital.

### 4. Responding to change over following a plan

An important benefit of the agile is that it encourages frequent reviewing and retooling of current plans based on new information that the team is continually gathering and analyzing. The team does not get stuck in an outdated plan simply because it has committed to seeing it through.

#### 4.1.1 The 12 Agile Principles



*Figure 15: Agile Principles*

The Agile Manifesto does not outline any specific processes, procedures, or best practices for agile. The creators created a philosophical mindset for software development.

In theory, the way agile teams work is collaboratively determined by the team members themselves, based on the agile values and principles. Among the common agile approaches are Scrum, Extreme Programming, Kanban, disciplined agile delivery and large-scale derivatives like scaled agile framework (SAFe).

#### 4.2. Scrum

Scrum is the most popular agile framework. It is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems.



**Figure 16: Scrum Methodology**

It introduces 3 different roles:

- **The product owner** is responsible for representing the customer's best interest, This person has the ultimate authority over the final product.
- The self-organizing development **team** with 3-9 members
- **The Scrum master:** The person who leads the team guiding them to comply with the rules and processes of the methodology. The Scrum Master is in charge of keeping Scrum up to date, providing coaching, mentoring and training to the teams in case it needs it.

Ideally, the team applies Scrum processes based on a set of principles and values that must be acknowledged during the software development life cycle. It is the Scrum master's responsibility to uphold Scrum principles and values. Here are the six Scrum principles:

<b>Control over the empirical process</b>	There are three main ideas to empirical process control: transparency, inspection, and adaptation.
<b>Self-organization</b>	Each team members should manage their own tasks, solve problems independently, and are responsible to themselves and each other.



<b>Collaboration</b>	Daily standup meetings are an opportunity to collaborate and problem-solve.
<b>Time boxing</b>	It is a practice where a fixed amount of time is allocated for certain activities or objectives, with the goal of eliminating wasted time and delays.
<b>Value-based prioritization</b>	It involves organizing and prioritizing tasks based on their value and how they need to be completed.
<b>Iterative development</b>	The objectives in product development are consistently reviewed and updated to create the best quality product and delivery process.

***Table 1: Six Scrum principles***

If you are applying Scrum in your workplace, you will be known with the cycle and each of its artifacts. Artifacts are pieces of information and tools that keep the project on-track:

- **Product Backlog** includes the necessary features and functionalities that need to be added to the software. It is continuously updated throughout the project lifecycle as new information comes in, achievements are made and obstacles are overcome.
- **Sprint Backlog** is where teams go through the product backlog to figure out how to achieve the most important objectives. A successful sprint backlog will allow everyone to see which task is being worked on, and by whom, and will reinforce the shared sprint goal.
- **The product increment** is the sum of all the tasks, use cases, user stories, product backlogs and any element that was developed during the sprint and that will be made available to the end user in the form of Software.

According to the Scrum Guide, each of the three artifacts contains a commitment that is used to measure success:

1. Product Backlog → Product Goal
2. Sprint Backlog → Sprint Goal

3. Product Increment → Definition of the JOB.
---

*Table 2: Scrum Guide*

Scrum provides benefits to organizations, product development teams, and individuals. Here are some of the benefits of Scrum:

- **Timely prediction:** You can estimate the average speed of the team and it is possible to estimate when a certain feature in the product backlog will be delivered.
- **Improved team morale:** Being part of a self-organizing team enables people to be proactive, innovative, and focused.
- **Better quality product:** The working method and the need to obtain a functional version after each iteration, helps to obtain a higher quality software.
- **Time to Market reduction:** The client can start using the most important functionalities of the project before the product is completely ready.
- **Flexible to changes:** The methodology is designed to adapt to the changing requirements that complex projects entail.
- **Compliance of expectations:** The product owner verifies that the requirements have been met and transmits feedback to the team.










Each of the Scrum events help on adaptation of some of the aspects of the process, the product, progress or relationships. The events are:

- **Sprint** is the basic unit of work for a Scrum Team, that makes the difference between Scrum and other models for agile.
- **Sprint Planning:** The goal of that is to define what is going to do in the Sprint and how it is going to be done.
- **Daily Scrum:** The objective is to evaluate the progress and trend until the end of the Sprint. It is a brief meeting that takes place daily during the Sprint period.
- **Sprint review:** Here we show what work has been reached with regards to the product backlog for future deliveries.
- **Sprint Retrospective:** It is the phase that the team make a report after the completed goals of the finished sprint. This serves to implement improvements from the point of view of the development process.

In summary, the Scrum methodology is a learning and disciplinary process that enables the Scrum team to identify ways to improve and deliver the best quality product to the end-user.

#### 4.3. Work Item Process

Track various types of work using the default work item types – such as user stories, bugs, features, and epics. Or, customize these types or create your own. Each work item form provides a standard set of system fields and controls. The Deployment, Development, and Related Work controls support tracking when code is released or changed, and relationships between work items. Azure Boards is designed to support software development processes through the default process models selected for a project.

Basic	Agile	Scrum
 Epic	 Bug	 Bug
 Issue	 Epic	 Epic
 Task	 Feature	 Feature
	 Issue	 Impediment
	 Task	 Product Backlog Item
	 Test Case	 Task
	 User Story	 Test Case

**Figure 17: Work Item Process**

**Epic** is a large story that cannot be simply achieved in a single sprint. It takes months to accomplish an epic. They are meant to be split into multiple, smaller stories called **user stories**. User Stories answers the “who”, “what” and “why” of a project in a simple language. Below each epic is a more detailed set of user stories and for those stories to turn into workable components, the Scrum team has to identify and sort Tasks. **Tasks** can range from a few hours to several hours and are assigned to team members who have the skills or expertise to do them. A **feature** represents a shippable software component and reflects a service that fulfills some critical stakeholder needs. **Issues** are used to track events that may block progress or shipping a user story. Bugs are used to track code defects.

The difference between Agile and Scrum items are the switch of “user story” with “product backlog item” and “issue” with “impediment”. The first is that a bug is now managed with requirements instead of with tasks, means that a bug gets a separate card on the board and is visible in the backlog. Also, impediments are not shown on the board or on the backlog.

## Chapter 5

### 5. Integrated Infrastructure

By integrated infrastructure, we mean infrastructure that seamlessly connects with the entire ecosystem of tools you use to deploy and manage applications. At first glance, your IT infrastructure might seem like your house's basement. It supports everything, but you do not pay much attention to it or spend much time or energy trying to make it beautiful. The fact is that infrastructure is not so simple and generic.

#### 5.1 The key to IT Success

The way you design your infrastructure, as well as your ability to integrate it with the ever-changing set of tools that you need to keep workloads running efficiently, plays a critical role in defining overall business success. Some examples of how infrastructure interfaces with tools:

- CI/CD pipeline needs to be able to integrate effectively with the infrastructure that hosts CI servers, testing tools and build tools.
- Security and access-control tools must be able to apply security policies and restrictions to host infrastructure to help keep applications secure.
- Monitoring and performance optimization tools need to work easily with whichever infrastructure hosts your workloads.
- Release automation software must be able to deploy seamlessly to the infrastructure that will host live applications.

There is more, but the idea is that an integrated infrastructure is one that can easily interface with all of the tools you use.

#### 5.2 Achieving Infrastructure Integration

The strategy is to identifying a toolset that offers the best of both worlds: rich features and tight integration, and designing your infrastructure around them. Before you go and migrate all of your workloads to one public cloud or another, identify the various tools you depend on and evaluate how well each of them integrates with the clouds (or other infrastructure solutions) available to you. You may find that you should choose an infrastructure strategy built on multi-cloud or hybrid cloud to achieve better integration with your toolsets. This type of approach will make infrastructure architecture more complex, but it enables better infrastructure integration, which delivers more value in the long run, even if the tools you use probably will change in future. It is important to know that the tools you use today can, and probably will, change in the future. Thus, you want to ensure that your strategy enables tight infrastructure integration over the long term.

Here again, a multi-cloud or hybrid cloud strategy is probably the best approach for maximizing flexibility and integrability, although you will need to assess your specific needs by looking at

your tool first, and then plan an infrastructure strategy that will accommodate them as they evolve and grow.

## Chapter 6

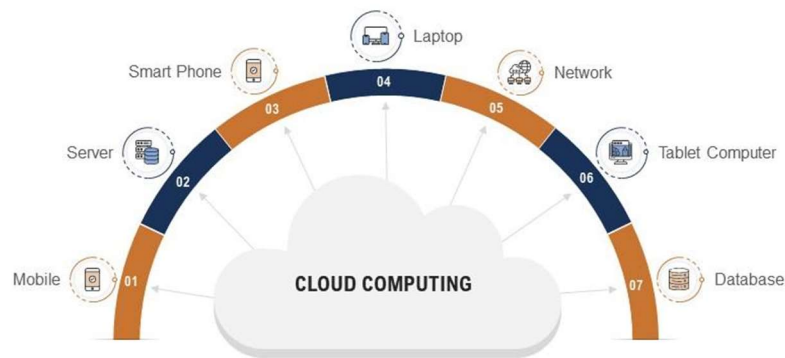
### 6. Cloud adoption



***Figure 18: Cloud & Devops***

DevOps is about the improvement process, while cloud is about technology and services. They are a powerhouse. While each offers greater effectiveness and business impact, together they are able to drive meaningful IT transformation that directly impacts business goals. DevOps won't have much value without the cloud and vice-versa.

- Cloud computing



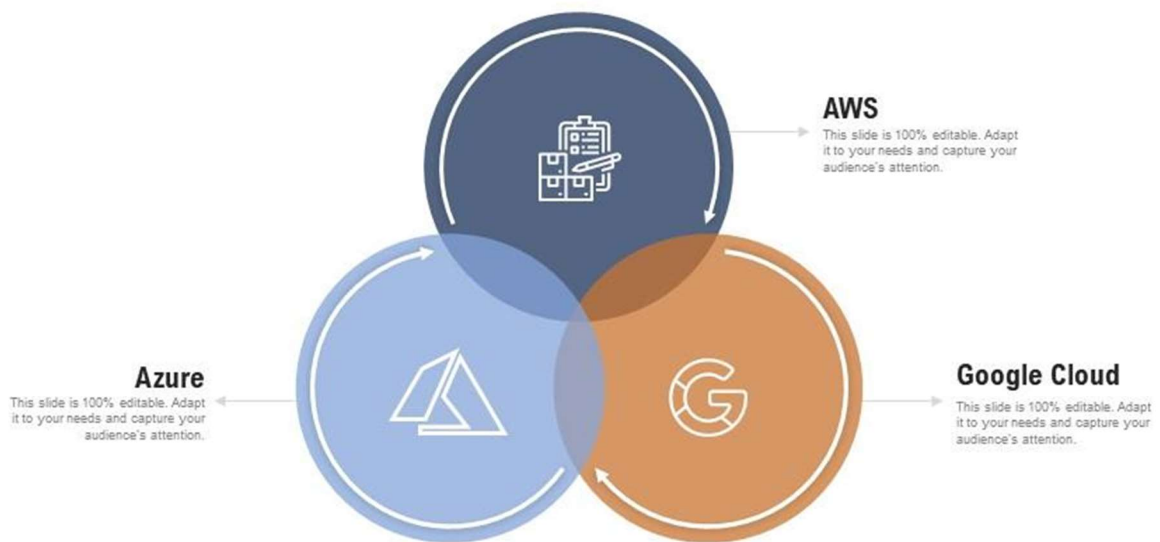
**Figure 19: Cloud Computing**

Cloud computing advances IT transformation and it can enable companies to double down on their work to streamline and embed DevOps processes for greater efficiencies that are truly transformative. It can play a role here, as it can help codify and automate new processes. Reasons:

- **Cost effective, Speed, Global Scale, Productivity** which eliminate a lot of “old” tasks
- **Performance** where reduce latency and high efficient computing hardware
- **Reliability:** Backup & Recovery and easier continuity – also less expensive
- **High security** with automated, repeatable processes and most importantly, develop security controls from the very beginning.

Type of Cloud computing: Public (Microsoft Azure) / Private (Physically located on the company’s datacenter) / Hybrid (Mix of them)

Types of Cloud Services: **IaaS** (Servers, VMs, storage, operating systems), **PaaS**(Refers to Cloud Computing: easier for developers to create web or mobile apps), **Serverless**(building app functionality) and **SaaS** (method for delivering software app over the Internet)



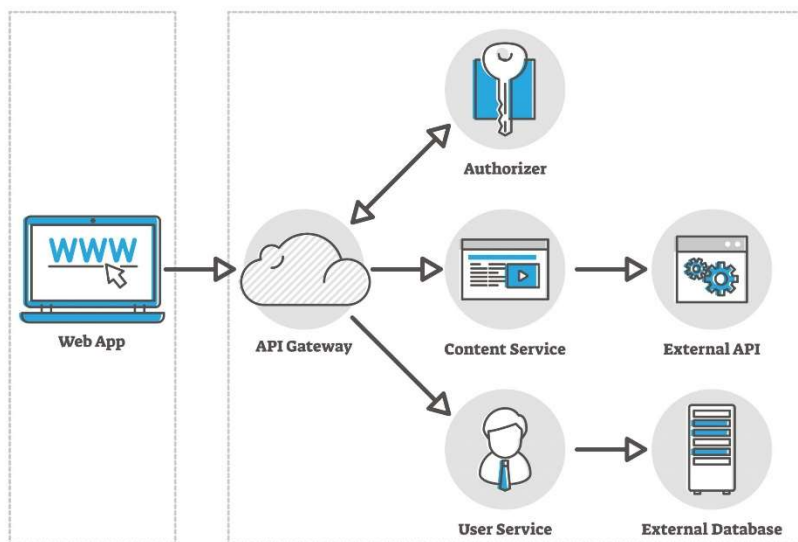
**Figure 20: Biggest Cloud Computing Provider**

- **Serverless Computing**

Serverless computing enables developers to build applications faster by eliminating the need for them to manage infrastructure. So, the cloud provider automatically provisions, scales and manages the infrastructure required to run the code. This approach will give the advantage to developers to increase their productivity and bring products faster for the market, and it allows to optimize much better resources.

- **Functions** (Execute code: written in the language of your choice)
- **Kubernetes** (Developers bring their own containers to fully managed, Kubernetes-orchestrated clusters that can automatically scale up and down with sudden changes)
- **Workflows** (Developers can integrate different services without coding those interactions, having to maintain glue code or learning new APIs or specifications)
- **App environments** (The back and front end are hosted on fully managed services that handle scaling, security and compliance requirements.)
- **API gateway** (A fully managed and centralized API gateway enables developers to publish, manage, secure and analyse them at global scale)

# SERVERLESS



**Figure 21: Serverless Computing Architecture**

To conclude, cloud computing is gaining popularity today. It contributes to the improvement of various types of businesses. Even though there are some issues people may face using it, the advantages outweigh the drawbacks. Thus, more and more companies are interested in how to implement such solutions and save their money. Serverless architecture is also on-demand nowadays. It lets companies focus on their services and products. They do not worry about the infrastructure and save money as they pay only for the things they use.

The main difference is the second option is cheaper and lets you focus more on the services rather than on the infrastructure.

## 6.1 Kubernetes

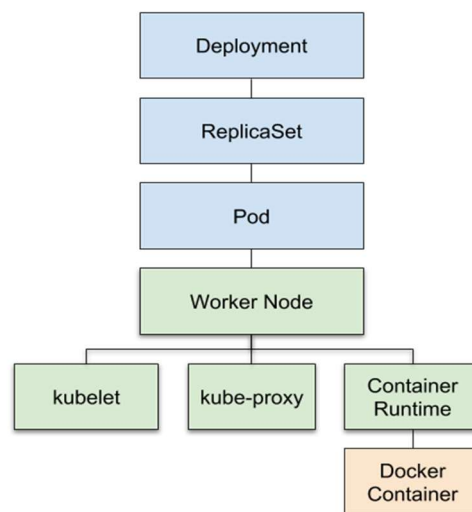


**Figure 22: Kubernetes**



Kubernetes is open-source orchestration software which it is great for deploying and managing reliable applications, scalable applications without effecting the end-users. BUILD – DELIVER – SCALE faster with K8s. Through K8s, you have workloads portable. You define complex containerized applications and deploy them across multiple clusters. Build more extensions and plugins that add capabilities such as security, monitoring and management to Kubernetes. Every K8s version support APIs that make it easier to use them.

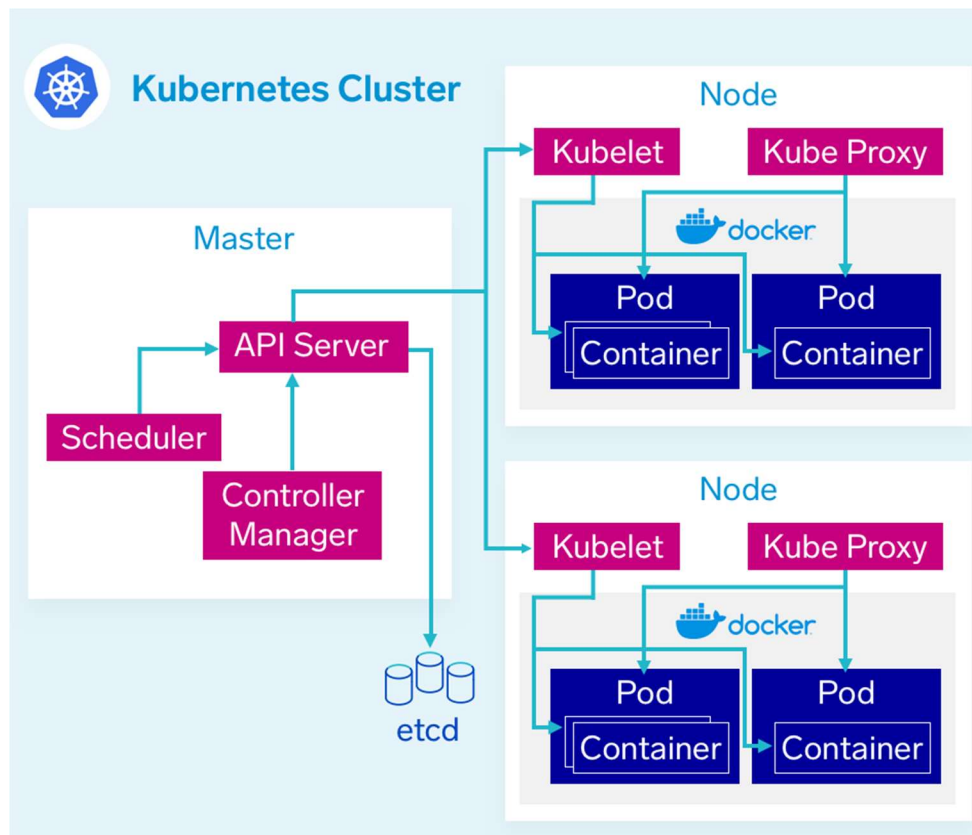
- Deliver code faster with CI/CD (Azure Pipelines)
- Manage resources effectively with infrastructure as code
- Accelerate the feedback loop with constant monitoring
- Balance speed and security with DevOps (DevSecOps)



***Figure 23: Kubernetes 6 Levels***

Deployments create and manage Replica Sets, which create and manage Pods, which run on Nodes, which have a container runtime, which run the app code you put in your Docker image.

- **Kubernetes Architecture**



**Figure 24: Kubernetes Cluster**

**Table 3: Kubernetes Components**

<i>Kubernetes Components</i>	<i>Functions</i>
<i>Master</i>	It manages, plans, schedules and monitors nodes.
<i>Node</i>	It hosts the application as the container. It could be a physical or virtual server
<i>Pods</i>	It is made up of one or more containers
<i>API Server</i>	It is responsible for orchestrating all operations within the cluster
<i>Scheduler</i>	It is responsible for scheduling applications or containers on Nodes
<i>Controller Manager</i>	Controllers available that take care of different areas

<i>ETCD</i>	It is a database that stores information about the cluster, in a key-value format
<i>Kubelet</i>	It ensures that the necessary rules are in place on the worker nodes to allow the containers running on them to reach each other
<i>Kube Proxy</i>	It ensures that the necessary rules are in place on the worker nodes to allow the containers running on them to reach each other

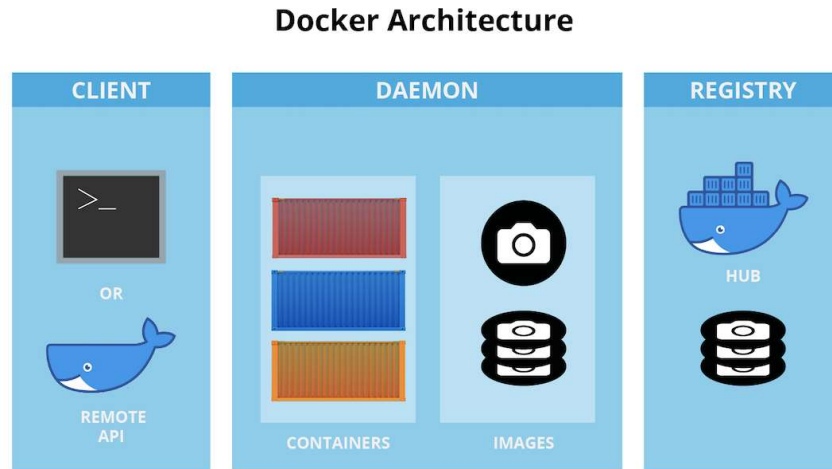
## 6.2 Kubernetes & Docker

As we talk before for Kubernetes, which is an open-source system for automating the deployment, scaling and management of containerized applications, we should know that Docker is an open platform for developing, shipping and running applications. It is important to start with the foundational technology that ties them together: **containers**.

**Container** solve a critical issue of portability allowing you to separate code from the underlying infrastructure it is running on. Developers could package up their application, including all of the bins and libraries it needs to run correctly, into a small container image. In production that container can be run on any computer that has a containerization platform. Let's make a compare with Virtual Machine (VM). Both are based on virtualization technologies, but while a container virtualizes an OS, a VM leverages a hypervisor (*a lightweight software layer between the VM and a computer's hardware*) to virtualize physical hardware. The absence of a guest host significantly reduces the size of a container, making it fast, and portable.

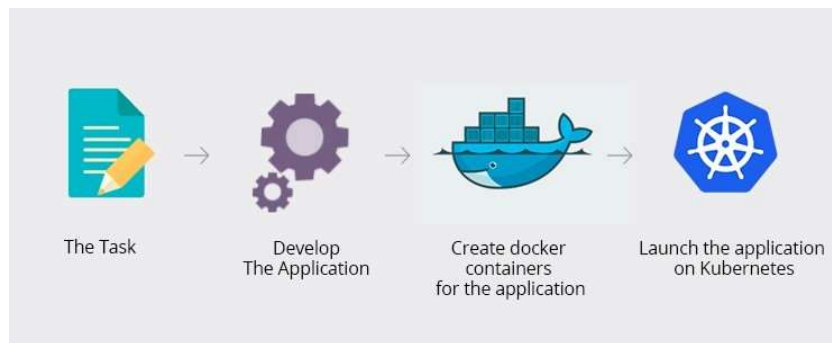
The portability of containers eliminates many of the conflicts that come from differences in tools and software between functional teams, which makes them particularly well-suited for DevOps workflows, easing the way for developers and IT operations to work together across environments.

To rephrase it, **Docker** is a toolkit that makes it easier, safer and faster for developers to build, deploy and manage containers. Only one process can run in each container, so an application is able to run continuously while one part of it is undergoing and update or being repaired. Docker image act as a set of instructions to build a Docker container. While, Docker containers can run across any data center or cloud environment.



**Figure 25: Docker Architecture**

Although **Kubernetes** and **Docker** are distinct technologies, they are highly complementary and make a powerful combination. Docker provides the containerization piece, enabling developers to easily package applications into small, isolated containers via the command line. When demand surges, Kubernetes provides orchestration of Docker containers, scheduling and automatically deploying them across IT environments to ensure high availability. Later versions of Docker have built-in integration with Kubernetes, which enables development teams to more effectively automate and manage all the containerized applications that Docker helped them build.



**Figure 26: Combining Docker and Kubernetes**

## Chapter 7

### 7. Git

Git is a DevOps tool used for source code management. It is an open-source version control system used to handle small to very large projects efficiently. With Git, we can track changes in the source code. It allows multiple developers to work together and supports non-linear development through its thousands of parallel branches. A branching strategy helps define how

the delivery team functions and how each feature, improvement, or bug fix is handled. It also reduces the complexity of the delivery pipeline by allowing developers to focus on developments and deployments only on the related branches without affecting the entire product.



*Figure 27: Git*

The biggest advantage of a Git branch is that it is 'lightweight', meaning that data consists of a series of snapshots so with every commit you make, Git takes a picture of what your files look like at that moment and stores a reference to that snapshot.

- **Git key concepts**

- Version Control System: Git helps maintain various versions of the codebase at different stages of the development lifecycle. It is called a source code manager.
- Commit: When a developer makes code changes, the changes are saved in the local repository. Every commit saves a copy of the changed/added files within Git.
- Push: This sends the recent commits from developer's local repository to a remote server like GitHub, GitLab, or Bit Bucket.
- Pull: This downloads any changes made from the remote Git repository and merges them into the developer's local repository.
- SHA (Secure Hash Algorithm): This is a unique ID given to each commit.
- Branch: When you diverge from the mainline of software development and continue to work/code without messing with the main/master development line.

There are two main branching strategies: Git Flow and Trunk based development and other 3 less important: *GitHub*, *GitLab*, *One Flow*.

### **7.1 Git Flow**

The main idea of Git Flow is to isolate your work into different types of branches. There are five branch types in total:

- a. Master is where the most recently released code that is in the user's hands lives.

- b. Develop is the branch where all development going into the next release lives.
  - c. Feature is the branch that helps developers to work on different features in isolation and makes possible the ability to merge their features back into the develop branch which is the active development version of the applications.
  - d. Release is a branch that is based on the develop branch. Once develop has acquired enough features for a release, you fork a release branch off of develop. You can only fix bug or documentation generation on this branch.
  - e. Hot Fix branches are necessary to act immediately upon an undesired status of master. It is like release and feature branches except they are based on master instead of develop.
- **The benefits of Git Flow:**
    1. You can organize your work easily because of the various types of branches.
    2. Release branches allows you to support multiple versions of production code, continuously.
    3. The systematic development process allows for more effective testing.
  - **The challenges of Git Flow:**
    1. It is now able to support Continuous Delivery or Integration.
    2. On the cases where the complexity of the product is higher, the Git Flow model could overcomplicate and slow the development process and release cycle.

## 7.2 Trunk based development

In the trunk-based development model, all developers work on a single branch with open access to it. It is much simpler, just code and run it. We only have the master branch that we need to think about. At the most cases, developers that work in such style should be experienced so that you know they won not lower source code quality. The only way to review code in such an approach is to do full source code review. It is also recommended to use this model on small teams, trying to get their app to the users as fast as possible.

- **The benefits of Trunk-based development**
  1. When you are working on your minimum viable product by using this method, you get maximum development speed with minimum formality.

2. On cases where the customers want something different and to use this method to pivot into a new direction. There will be a change to your product as fast as possible.
3. There is no place for uncertainty if the majority of your staff are senior Devs. Let them carry out their duties with the independence they require to produce a flawless result.

- **The challenges of Trunk-based development**

1. The Trunk method, it is not good when you run an open source project. After all, anyone can contribute, including online trolls.
2. It is probably impossible for large teams or new engineers to maintain strict control over what is happening with a well-known product worth millions of dollars.

### **7.3 GitHub, GitLab, One Flow**

- GitHub Flow is a way simpler workflow. We do not have “releases” because we deploy to production every day or several times a day. It is not recommended when multiple versions in production are needed.
- GitLab Flow offers a transparent and effective way between the code and issue tracker. Any change to the code should start with an issue that describe the goal. You should work with release branches only if you need to release software to the outside world. So, each branch contains a minor version. It is more complex than the GitHub Flow, on the same level as Git Flow.
- The One Flow’s purpose is to have one eternal branch in your repository. So, every new production release is based on the previous release. Here we cannot find the DEVELOP branch. It is not recommended for projects with Continuous Delivery.

### **7.4 Top Git Hosting Services for 2022**



***Figure 28: Top Git Hosting 2022***

A web hosting service is a type of Internet hosting service that hosts websites for clients. It offers the facilities required for them to create and maintain a site and makes it accessible on the World Wide Web. “Hosting” means that as users we have access to what is offered to us. A definite plus is the lack of the need to create your own infrastructure, taking care of safety and operating time. On the downside, of course, there is the cost and dependence on a given service provider.

- Here is the list of the best host Git Repository online

***Table 4: The best host Git Repository***

<div> <div></div> <div>GitHub</div> <div></div> </div>	<p>It is the most used web-based Git repository hosting service because it provides many useful features and is in many cases free. Code review and other collaboration features work great here. GitHub supports not.</p>
<div> <div></div> <div>GitLab</div> <div></div> </div>	<p>In terms of basic functionality, it is very similar to GitHub. Git Lab is an open-source end-to-end software development platform with built-in version control, issue tracking, code review, CI/CD, and more. Self-host GitLab on your own servers, in a container, or on a cloud provider.</p>
<div> <div></div> <div>Bitbucket</div> <div></div> </div>	<p>One of the giants in the industry. The biggest advantage of Bitbucket Server is certainly that it comes from Atlassian – the company that develops highly popular products like Jira and Confluence. Bitbucket gives teams one place to plan projects, collaborate on code, test and</p>

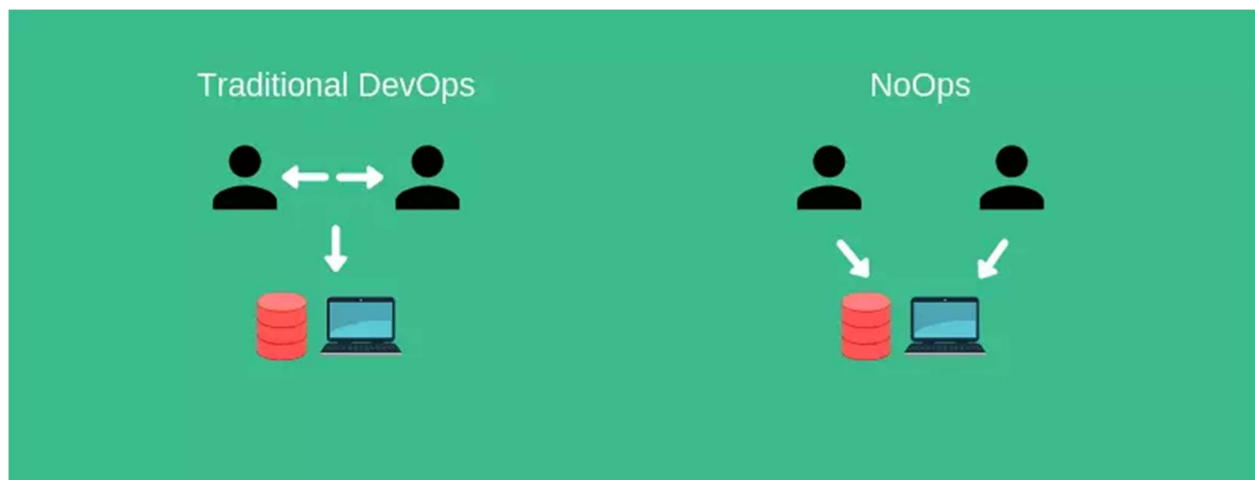


	<p>deploy. Free for 5 users and you can get unlimited number of free private repositories.</p>
<hr/> <p>Azure DevOps °Repos°</p> <hr/>	<p>It offers code hosting, CI/CD, and planning tool. It is a platform for any development stack and is addressed to business users rather than private ones.</p>
<hr/> <p>AWS Code Commit</p> <hr/>	<p>Amazon's AWS platform includes hosting for Git repositories. It also comes with features that facilitate collaboration, such as code review and access control. It only offers private repos and it is free for up to 5 active Users. AWS Code Commit fasten the development lifecycle. You can easily transfer incremental changes instead of the entire application.</p>

## Chapter 8

### 8. To DevOps OR NOT to DevOps?

Nowadays, DevOps is deeply integrated into the DNA of all cloud-first organizations and is today more of a norm than a rarity. Cloud applications demand agility, and DevOps delivers it. Cloud service providers are also investing heavily in their automation ecosystem. You can easily provision your application components using automation templates or just a few API calls, meaning minimal to no human intervention. This leads us to NoOps, which means that an IT environment can become so automated and abstracted from the underlying infrastructure that there is no need for a dedicated team to manage software in-house. No Ops aims to improve productivity and deliver results much faster than DevOps.



*Figure 29: DevOps & NoOps*

- The benefits of NoOps:
  - Use the full power of the cloud. Microservices and API-based application architectures fit the bill perfectly, as they offer precise modularity along with automation. The current increase in database/container/function -as-a-service options in the cloud favor NoOps.
  - More automation, less headcount. The idea to focus to services that are deployable by design without manual intervention. They essentially seek to eliminate the manpower required to support the ecosystem for your code.

- Shift from operations to business results. NoOps ideally eliminates any dependency on the operations team, which further reduces time to market.
- **The challenges of NoOps:**
  - The need of Ops is still on, because expecting developers to take care of this would nullify the benefits of NoOps and take away their focus from delivering business outcomes. It is also not a practical approach, considering that developers do not necessarily have the required skill sets to address operational issues.
  - The importance of Security: Automated deployments aligned with security best practices will not completely eliminate the need for you to take care of security. The operations team works to enforce controls that protect applications from threats and vulnerabilities, together with the security team. Reducing, or eliminating, the operations team could result in you needing to increase your investment in a security team to ensure the security and compliance of your environments.
  - Considering that not all environments can transition to NoOps. Hybrid deployments and legacy infrastructures would pose a bottleneck. Automation is still possible, but human intervention cannot be entirely eliminated in these cases.

In conclusion, DevOps emphasizes continual improvement, and NoOps is the next step in DevOps' progression. DevOps engineers' roles also alter as a result of the chance to master the new techniques and procedures needed for NoOps. Service providers provide development teams with the necessary cloud infrastructure, patching, backups, and resources so they can operate independently rather than together. NoOps focuses more on a change in culture and procedure, and this change won't take place overnight.

## Chapter 9

### 9. Introduction

In this chapter, proof-of-concept implementation of DevOps environment for a small-scale web application project will be present. The goal is Deploying .Net Microservices with Kubernetes, move cloud Azure Kubernetes Services, automating with Azure DevOps. This is a project named Shopping, suggested by the Orbyta Tech through a course on Udemy.



***Figure 30: Orbyta logo***

We are going to explain how to deploy .Net Microservices into Kubernetes and moving deployments to the cloud Azure Kubernetes Services with using Azure Container Registry. In the end we also gone automate with CI/CD pipelines of Azure DevOps and GitHub. Additionally, we'll use the Docker environment to containerize our microservices, upload images to the Docker Hub, and establish our microservices on Kubernetes. With the same setup, we will move our Azure Kubernetes Service deployment to the cloud using Azure Container Registry.

## **9.1 Building process**

The process requires that there is a company directory of projects and that there is a standardization of the process itself. A brief architecture document is requested for opening the project that allows the DevOps team to define the guidelines for a possible development environment. Once the development environment has been defined, the DevOps team takes care of updating the architecture documents with details of the DevOps/Pipeline/Azure environment and costs.

### **9.1.1 Qualification**

At the address <https://orbytait.sharepoint.com/Lists/IT%20Projects.xxx> there is the list of projects which is update and maintained by tech leads.

The screenshot shows the Orbyta Developers Hub interface. At the top, there's a navigation bar with 'ORBYTA', 'Company', 'HR', and 'Support'. Below it, a dark blue header contains 'Developers Hub' and a 'Not following' status. A toolbar includes buttons for '+ New', 'Edit in grid view', 'Share', 'Export', 'Automate', 'Integrate', and more. The main section is titled 'IT Projects' and displays a table with the following columns: ID, Cliente, Titolo, Title, Descrizione, Responsabile, and Team.

ID	Cliente	Titolo	Title	Descrizione	Responsabile	Team
1		4shop	4shop			
2		QBO.next	QBO.next	Si tratta di un tool di pianificazione di test in particolare per ambito automotive. Il tool prevede: - configurazione		
3		e-STEP	e-STEP	La dashboard da realizzare sulla piattaforma e-Shop In Store dovrà essere progettata secondo una struttura modulare		

**Figure 31: Orbyta List Projects**

At the opening project, we must be qualified by entering:

The screenshot shows the 'New item' form for creating a project. It has a sidebar on the left with fields for 'Cliente', 'Title', 'Descrizione', 'Responsabile', 'Team', and 'Topics'. The main area on the right contains several sections: 'Teams' with a URL field, 'DevOps\_req' with a checked 'Yes' checkbox for 'Richiesto Devops?', 'DevOps' with a URL field, 'Azure\_Dev\_req' with an unchecked 'Yes' checkbox for 'Richiesto ambiente sviluppo azure', 'ArchDoc' with a URL field, 'Stato' with a dropdown set to 'APERTO', and 'Allegati' with an 'Add attachments' button. At the bottom right are 'Save' and 'Cancel' buttons.

**Figure 32: Project creation**

**Table 5: Fill up the project**

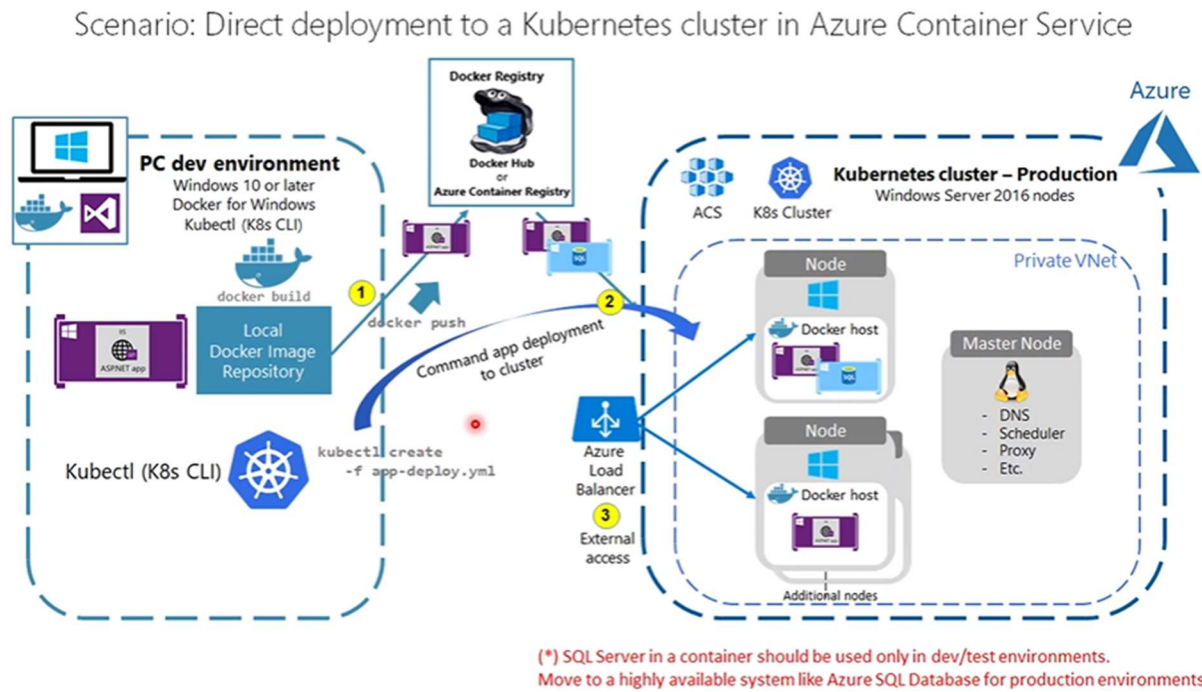
<ul style="list-style-type: none"><li>• Client name</li></ul>
<ul style="list-style-type: none"><li>• Title project</li></ul>
<ul style="list-style-type: none"><li>• Description (synthetic)</li></ul>
<ul style="list-style-type: none"><li>• Responsible</li></ul>
<ul style="list-style-type: none"><li>• Teams: Yes/No (If it will be created by team infra)</li></ul>
<ul style="list-style-type: none"><li>• Teams: Name/URL</li></ul>
<ul style="list-style-type: none"><li>• MS DevOps: Yes/No (If you need a DevOps, it is created by the team Infra and access to resources is given. The architecture document is updated.)</li></ul>
<ul style="list-style-type: none"><li>• DevOps: URL (if MS DevOps was insert by team infra, if not another URL)</li></ul>
<ul style="list-style-type: none"><li>• Azure DevOps: Yes/No</li></ul>
<ul style="list-style-type: none"><li>• Architecture document: The architecture document is checked (or drawn up if it does not exist) and the project is better defined.</li></ul>
<ul style="list-style-type: none"><li>• Status: Open/Close/In qualifying</li></ul>
<ul style="list-style-type: none"><li>• Attach files that helps on project.</li></ul>

To conclude, we are going to fill the last procedures:

- Test environment: If you need an Orbyta test environment, it will be created on Azure by the DevOps team. It is given access to the team's resources. The architecture document is updated, again. The guide line:
  - Create RG ad Hoc, the developers job.
  - Create the resource following the architecture document
- Cost Monitor: A spending limit is set and the indicative spending on SharePoint Online, is assigned.
- Set Up & define Repository: The Infra team creates the DevOps environment by defining the process type (Agile/Scrum), qualifies the team. Be careful who has a Visual Studio subscription to activate it. With the project team, it defines the repositories and the type of pipeline (Build & Deploy).

## 9.2 Shopping Project Overview

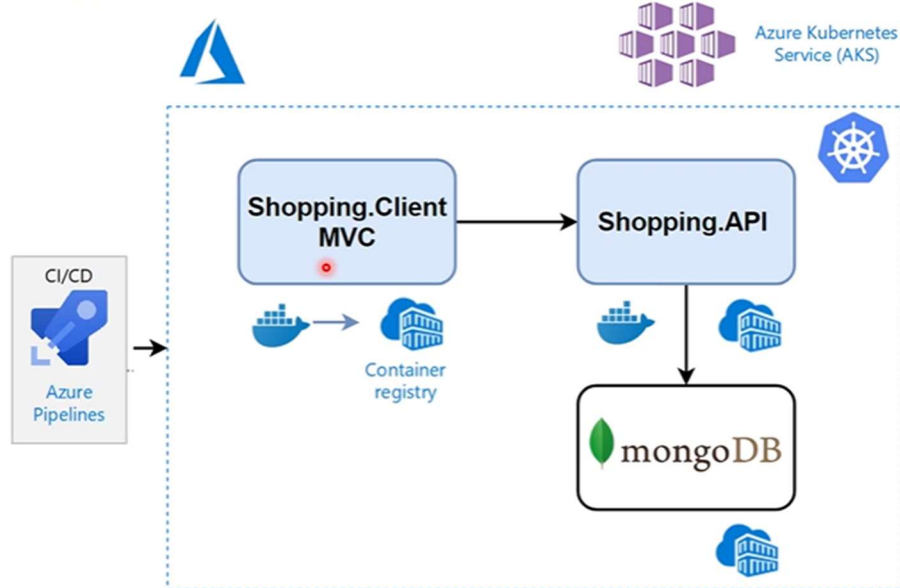
Firstly, we must build, test and deploy an application. It will have a typical web application structure.



**Figure 33: Web Application Structure**

In the project, we will have 3 microservices which we are going to develop and deploy together.

## Big Picture



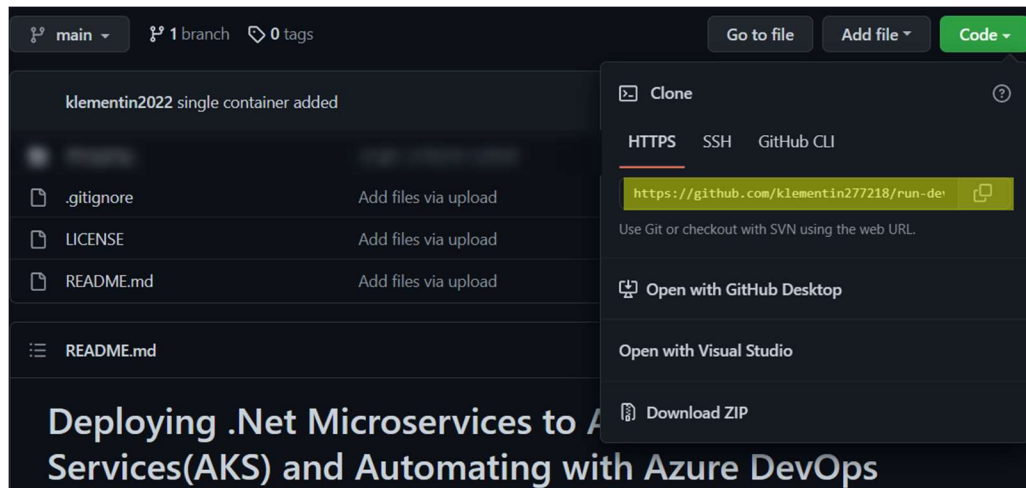
**Figure 34: Shopping project structure**

### 9.2.1 Microservices 1 – Shopping MVC Client Application

We are going to develop Shopping.Client MVC Application for consuming API Resource. The project includes own data inside it, as a standalone Web application. Then we will add container support, push docker images to Docker Hub and see the deployment options like “Azure Web App for Container” resources for one web application.

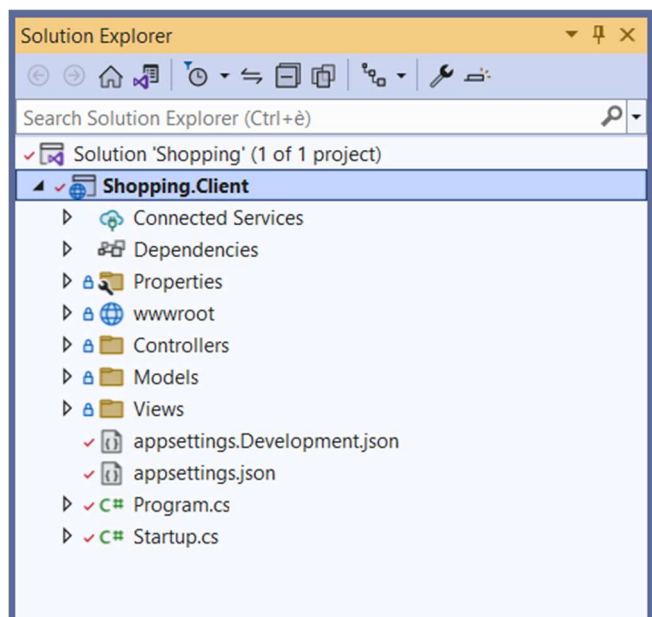
- Firstly, we will clone our repository and after that we will start the project. From the github page, where we have our repository **run-devops**, we will clone it at Visual Studio 2022 app. After that we will create our project named “Shopping”.





**Figure 35: GitHub repository**

Under the Shopping project, we are going to create a new project design. The template will be ASP.NET Core Web Application (Model-View-Controller) and the project name is Shopping.Client.



**Figure 36: Solution Explore**

We are going to develop shopping client MVC data model and context objects. Now, we add model class inside Model folder. Create a model class name Product.cs and its code is as below.

```

namespace Shopping.Client.Models
{
    public class Product
    {
        public string Id { get; set; }
        public string Name { get; set; }
        public string Category { get; set; }
        public string Description { get; set; }
        public string ImageFile { get; set; }
        public decimal Price { get; set; }
    }
}

```

After that we will do some changes in code at Index.cshtml and HomeController.cs, to list the products into the index page. Index page operation are handled from the home controller class.

- We have arrived at the moment to create Docker container for our Shopping.Client. We will have Dockerfile, which his purpose is when we ask the docker to extract the image of our project, it will search for a file which name is the file Dockerfile in the project. So this will make our application work accordingly to the settings in our file into the docker.

```

1 #See https://aka.ms/containerfastmode to understand how Visual Studio uses this Dockerfile to build your images for faster debugging.
2
3 FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
4 WORKDIR /app
5 EXPOSE 80
6
7 FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
8 WORKDIR /src
9 COPY ["Shopping.Client/Shopping.Client.csproj", "Shopping.Client/"]
10 RUN dotnet restore "Shopping.Client/Shopping.Client.csproj"
11 COPY . .
12 WORKDIR "/src/Shopping.Client"
13 RUN dotnet build "Shopping.Client.csproj" -c Release -o /app/build
14
15 FROM build AS publish
16 RUN dotnet publish "Shopping.Client.csproj" -c Release -o /app/publish
17
18 FROM base AS final
19 WORKDIR /app
20 COPY --from=publish /app/publish .
21 ENTRYPOINT ["dotnet", "Shopping.Client.dll"]

```

**Figure 37: Dockerfile code**


Basically, the file consists two main parts: the building application and the publishing and running application. To run docker, it is important to have install it on computer. After build the docker, we will have Shopping.Client container and images. We can check that through docker ps and docker images. The last thing to do, it is to create



the repository at docker hub and tag docker image with repository name. Once we have it, we push our image to docker hub. The code is as below:


```
docker login
```



```
docker tag [image id] 7991575001/shoppingapp
```

```
PS C:\Users\Klementin Hasani\source\repos\run-devops\Shopping\Shopping.Client> docker push 7991575001/shoppingapp
Using default tag: latest
The push refers to repository [docker.io/7991575001/shoppingapp]
702ba3329e69: Pushed
5f70bf18a086: Pushed
cb28f2edcc76: Pushed
303f78916dd4: Pushed
c6a55939716e: Pushed
c152578bc6ac: Pushed
531af3c69d62: Pushed
92a4e8a3140f: Pushed
latest: digest: sha256:ddb1ccad591ea152d991c126fbf3e4c3e351bde23fdcea0f77d683c938ccd02b size: 1995
```

 **7991575001 / dockerapp**

**Description**  
dockerapp   
 Last pushed: 5 minutes ago

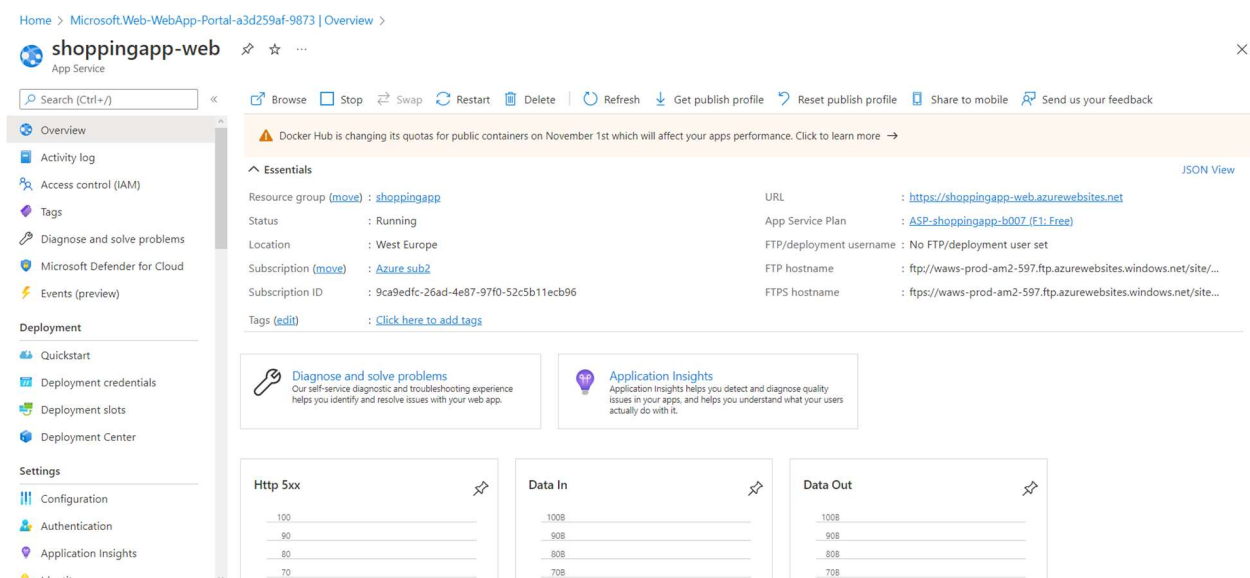
**Tags and Scans**  **VULNERABILITY SCANNING - DISABLED** [Enable](#)  
This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
 latest		5 minutes ago	5 minutes ago

[See all](#) [Go to Advanced Image Management](#)

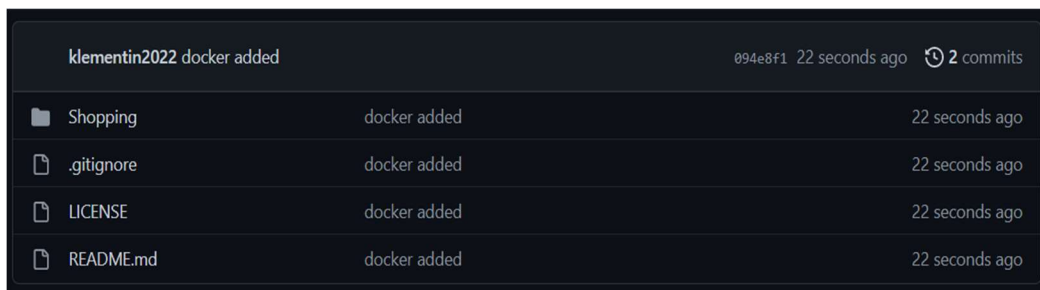
**Figure 38: DockerHub push**

- Finally, we are going to automate build using GitHub and Docker Hub. At Build option of Docker Hub, we will connect it with Git Hub and accept it from Git. Save all the procedure and go to Visual Studio to commit all and push, by adding a comment “docker added”. Meanwhile, on Azure we will create a Web App.



**Figure 39: shoppingapp-web service**

- Once the image is built in Docker Hub, Azure will capture the docker image and deploy the application. There is the URL web application. This is the one container deployment process.



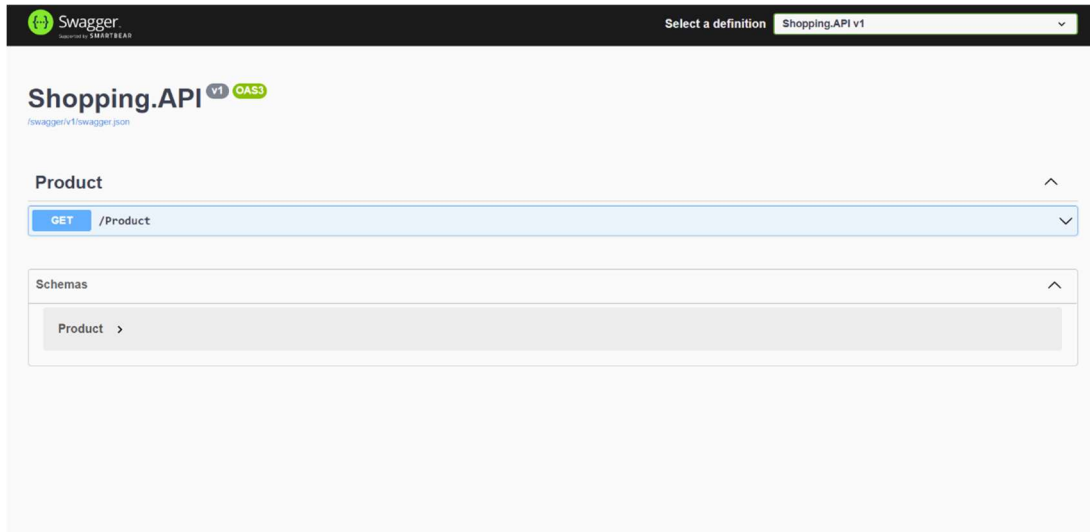
**Figure 40: Update github**

## 9.2.2 Microservices 2 – Shopping API Application

After that we are going to develop Shopping.API Microservice with MongoDB and compose all docker containers. This API project will have Products data, which Shopping Client will consume. We will containerize API application with creating Dockerfile and push images to Azure Container Registry.

- As before with ShoppingClient, we will create the Shopping.API project, instead this time ASP.NET Core Web API template is the chosen one. We add the Product.cs inside

Models folder, ProductController.cs inside Controllers folder and the ProductContext.cs inside Data folder. Now we have everything, that is needed to make project works. Through Swagger, we will have the output of project that will be like below. To remember that Swagger allows you to describe the structure of your APIs.



**Figure 41: Shopping.API Swagger**

Shopping.Client Home Privacy				
Products				
Name	Category	Description	ImageFile	Price
Tesla Model X	Hybrid	The Tesla Model X is a battery electric mid-size luxury crossover produced by Tesla, Inc. since 2015.	product-1.png	40.00
Audi Q8	Diesel	The Audi Q8 is a mid-size luxury crossover SUV coupé made by Audi that was launched in 2018.	product-2.png	84.00
Mercedes-Benz GLA	Diesel	The Mercedes-Benz GLA is a subcompact luxury crossover SUV manufactured and marketed by Mercedes-Benz over two generations.	product-3.png	75.00
Fiat 500	Petrol	The Fiat 500 is a rear-engined, four-seat, small city car that was manufactured and marketed by Fiat Automobiles from 1957 until 1975 over a single generation in two-door saloon and two-door station wagon bodystyles.	product-4.png	22.00
Chery Tiggo 5	Diesel/Hybrid	The Chery Tiggo 5 is a compact crossover produced by Chery under the Tiggo product series.	product-5.png	38.00
Jeep Compass	Hybrid	The Jeep Compass is a compact crossover SUV introduced for the 2007 model year, and is now in its second generation.	product-6.png	44.00

**Figure 42: Shopping.Client Website**

Through the code below, we make possible that Client application consume from Shopping.API. We are using HTTP client and giving product suffix.

```
public async Task<ActionResult> Index()
{
    var response = await _httpClient.GetAsync("/product");
    var content = await response.Content.ReadAsStringAsync();
    var productList = JsonConvert.DeserializeObject<IEnumerable<Product>>(content);

    return View(productList);
}
```

### 9.2.3 Microservices 3 – MongoDB Database

Now it is time to create a real database which is No-SQL MongoDB. As a definition, MongoDB is an open-source database that uses a document-oriented data model and a non-structured query language. This best-in-class automation and established practices offer to deploy fully managed MongoDB across AWS, Google Cloud, and Azure.

Back on project, we will pull MongoDB docker image from docker hub and create connection with our API project. At Shopping.API, we are going to install MongoDB.Driver to provide connection with Mongo.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="MongoDB.Driver" Version="2.17.1" />
    <PackageReference Include="Swashbuckle.AspNetCore" Version="6.2.3" />
    <PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="5.0.1" NoWarn="NU1605" />
    <PackageReference Include="Microsoft.AspNetCore.Authentication.OpenIdConnect" Version="5.0.1" NoWarn="NU1605" />
    <PackageReference Include="Microsoft.VisualStudio.Azure.Containers.Tools.Targets" Version="1.10.9" />
  </ItemGroup>

</Project>
```

**Figure 43: MongoDB.Driver**

The package reference MongoDB.Driver is included in our project. After that, we add Database Settings which include Connection String and Database Name/Collection Name, at appsettings.json file.

```
{
  "DatabaseSettings": {
    "ConnectionString": "mongodb://shoppingdb:27017",
    "DatabaseName": "ProductDb",
    "CollectionName": "Products"
  }
}
```

It remains to make the connection available for Mongo docker container from Shopping.API. So once the context is created, this will create Mongo connection and see the database. After that the whole application uses existing configurations.

```
0 references
public ProductContext(IConfiguration configuration)
{
    var client = new MongoClient(configuration["DatabaseSettings:ConnectionString"]);
    var database = client.GetDatabase(configuration["DatabaseSettings:DatabaseName"]);

    Products = database.GetCollection<Product>(configuration["DatabaseSettings:CollectionName"]);
    SeedData(Products);
}

3 references
public IMongoCollection<Product> Products { get; }

1 reference
private static void SeedData(IMongoCollection<Product> productCollection)
{
    bool existProduct = productCollection.Find(p => true).Any();
    if (!existProduct)
    {
        productCollection.InsertManyAsync(GetPreconfiguredProducts());
    }
}
```

**Figure 44: MongoDB added**

We should register these contexts object, so we add: "services.AddScoped<ProductContext>();" at ConfigureServices method of Startup.cs. This is mandatory step for creating objects from ASP.NET Core.

- The last step to update our code, it will be at ProductController.cs, where we are going to implement Index Get method with calling /products api from shopping.api project.

```
public async Task<IActionResult> Index()
{
    var response = await _httpClient.GetAsync("/ product");
    var content = await response.Content.ReadAsStringAsync();
    var productList = JsonConvert.DeserializeObject<IEnumerable<Product>>(content);

    return View(productList);
}
```

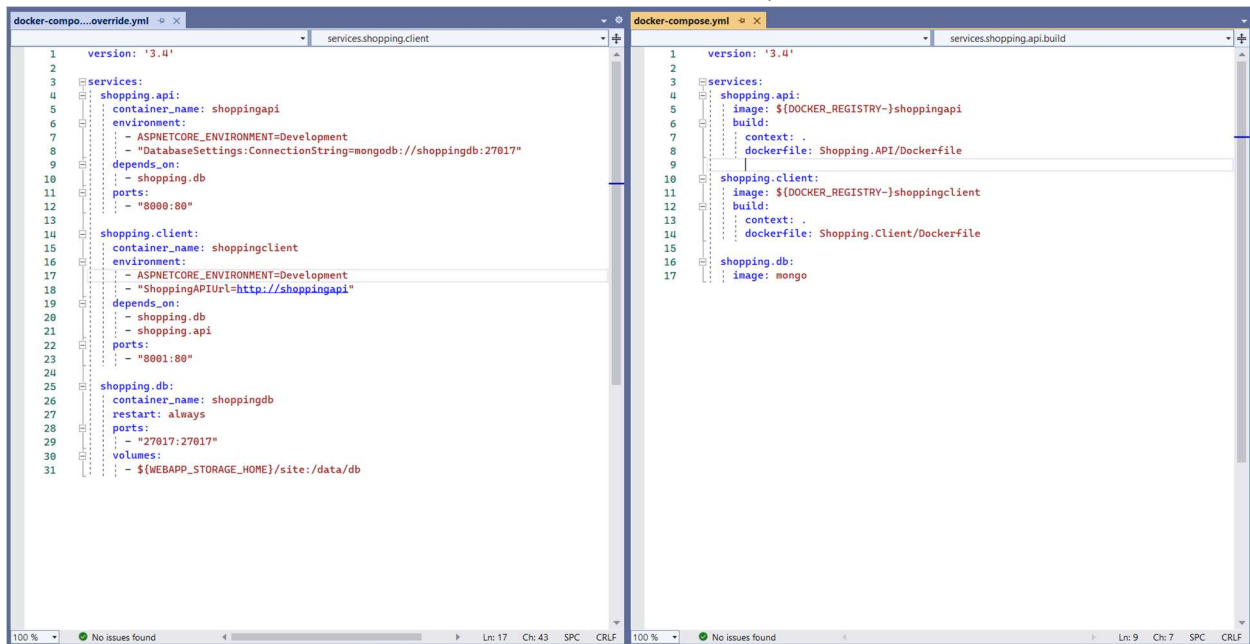
We will get the official images of Mongo on docker hub and we will use the suggested code to pull Mongo image in our local computer:

```
docker pull mongo
```

Then we run the existing image from my local and create a container, indicating the same port number:

```
docker run -d -p 27017:27017 --name shopping-mongo mongo
```

- Currently, we are in the phase where we will use Docker Compose to multi-containerize all Microservices. You can create several container definitions in a single file using Docker Compose, and you can execute the application by bringing up all the prerequisites that it requires. For Shopping.API, we must first construct a DockerFile, but this time, we also require orchestration with Client-API and MongoDB, thus we must also create docker compose. A docker-compose.yml and override file are created by Visual Studio and placed in the solution's docker-compose node. All the services that will be deployed in a Docker environment are defined in the Docker Compose file.



**Figure 45: docker-compose.override.yml & docker-compose.yml**

In our case we have also Shopping.Client application, where we do the same procedure. Now both services are included. One more left as we will add MongoDB database. We are going to run docker-compose and we have shoppingclient on port 8001, shoppingapi on port 8000, mongo on port 27017.



```

Developer PowerShell
+ Developer PowerShell
** Visual Studio 2022 Developer PowerShell v17.3.0
** Copyright (c) 2022 Microsoft Corporation
*****
PS C:\Users\Klementin Hasani\source\repos\run-devops\Shopping> docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED        STATUS        PORTS                               NAMES
c0d960f0f485   shoppingclient    "tail -f /dev/null"     3 hours ago   Up 3 hours   0.0.0.0:8001->80/tcp               shoppingclient
cf4d9d2f31ad   shoppingapi       "tail -f /dev/null"     3 hours ago   Up 3 hours   0.0.0.0:8000->80/tcp               shoppingapi
720b96d8a6a6   mongo             "docker-entrypoint.s..." 3 hours ago   Up 3 hours   0.0.0.0:27017->27017/tcp           shoppingdb
PS C:\Users\Klementin Hasani\source\repos\run-devops\Shopping> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
shoppingapi    latest   3c8eafa49175   22 hours ago   226MB
shoppingclient latest   2416f3a6f243   22 hours ago   234MB
<none>         <none>   822eb4c61f9b   22 hours ago   226MB
<none>         <none>   5c11494679c8   22 hours ago   234MB
<none>         <none>   fdc0f1d7e9ce   22 hours ago   226MB
mongo          latest   d98599fddf65   8 days ago     696MB

```

**Figure 46: PowerShell – docker ps & docker images**

- Test the application, where we will see API and Client pages:

<http://localhost:8000/swagger/index.html>  
<http://localhost:8001/product>

As we can see at *Figure 46: PowerShell – docker ps & docker images*, we have created docker images for our microservices and we compose docker container. So now, we are going to deploy these docker container images on Kubernetes clusters. We are going to install and run Kubernetes on local environment. On Kubernetes section that we find at Docker Settings, we will enable Kubernetes.

- We are going to use Visual Studio Code, to create our yaml files. First, we create the folder **k8s** and inside it, **mongo.yaml** that will have this structure:

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo-deployment
  labels:
    app: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:

```

```

containers:
  - name: mongodb
    image: mongo
    ports:
      - containerPort: 27017
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
    env:
      - name: MONGO_INITDB_ROOT_USERNAME
        valueFrom:
          secretKeyRef:
            name: mongo-secret
            key: mongo-root-username
      - name: MONGO_INITDB_ROOT_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mongo-secret
            key: mongo-root-password
---
apiVersion: v1
kind: Service
metadata:
  name: mongo-service
spec:
  selector:
    app: mongodb
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017

```

As you can see that we have used `valueFrom` and `secretKeyRef` in order to access secret data. K8s manage our data with these definitions. It includes the Deployment and Service.

- Now, we are going to create mongo-secret.yaml and mongo-configmap.yaml file, that it looks like below:

```

apiVersion: v1
kind: Secret
metadata:

```

```

name: mongo-secret
type: Opaque
data:
  mongo-root-username: dXN1cm5hbWU=
  mongo-root-password: cGFzc3dvcmQ=

```

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: mongo-configmap
data:
  connection_string: mongodb://username:password@mongo-service:27017

```

Through the code:

`kubectl apply -f .\mongo.yaml`, we will have a mongo secret definition and we apply the yaml file into our local Kubernetes.

`kubectl apply -f .\mongo-configmap.yaml`, this file will store mongo connection string information.

- We are going to build Shopping Docker Images, tag and push to docker hub. By this way, Kubernetes retrieves images from the docker hub. For mongodb, k8s pull the official mongo docker hub image. I am going to run docker compose command.

```
docker-compose -f docker-compose.yml -f docker-compose.override.yml up -d
```

Now, I have a shopping database, API and client. To stop and remove the container, we will write the code:

```
docker-compose -f docker-compose.yml -f docker-compose.override.yml down.
```

Next step is to tag and push to docker hub. Start with both images, shoppingapi and shoppingclient.

```

docker tag f5cd1c0f8307 7991575001/shoppingapi
docker tag 2416f3a6f243 7991575001/shoppingclient

```

```

PS C:\Users\Klementin Hasani\source\repos\run-devops\Shopping> docker images
REPOSITORY              TAG                IMAGE ID           CREATED           SIZE
7991575001/shoppingapi   latest            f5cd1c0f8307      2 days ago       226MB
shoppingapi              latest            f5cd1c0f8307      2 days ago       226MB
7991575001/shoppingclient latest            2416f3a6f243      2 days ago       234MB
shoppingclient           latest            2416f3a6f243      2 days ago       234MB

```

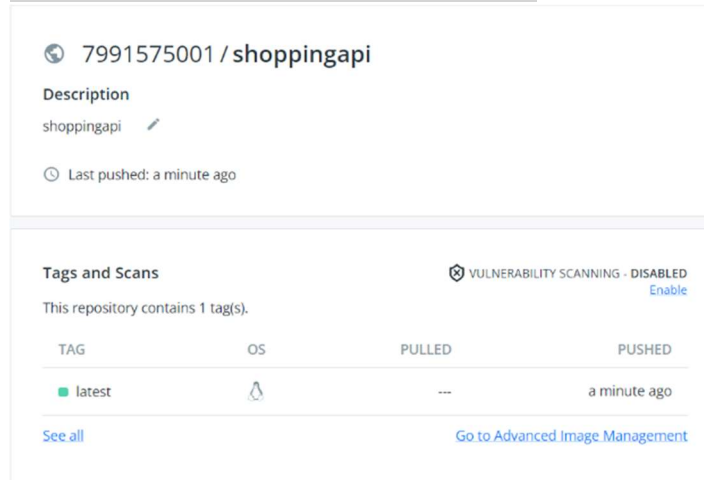
**Figure 47: PowerShell – docker images**

These two are our public repositories, where we going to push our images.

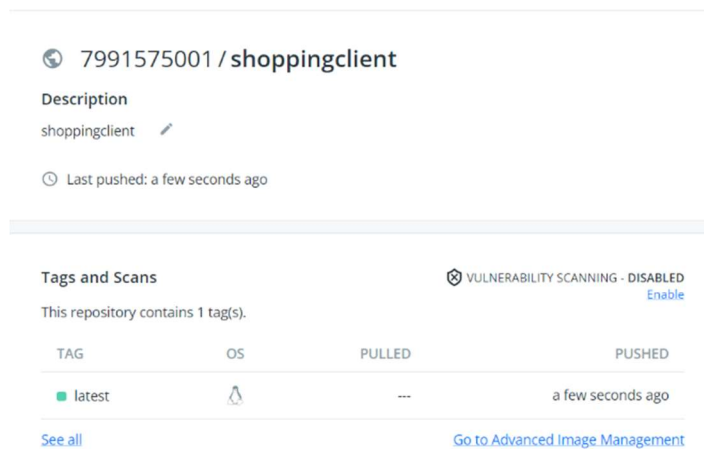
```
docker login
```

```
docker push 7991575001/shoppingapi
```

```
docker push 7991575001/shoppingclient
```



**Figure 48: shoppingapi repositories – push successfully**



**Figure 49: shoppingclient repositories – push successfully**

- We are going to create shopping.API Kubernetes Deployment and Service yaml file, under Kubernetes folder.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: shoppingapi-deployment
  labels:
```

```

    app: shoppingapi
spec:
  replicas: 1
  selector:
    matchLabels:
      app: shoppingapi
  template:
    metadata:
      labels:
        app: shoppingapi
    spec:
      containers:
        - name: shoppingapi
          image: 7991575001/shoppingapi:latest
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 80
          env:
            - name: ASPNETCORE_ENVIRONMENT
              value: Development
            - name: DatabaseSettings__ConnectionString
              valueFrom:
                configMapKeyRef:
                  name: mongo-configmap
                  key: connection_string
      resources:
        requests:
          memory: "64Mi"
          cpu: "250m"
        limits:
          memory: "128Mi"
          cpu: "500m"
---
apiVersion: v1
kind: Service
metadata:
  name: shoppingapi-service
spec:
  type: NodePort
  selector:
    app: shoppingapi
  ports:
    - protocol: TCP
      port: 8000
      targetPort: 80

```

```
nodePort: 31000
```

In order to add test our API, we set the service as a nodeport. So we can run our shoppingapi yaml file on k8s. The shopping.API deployment works with configmap, so together with Mongo project are running on K8s cluster, successfully.

- We are going to create shopping.client Kubernetes deployment and service yaml file, under Kubernetes folder. Also we are going to create shoppingapi-configmap.yaml file, the same way with mongo-configmap.yaml, for storing shopping.API url.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: shoppingapi-configmap
data:
  shoppingapi_url: http://shoppingapi-service:8000
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: shoppingclient-deployment
  labels:
    app: shoppingclient
spec:
  replicas: 1
  selector:
    matchLabels:
      app: shoppingclient
  template:
    metadata:
      labels:
        app: shoppingclient
    spec:
      containers:
        - name: shoppingclient
          image: 7991575001/shoppingclient:latest
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 80
          env:
            - name: ASPNETCORE_ENVIRONMENT
              value: Development
```

```

      - name: ShoppingAPIUrl
        valueFrom:
          configMapKeyRef:
            name: shoppingapi-configmap
            key: shoppingapi_url
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
---
apiVersion: v1
kind: Service
metadata:
  name: shoppingclient-service
spec:
  type: NodePort
  selector:
    app: shoppingclient
  ports:
    - protocol: TCP
      port: 8001
      targetPort: 80
      nodePort: 30000

```

We are defining the **NodePort** because we would like to call from external system, in our case local environment. As you can see that, we have finally deployed our microservices into local Kubernetes environment.

```
kubectl apply -f .\shoppingapi-configmap.yaml
kubectl apply -f .\shoppingclient.yaml
```

- Shoppingclient => <http://localhost:30000>
- ShoppingApi => <http://localhost:31000>

```

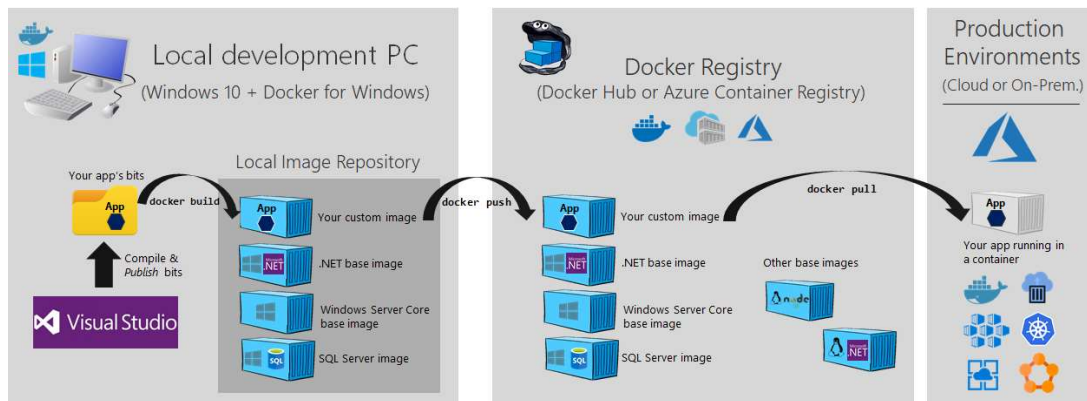
PS C:\Users\Klementin Hasani\source\repos\run-devops\k8s> kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
mongo-deployment-59698586f8-8dgj2   1/1     Running   0           44h
shoppingapi-deployment-974cf7f4c-88dvc 1/1     Running   0           103m
shoppingclient-deployment-74f5c75756-x7jm4 1/1     Running   0           39s

```

**Figure 50: PowerShell – kubectl get pod**

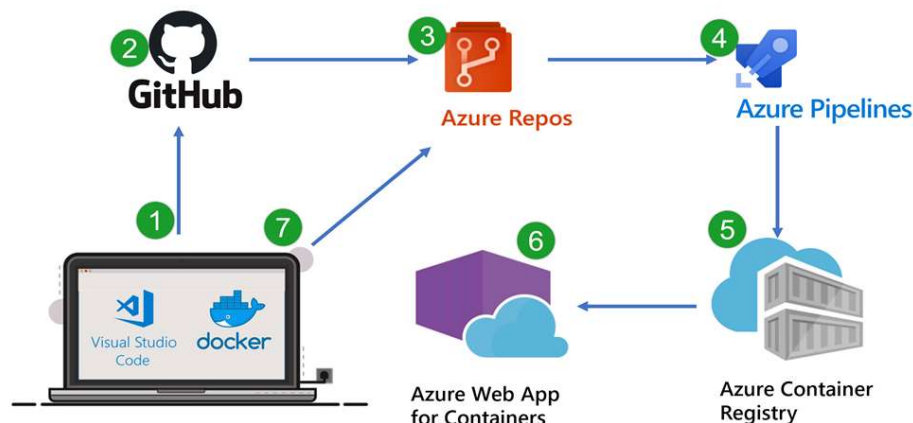
## 9.2.4 Azure Kubernetes Services deployment

In this section, we will deploy shopping microservices into Cloud Azure Kubernetes Service with using Azure Container Registry. So far, we have docker and Kubernetes deployment on local environment also test it. Now, it is time to move Kubernetes to the cloud on which is Azure Kubernetes Service.



**Figure 51: Step of the process from local to cloud**

As you can see at the picture, we have already finished our local development, dockerize all images and push the docker hub. But this time, we will push image to Azure Container Registry (ACR), and deploy our current Kubernetes configurations into the Azure Kubernetes Service (AKS) with pull images from ACR.



**Figure 52: Structure of ACR DevOps Pipeline**



Example use of ACR DevOps Pipeline

- We are going to deploy an ACR instance and push a container image to it. Firstly, we should download and install Azure CLI. You can check if it is installed: `az --version`. Then we login on Azure and need to create a resource group to make possible the creation of ACR. Azure resource group is a logical container into which Azure resources are deployed and managed. Code below for the creation of resource group:

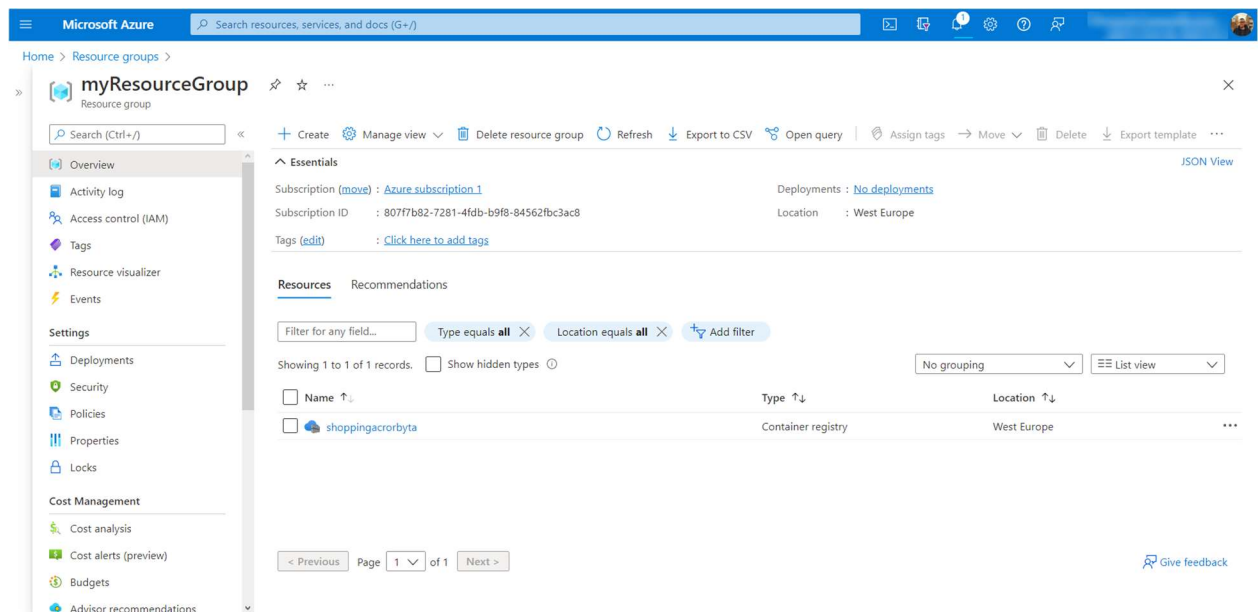
```
PS C:\Users\Klementin Hasani\source\repos\run-devops\Shopping> az group create --name myResourceGroup --location westeurope
{
  "id": "/subscriptions/807f7b82-7281-4fdb-b9f8-84562fbc3ac8/resourceGroups/myResourceGroup",
  "location": "westeurope",
  "managedBy": null,
  "name": "myResourceGroup",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

**Figure 53: Creation of resource group - command line**

Now, we go for the creation of Azure Container Registry instance with the command:

```
az acr create --resource-group myResourceGroup --name shoppingacrorbyta --sku Basic
```

and we must provide our own registry name, which must be global unique. We can verify that we created ACR correctly, through the Microsoft Azure portal.



**Figure 54: Azure Container Registry**

It will be good to enable **admin user** at portal because it will be needed when you deploy a container image in the portal from the registry directly to Azure container registry or azure web app for containers. Since we are going to deploy our images into Azure

Kubernetes service, so that is good to use the admin account in the Azure container registry. You can do it through the code: `az acr update -n shoppingacrorbyta --admin-enabled true`, or at Access keys section that you can find at Settings of the container.

- In order to use shopping containers image with ACR, the image needs to be tagged with the login server address of your registry.

```
docker tag shoppingapi:latest shoppingacrorbyta.azurecr.io/shoppingapi:v1
docker tag shoppingclient:latest shoppingacrorbyta.azurecr.io/shoppingclient:v1
```

```
PS C:\Users\Klementin Hasani\source\repos\run-devops\Shopping> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
shoppingacrorbyta.azurecr.io/shoppingapi	v1	f5cd1c0f8307	6 days ago	226MB
shoppingapi	latest	f5cd1c0f8307	6 days ago	226MB
shoppingclient	latest	e2af377c8b80	6 days ago	234MB
shoppingacrorbyta.azurecr.io/shoppingclient	v1	e2af377c8b80	6 days ago	234MB
kubernetesui/dashboard	v2.6.0	1042d9e0d8fc	2 months ago	246MB
kubernetesui/metrics-scraper	v1.0.8	115053965e86	2 months ago	43.8MB
k8s.gcr.io/kube-apiserver	v1.24.0	529072250ccc	3 months ago	130MB
k8s.gcr.io/kube-proxy	v1.24.0	77b49675beae	3 months ago	110MB
k8s.gcr.io/kube-scheduler	v1.24.0	e3ed7dee73e9	3 months ago	51MB
k8s.gcr.io/kube-controller-manager	v1.24.0	88784fb4ac2f	3 months ago	119MB
k8s.gcr.io/etcd	3.5.3-0	aebe758cef4c	4 months ago	299MB
k8s.gcr.io/pause	3.7	221177c6082a	5 months ago	711kB
k8s.gcr.io/coredns/coredns	v1.8.6	a4ca41631cc7	10 months ago	46.8MB
docker/desktop-vpnkit-controller	v2.0	8c2c38aa676e	15 months ago	21MB
docker/desktop-storage-provisioner	v2.0	99f89471f470	15 months ago	41.9MB
k8s.gcr.io/kubernetes-dashboard-amd64	v1.10.1	f9aed6605b81	3 years ago	122MB

**Figure 55: tag image containers**

After we tagged successfully, the images are ready for pushing to the Azure Container Registry.

```
docker push shoppingacrorbyta.azurecr.io/shoppingapi:v1
docker push shoppingacrorbyta.azurecr.io/shoppingclient:v1
```

```
PS C:\Users\Klementin Hasani\source\repos\run-devops\Shopping> az acr repository list --name shoppingacrorbyta --output table
```

Result
shoppingapi
shoppingclient

**Figure 56: ACR repository**

We have pushed correctly our two microservice images into ACR. Finally, we have finished to create ACR, enable admin account and push our shopping images to the ACR.

- Now, we are going to deploy a Kubernetes AKS cluster with attaching ACR. Install the Kubernetes CLI and configure kubectl to connect to our AKS cluster. It is a mandatory step in order to connect to the cloud AKS. The code below, describe the process:

```
az aks create --resource-group myResourceGroup --name myAKSCluster --node-count 1 --generate-ssh-keys --attach-acr shoppingacrorbyta
```

```
az aks install-cli
```

```
az aks get-credentials --resource-group myResourceGroup --name myAKSCluster
```

As you can see that we have created AKS and connect Kubernetes from our local computer with using kubectl commands.

```
PS C:\Users\Klementin Hasani\source\repos\run-devops> kubectl get all
NAME                                TYPE          CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
service/kubernetes                 ClusterIP     10.0.0.1     <none>        443/TCP    23m
PS C:\Users\Klementin Hasani\source\repos\run-devops> kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
aks-nodepool1-88532894-vmss000000 Ready    agent    23m   v1.22.11
```

**Figure 57: Azure Kubernetes Service creation – command line**

- We need to create image pull secret, which is used by Kubernetes to store information needed to authenticate to our registry.

```
PS C:\Users\Klementin Hasani\source\repos\run-devops> kubectl create secret docker-registry acr-secret --docker-server=shoppingacrorbyta.azurecr.io --docker-username=shoppingacrorbyta --docker-password=NGJCKxPwB8xJ5P/Cx0dounExFngV1P6 --docker-email=klementin.hasani@orbyta.it
secret/acr-secret created
PS C:\Users\Klementin Hasani\source\repos\run-devops> kubectl get secret
NAME                                TYPE          DATA   AGE
acr-secret                         kubernetes.io/dockerconfigjson 1       10m
default-token-s76v9               kubernetes.io/service-account-token 3       144m
```

**Figure 58: Pull-Secret image**

In order to use these image pull-secret, we should add the configuration. Once you have created the image, you can use it creating Kubernetes pods and deployments. So, we need to provide the name of the secret under the image pull-secret section in the deployment files.

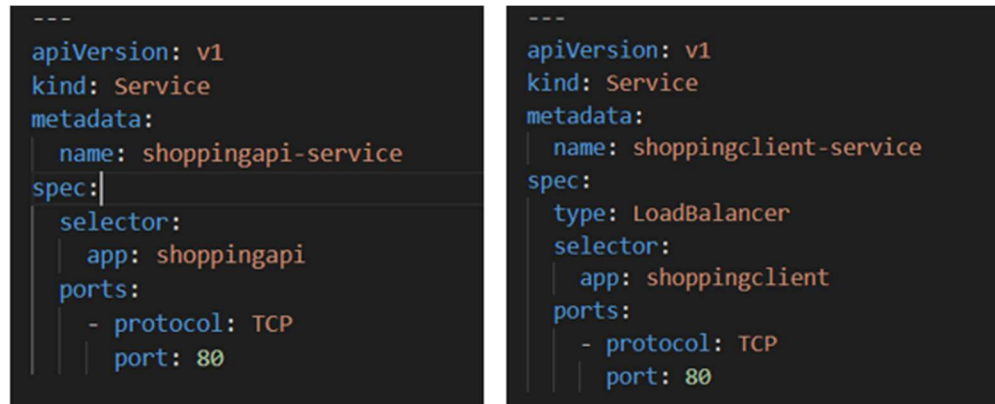
- After the creation of Azure Kubernetes Service and pull-secret, we are going to build and deploy Shopping applications and services into an AKS cluster.
  - First of all, we need to edit existing K8s manifest yaml files for deploying AKS. To remember that AKS folder has the same file like at K8s folder, but with some changes that we will make.



**Figure 59: shopping.client & shopping.api deployment section**

We have replaced image names and we edit image-pull secret configurations into deployment container configurations, in order to allow to pulling image from AKS.

- Now we have to update service section and the images below shows us the new look of service.



**Figure 60: : shopping.api & shopping.client service section**

We are going to deploy AKS on line cloud so we do not need to port-forwarding. Every pod take new IP from Cloud AKS. We set the default port = 80 and we do not have anymore targetPort.

- Now it is time to run configured K8s Manifest yaml files on AKS and deploy the application.

Run the command: `kubectl apply -f .\aks\`

We can check, all k8s resource created successfully on the AKS. We have PODs, Services, Deployments and Replica Sets.

```
PS C:\Users\Klementin Hasani\source\repos\run-devops> kubectl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/mongo-deployment-9c5b4dddb-wl24q	1/1	Running	0	33s
pod/shoppingapi-deployment-5f68b746d9-9fbdz	1/1	Running	0	33s
pod/shoppingclient-deployment-d76cbfcf6-xzn2w	1/1	Running	0	32s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.0.0.1	<none>	443/TCP	7h11m
service/mongo-service	ClusterIP	10.0.207.6	<none>	27017/TCP	33s
service/shoppingapi-service	ClusterIP	10.0.227.171	<none>	80/TCP	32s
service/shoppingclient-service	LoadBalancer	10.0.10.220	20.103.202.190	80:30078/TCP	32s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/mongo-deployment	1/1	1	1	33s
deployment.apps/shoppingapi-deployment	1/1	1	1	33s
deployment.apps/shoppingclient-deployment	1/1	1	1	32s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/mongo-deployment-9c5b4dddb	1	1	1	33s
replicaset.apps/shoppingapi-deployment-5f68b746d9	1	1	1	33s
replicaset.apps/shoppingclient-deployment-d76cbfcf6	1	1	1	32s

**Figure 61: : Kubernetes resource: PODs, Services, Deployments, Replica Sets**

We can test the application at our EXTERNAL-IP: <http://20.103.202.190>

Shopping.Client Home Privacy

## Products Single Container

Name	Category	Description	ImageFile	Price
Tesla Model X	Hybrid	The Tesla Model X is a battery electric mid-size luxury crossover produced by Tesla, Inc. since 2015.	product-1.png	40.00
Audi Q8	Diesel	The Audi Q8 is a mid-size luxury crossover SUV coupé made by Audi that was launched in 2018.	product-2.png	84.00
Mercedes-Benz GLA	Diesel	The Mercedes-Benz GLA is a subcompact luxury crossover SUV manufactured and marketed by Mercedes-Benz over two generations.	product-3.png	75.00
Fiat 500	Petrol	The Fiat 500 is a rear-engined, four-seat, small city car that was manufactured and marketed by Fiat Automobiles from 1957 until 1975 over a single generation in two-door saloon and two-door station wagon bodystyles.	product-4.png	22.00
Chery Tiggo 5	Diesel/Hybrid	The Chery Tiggo 5 is a compact crossover produced by Chery under the Tiggo product series.	product-5.png	38.00
Jeep Compass	Hybrid	The Jeep Compass is a compact crossover SUV introduced for the 2007 model year, and is now in its second generation.	product-6.png	44.00

© 2022 - Shopping.Client - [Privacy](#)

**Figure 62: shopping.client webpage – External IP**

- We have a working Kubernetes cluster in AKS and we deployed the Shopping microservices. The next step will be the autoscale of the pods in the shopping app. On AKS folder, we create the new yaml file named shoppingautoscale. We will give a minimum and maximum of replica count.

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: shoppingapi-hpa
```



```
spec:
  maxReplicas: 10 # define max replica count
  minReplicas: 2 # define min replica count
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: shoppingapi-deployment
  targetCPUUtilizationPercentage: 50 # target CPU utilization
---
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: shoppingclient-hpa
spec:
  maxReplicas: 10 # define max replica count
  minReplicas: 3 # define min replica count
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: shoppingclient-deployment
  targetCPUUtilizationPercentage: 50 # target CPU utilization
```

We will go to deploy **v2** of shopping client microservices to AKS with zero downtime. Firstly, we make a change at index client file and then we will update container image by tagging and pushing the new client image **v2**. We start updating the code on client yaml file by upgrading from one to two the version and then deploy microservices to AKS.

ShoppingClient Home Privacy

---

### Products - Version Update 2.0 from AKS Deployment

Name	Category	Description	ImageFile	Price
Tesla Model X	Hybrid	The Tesla Model X is a battery electric mid-size luxury crossover produced by Tesla, Inc. since 2015.	product-1.png	40.00
Audi Q8	Diesel	The Audi Q8 is a mid-size luxury crossover SUV coupé made by Audi that was launched in 2018.	product-2.png	84.00
Mercedes-Benz GLA	Diesel	The Mercedes-Benz GLA is a subcompact luxury crossover SUV manufactured and marketed by Mercedes-Benz over two generations.	product-3.png	75.00
Fiat 500	Petrol	The Fiat 500 is a rear-engined, four-seat, small city car that was manufactured and marketed by Fiat Automobiles from 1957 until 1975 over a single generation in two-door saloon and two-door station wagon bodystyles.	product-4.png	22.00
Chery Tiggo 5	Diesel/Hybrid	The Chery Tiggo 5 is a compact crossover produced by Chery under the Tiggo product series.	product-5.png	38.00
Jeep Compass	Hybrid	The Jeep Compass is a compact crossover SUV introduced for the 2007 model year, and is now in its second generation.	product-6.png	44.00

---

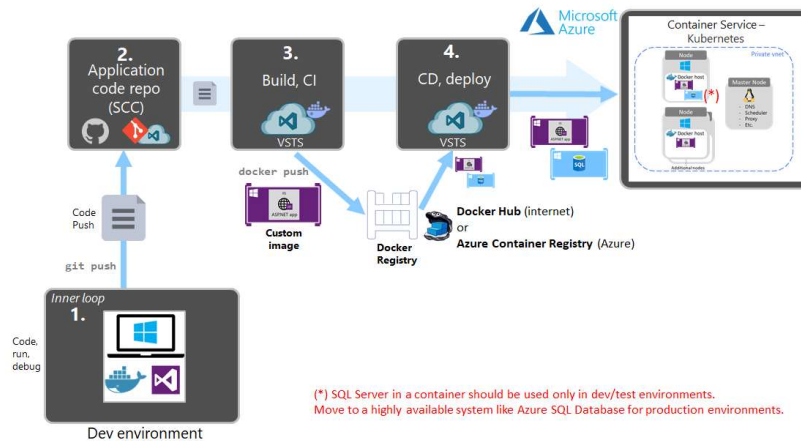
© 2022 - ShoppingClient - [Privacy](#)

**Figure 63: shopping.client webpage – Update V2**

On <http://20.103.202.190>, this is the updated view of the shopping client webpage. We can get back by changing the version at shoppingclient.yaml.

- Our main target is to automate deployments of Shopping Microservices into AKS with using Azure CI/CD Pipelines. This is the example scenario that that explain the process:

Scenario: Deploy to Kubernetes through CI/CD pipelines



**Figure 64: Automate scenario process**

In order to work with Azure Pipelines we need to create an organization first and then the project called “Shopping” inside of the organization. We create the folder pipeline at Visual Studio and into that we added the shoppingapi-pipeline.yaml.

```
# Deploy to Azure Kubernetes Service
# Build and push image to Azure Container Registry; Deploy to Azure Kubernetes
Service
# https://docs.microsoft.com/azure/devops/pipelines/languages/docker

trigger:
  branches:
    include:
      - main
  paths:
    include:
      - Shopping/Shopping.API/*
      - aks/shoppingapi.yaml

resources:
- repo: self

variables:

  # Container registry service connection established during pipeline creation
  dockerRegistryServiceConnection: 'f3961f5d-b552-46dc-b700-a86d951bc5a5'
  imageRepository: 'shoppingapi'
```

```

containerRegistry: 'shoppingacrorbyta.azurecr.io'
dockerfilePath: '**/Dockerfile'
tag: '$(Build.BuildId)'
imagePullSecret: 'shoppingacrorbyta1071d5af-auth'

# Agent VM image name
vmImageName: 'ubuntu-latest'

stages:
- stage: Build
  displayName: Build stage
  jobs:
  - job: Build
    displayName: Build
    pool:
      vmImage: $(vmImageName)
    steps:
    - task: Docker@2
      displayName: Build and push an image to container registry
      inputs:
        command: buildAndPush
        repository: $(imageRepository)
        dockerfile: $(dockerfilePath)
        containerRegistry: $(dockerRegistryServiceConnection)
        buildContext: $(Build.SourcesDirectory)/Shopping
        tags: |
          $(tag)

    - upload: aks
      artifact: aks

- stage: Deploy
  displayName: Deploy stage
  dependsOn: Build

  jobs:
  - deployment: Deploy
    displayName: Deploy
    pool:
      vmImage: $(vmImageName)
    environment: 'klementin277218rundevoops.default'
    strategy:
      runOnce:
        deploy:

```



```

steps:
- task: KubernetesManifest@0
  displayName: Create imagePullSecret
  inputs:
    action: createSecret
    secretName: $(imagePullSecret)
    dockerRegistryEndpoint: $(dockerRegistryServiceConnection)

- task: KubernetesManifest@0
  displayName: Deploy to Kubernetes cluster
  inputs:
    action: deploy
    manifests: |
      $(Pipeline.Workspace)/aks/shoppingapi.yaml

    imagePullSecrets: |
      $(imagePullSecret)
    containers: |
      $(containerRegistry)/$(imageRepository):$(tag)

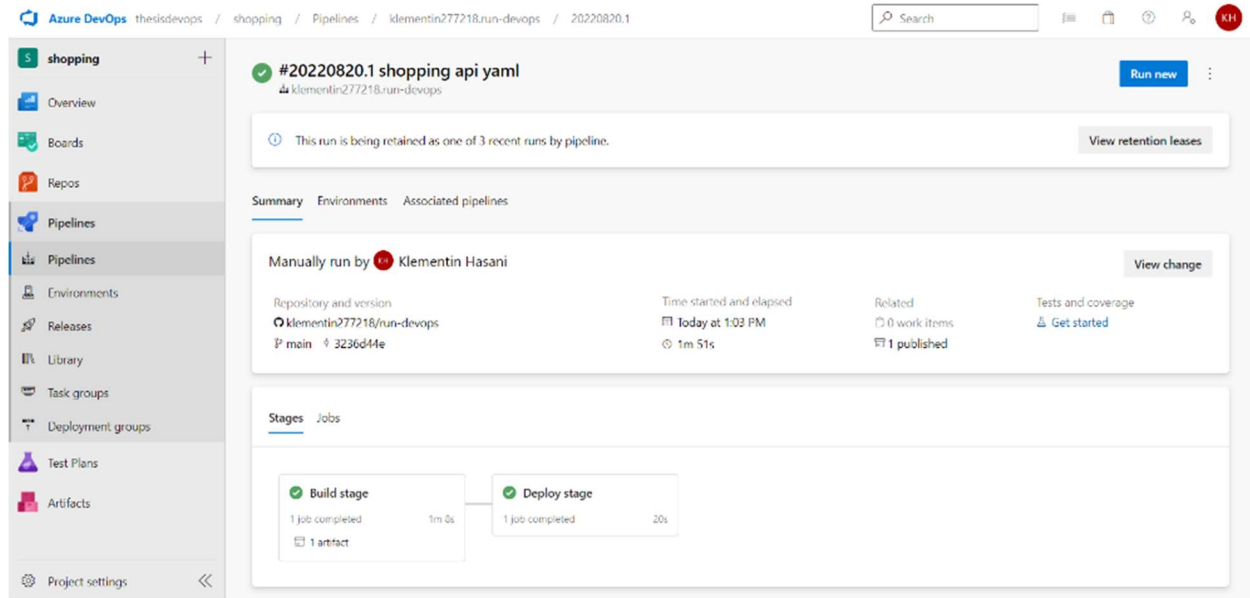
```

Now, we are going to create our first pipeline for shoppingapi microservices. Go to pipeline section and create it with these conditions:

**Table 6: Configure Pipeline shoppingapi**

CONFIGURE YOUR PIPELINE	SELECTED TEMPLATE
Connect	Github
Select a repository	run-devops (private repository)
Configure your pipeline	Existing Azure Pipelines Yaml file
Path	/pipelines/shoppingapi-pipeline.yaml
Review your pipeline YAML	Run

We have successfully build and deploy our shopping.api microservices with full automation, with writing our pipelines yaml file.



**Figure 65: Shopping API - pipeline**

Shopping API pipeline will trigger automatically if only, in shopping API class will be any change. As you can see, we should have separate pipelines for microservices, because we developed multi-container microservices application.

- Into folder pipeline, we also create shoppingclient-pipeline.yaml, the same structure as the shoppingapi-pipeline.yaml:

```
# Deploy to Azure Kubernetes Service
# Build and push image to Azure Container Registry; Deploy to Azure Kubernetes Service
# https://docs.microsoft.com/azure/devops/pipelines/languages/docker

trigger:
  branches:
    include:
      - main
  paths:
    include:
      - Shopping/Shopping.Client/*
      - aks/shoppingclient.yaml

resources:
- repo: self

variables:
```

```

# Container registry service connection established during pipeline creation
dockerRegistryServiceConnection: 'f3961f5d-b552-46dc-b700-a86d951bc5a5'
imageRepository: 'shoppingclient'
containerRegistry: 'shoppingacrorbyta.azurecr.io'
dockerfilePath: 'Shopping/Shopping.Client/Dockerfile'
tag: '$(Build.BuildId)'
imagePullSecret: 'shoppingacrorbyta1071d5af-auth'

# Agent VM image name
vmImageName: 'ubuntu-latest'

stages:
- stage: Build
  displayName: Build stage
  jobs:
  - job: Build
    displayName: Build
    pool:
      vmImage: $(vmImageName)
    steps:
    - task: Docker@2
      displayName: Build and push an image to container registry
      inputs:
        command: buildAndPush
        repository: $(imageRepository)
        dockerfile: $(dockerfilePath)
        containerRegistry: $(dockerRegistryServiceConnection)
        buildContext: $(Build.SourcesDirectory)/Shopping
        tags: |
          $(tag)

    - upload: aks
      artifact: aks

- stage: Deploy
  displayName: Deploy stage
  dependsOn: Build
  jobs:
  - deployment: Deploy
    displayName: Deploy
    pool:
      vmImage: $(vmImageName)
    environment: 'klementin277218rundeavors.default'

```

```

strategy:
  runOnce:
    deploy:
      steps:
        - task: KubernetesManifest@0
          displayName: Create imagePullSecret
          inputs:
            action: createSecret
            secretName: $(imagePullSecret)
            dockerRegistryEndpoint: $(dockerRegistryServiceConnection)

        - task: KubernetesManifest@0
          displayName: Deploy to Kubernetes cluster
          inputs:
            action: deploy
            manifests: |
              $(Pipeline.Workspace)/aks/shoppingclient.yaml
            imagePullSecrets: |
              $(imagePullSecret)
            containers: |
              $(containerRegistry)/$(imageRepository):$(tag)

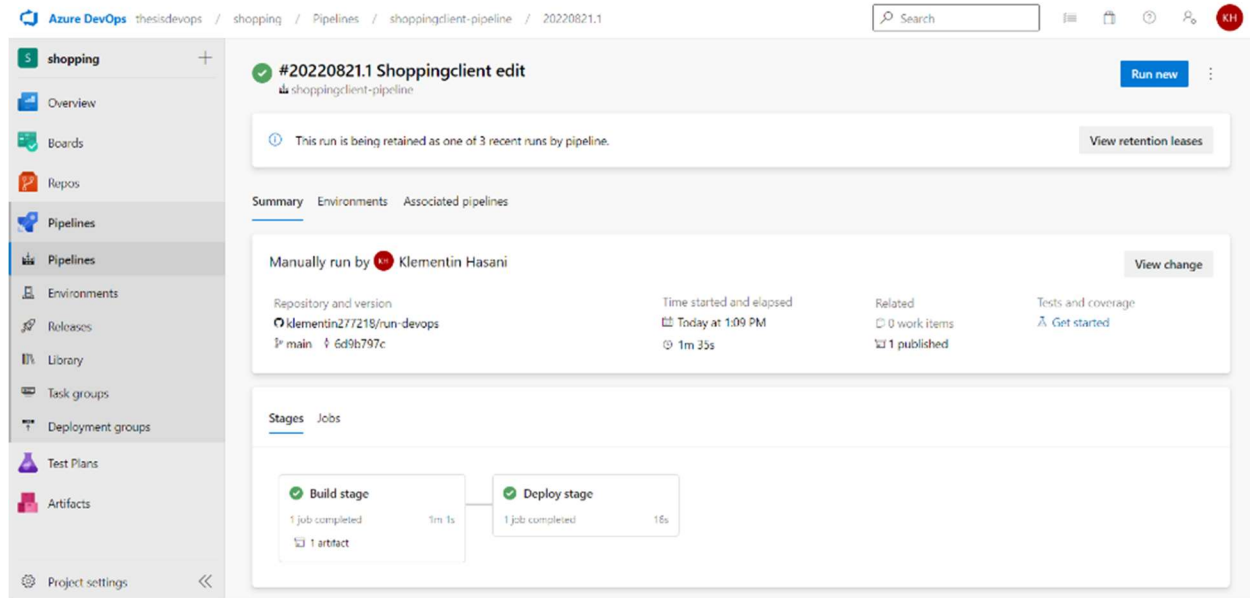
```

- Now, it is time to create the other pipeline for shoppingclient microservices.

**Table 7: Configure Pipeline shoppingclient**

CONFIGURE YOUR PIPELINE	SELECTED TEMPLATE
Connect	Github
Select a repository	run-devops (private repository)
Configure your pipeline	Existing Azure Pipelines Yaml file
Path	/pipelines/shoppingclient-pipeline.yaml
Review your pipeline YAML	Run

We have successfully build and deploy our shopping.client microservices with full automation, with writing our pipelines yaml file.



**Figure 66: Shopping Client – pipeline**

- I would like to check our client application and in order to do that, we should go to Services and ingresses of myAKSCluster. You can see the external IP of shoppingclient-pipeline, which is generated from the Azure Kubernetes Services.

Services		Ingresses						
Filter by service name		Filter by namespace						
Enter the full service name		All namespaces						
<input type="checkbox"/>	Name	Namespace	Status	Type	Cluster IP	External IP	Ports	Age ↓
<input type="checkbox"/>	kubernetes	default	✓ Ok	ClusterIP	10.0.0.1		443/TCP	4 days
<input type="checkbox"/>	kube-dns	kube-system	✓ Ok	ClusterIP	10.0.0.10		53/UDP,53/TCP	4 days
<input type="checkbox"/>	metrics-server	kube-system	✓ Ok	ClusterIP	10.0.254.33		443/TCP	4 days
<input type="checkbox"/>	mongo-service	default	✓ Ok	ClusterIP	10.0.207.6		27017/TCP	4 days
<input type="checkbox"/>	shoppingapi-service	default	✓ Ok	ClusterIP	10.0.227.171		80/TCP	4 days
<input type="checkbox"/>	shoppingclient-service	default	✓ Ok	LoadBalancer	10.0.10.220	20.103.202.190	80:30078/TCP	4 days

**Figure 67: Services on AKS**

The shoppingclient url is <http://20.103.202.190> and the output of the webpage as we expected:

Shopping.Client   Home   Privacy

---

## Products - Version Update 2.0 from AKS Deployment

Name	Category	Description	ImageFile	Price
Tesla Model X	Hybrid	The Tesla Model X is a battery electric mid-size luxury crossover produced by Tesla, Inc. since 2015.	product-1.png	40.00
Audi Q8	Diesel	The Audi Q8 is a mid-size luxury crossover SUV coupé made by Audi that was launched in 2018.	product-2.png	84.00
Mercedes-Benz GLA	Diesel	The Mercedes-Benz GLA is a subcompact luxury crossover SUV manufactured and marketed by Mercedes-Benz over two generations.	product-3.png	75.00
Fiat 500	Petrol	The Fiat 500 is a rear-engined, four-seat, small city car that was manufactured and marketed by Fiat Automobiles from 1957 until 1975 over a single generation in two-door saloon and two-door station wagon bodystyles.	product-4.png	22.00
Chery Tiggo 5	Diesel/Hybrid	The Chery Tiggo 5 is a compact crossover produced by Chery under the Tiggo product series.	product-5.png	38.00
Jeep Compass	Hybrid	The Jeep Compass is a compact crossover SUV introduced for the 2007 model year, and is now in its second generation.	product-6.png	44.00

---

© 2022 - Shopping.Client - [Privacy](#)

**Figure 68: Shopping Client webpage – The last update**

In this chapter, all microservices and deployment procedures were established gradually. We were successful in automating CI/CD processes and launching multi-container microservices apps. In conclusion, the application of the theoretical component was our primary objective. We achieved to deploy our multi-container microservices applications with automating all deployment process separately.

## References

- [1] Microsoft, What is Azure DevOps? , <https://docs.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops>, Application Insights overview, <https://docs.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview>, What is Azure Repos?, <https://docs.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops>, What is Azure Boards?, <https://docs.microsoft.com/en-us/azure/devops/boards/get-started/what-is-azure-boards?view=azure-devops>
- [2] Crystal Bedell, DevOps Automation: Why It's a Necessity for IoT, <https://www.iiotworldtoday.com/2019/09/17/devops-automation-why-its-a-necessity-for-iiot/> , 17 September 2019
- [3] Michiel Mulders, What Every Dev Company Needs to Know about NoOps Development, <https://www.sitepoint.com/noops-development/> , 13 August 2019
- [4] Stephen Watts, A Brief History of DevOps, <https://www.bmc.com/blogs/devops-history/> , 29 March 2019, Configuration Management in DevOps, <https://www.bmc.com/blogs/devops-configuration-management/#:~:text=These%20include%20coding%2C%20building%2C%20testing,an%20organization%20to%20increase%20agility>, 8 March 2019, Deep dive into Azure Test Plans, <https://azure.microsoft.com/it-it/blog/deep-dive-into-azure-test-plans/>, 19 September 2018
- [5] Azure, DevOps tutorial-an introduction, <https://azure.microsoft.com/en-in/solutions/devops/>, Serverless computing, <https://azure.microsoft.com/en-gb/resources/cloud-computing-dictionary/what-is-serverless-computing/>
- [6] Peter John, Building a DevOps pipeline for your App: Git Strategy, <https://proandroiddev.com/building-a-devops-pipeline-for-your-app-git-strategy-44f719230950> , 17 August 2019
- [7] Patrick Porto, 4 branching workflows for Git, <https://medium.com/@patrickporto/4-branching-workflows-for-git-30d0aaee7bf> , 27 Feb 2018
- [8] Proandroiddev, building a devops pipeline for your app introduction, <https://medium.com/proandroiddev/building-a-devops-pipeline-for-your-app-introduction-3e35>
- [9] MuleSoft, Benefits of a DevOps Environment, <https://www.mulesoft.com/resources/api/devops-environment-benefits>
- [10] Daniele Fontani, DevOps is Dead, Long Live NoOps, <https://betterprogramming.pub/devops-noops-difference-504dfc4e9faa> , 11 Nov 2019
- [11] Rhubbit, Agile Manifesto: le basi da cui partire, <https://www.rhubbit.it/agile-manifesto-le-basi-da-cui-partire/>

- [12] TechWorld with Nana, DevOps Tutorial for Beginners, <https://www.youtube.com/watch?v=3c-iBn73dDE>, Kubernetes Tutorial for Beginners, <https://www.youtube.com/watch?v=X48VuDVv0do>
- [13] Rowan Haddad, What are the best git branching strategies, <https://www.flagship.io/git-branching-strategies/>, 8 March 2022
- [14] GitLab, A guide to getting started in DevOps, <https://page.gitlab.com/resources-ebook-beginners-guide-devops.html>, Introduction to GitLab Flow, [https://docs.gitlab.com/ee/topics/gitlab\\_flow.html](https://docs.gitlab.com/ee/topics/gitlab_flow.html)
- [15] Mehmet Ozkaya, Deploying .Net Microservices with K8s, AKS and Azure DevOps, <https://www.udemy.com/course/deploying-net-microservices-with-k8s-aks-and-azure-devops/>, Github Run-Devops, <https://github.com/aspnetrun/run-devops>
- [16] Maribel Capuñay, And... What DevOps is? CI? CD? , <https://www.linkedin.com/pulse/what-devops-ci-cd-maribel-capuñay/> , 2 April 2021
- [17] Beth Braccio Hering, DevOps Engineer Career: Salary, Job Description, and Skills, <https://www.flexjobs.com/blog/post/devops-engineer-career-job-description-salary/>
- [18] Harshit Agarwal, Roadmap to IT Revolution: History of DevOps, <https://www.appknox.com/blog/history-of-devops#:~:text=The%20concept%20of%20DevOps%20emerged,it%20became%20quite%20a%20buzzword> , 4 Oct 2019
- [19] NetApp, What is DevOps ?, <https://www.netapp.com/devops-solutions/what-is-devops/>
- [20] Kavya Tolety, Top 5 Companies using DevOps in 2021 – All you need to know, <https://www.edureka.co/blog/companies-using-devops/#hp>, 16 December 2021
- [21] Hiren Dhaduk, How Netflix Became A Master of DevOps? An Exclusive Case Study, <https://www.simform.com/blog/netflix-devops-case-study/>, 24 February 2022
- [22] Christopher Null, 10 companies killing it at DevOps in 2020, [https://content.microfocus.com/optimize-devops-tb/companies-killing-devops-2020?lx=vm26kZ&utm\\_source=techbeacon&utm\\_medium=referral&utm\\_campaign=7014J000000dVOkQAM&\\_ga=2.18426169.1261138638.1654694337-1449737907.1654694325&xs=182762](https://content.microfocus.com/optimize-devops-tb/companies-killing-devops-2020?lx=vm26kZ&utm_source=techbeacon&utm_medium=referral&utm_campaign=7014J000000dVOkQAM&_ga=2.18426169.1261138638.1654694337-1449737907.1654694325&xs=182762) , <https://techbeacon.com/app-dev-testing/10-companies-killing-it-devops>
- [23] Jaymin Vyas, DevOps and Cloud: A Symbiotic Relationship, <https://devops.com/devops-and-cloud-a-symbiotic-relationship/>, 7 November 2018



- [24] Azure DevOps Labs, Create a CI/CD pipeline for .NET with the DevOps Starter Project, <https://www.azuredevopslabs.com/labs/vstsextend/azuredevopsprojectdotnet/>, 21 January 2022
- [25] JavaTpoint, Azure DevOps Repository, <https://www.javatpoint.com/azure-devops-repository>
- [26] Troy Micka, All About Azure Artifacts, <https://newsignature.com/articles/all-about-azure-artifacts/#:~:text=What%20is%20Azure%20Artifacts%3F,with%20teams%20of%20any%20size>, 27 February 2020
- [27] Wrike, What is the scrum methodology?, <https://www.wrike.com/scrum-guide/scrum-methodology/> , What is Scrum in Agile? , <https://www.wrike.com/project-management-guide/faq/what-is-scrum-in-agile/>
- [28] The University of Arizona Global Campus, What is scrum? , <https://www.uagc.edu/blog/what-is-scrum>
- [29] Digite, What is Scrum?, <https://www.digite.com/agile/scrum-methodology/>
- [30] Chris Tozzi, Why Integrated Infrastructure is the key to IT Success, <https://devops.com/why-integrated-infrastructure-is-the-key-to-it-success/>, 21 November 2019
- [31] Ilon Adams, Serverless Computing vs Cloud Computing, <https://dzone.com/articles/serverless-computing-vs-cloud-computing#:~:text=With%20cloud%20computing%2C%20clients%20run,manage%20the%20OS%20or%20middleware>, 24 Jun 2021
- [32] Katie Lane, Kubernetes vs. Docker: What Does it Really Mean? , <https://www.sumologic.com/blog/kubernetes-vs-docker/>, 3 September 2020
- [33] NovelVista, Git! An important DevOps Tool, <https://www.sumologic.com/blog/kubernetes-vs-docker/>, 27 July 2021
- [34] Tomasz Lisowski, Top Git hosting services for 2022, <https://gitprotect.io/blog/top-git-hosting-services-for-2022/>, 8 December 2021
- [35] Wahab Chetara, What is DevOps? New Engineering Insight, <https://gofiha.com/what-is-devops/>, 17 Nov 2021
- [36] Mahsa Tehrani, What exactly is a DevOps Pipeline?, <https://community.atlassian.com/t5/DevOps-discussions/Let-s-talk-about-DevOps-resources-for-a-robust-secure-pipeline/td-p/1732879>
- [37] Hardeep Singh, Roadway to IT Revolution: The history of DevOps, <https://medium.com/appknox/roadway-to-it-revolution-the-history-of-devops-c6ca08104c1f>

- [38] Bibek Chatterjee, Legacy System Administration To Cloud Enabled DevOps Transformation Journey - Part 2, <https://www.linkedin.com/pulse/legacy-system-administration-cloud-enabled-devops-part-chatterjee-1f/>
- [39] Muhammad Raza & Shanika Wickramasinghe, Automation In DevOps: Why & How To Automate DevOps Practices, <https://www.bmc.com/blogs/automation-in-devops/#:~:text=DevOps%20automation%20is%20the%20practice,Software%20deployment%20and%20release> , 14 April 2021, What is DevOps? A Comprehensive Introduction, <https://www.bmc.com/blogs/devops-basics-introduction> , 16 April 2021
- [40] MuleSoft, How a DevOps environment transforms organizations, <https://www.mulesoft.com/resources/api/devops-environment-benefits>
- [41] Vineet Chaturvedi, DevOps Tutorial : Introduction To DevOps, <https://www.edureka.co/blog/devops-tutorial#:~:text=United%20Airlines%3A,the%20company%20to%20save%20%24500%2C000> , 19 April 2022
- [42] Bittu Kumar, What is DevOps and Azure DevOps? , <https://www.linkedin.com/pulse/what-devops-azure-bittu-kumar/> , 17 June 2021
- [43] Aaron Bjork, Deep dive into Azure Boards, <https://azure.microsoft.com/en-us/blog/deep-dive-into-azure-boards/> , 13 September 2018
- [44] Rafael Medeiros, Deploying to Azure Using Azure DevOps and Terraform, <https://rafaelmedeiros94.medium.com/deploying-to-azure-using-azure-devops-and-terraform-57044407790c> , 14 December 2021
- [45] Ravi Shanker, Deep dive into Azure Test Plans, <https://azure.microsoft.com/en-in/blog/deep-dive-into-azure-test-plans/> , 19 September 2018
- [46] ProfessionalDevOps, Azure Test Plan, <https://www.professional-devops.com/azure-test-plan.html>
- [47] Modern Requirements, Documenting Azure DevOps Test Plans, <https://www.modernrequirements.com/blogs/documenting-azure-devops-test-plans/>
- [48] ByWeeknd, What is Azure Artefacts?, <https://byweeknd.com/109527/> , 4 December 2021
- [49] Barry Luijbregts, Why you should be using Azure DevTest Labs, <https://www.azurebarry.com/why-you-should-be-using-azure-devtest-labs/> , 11 April 2017
- [50] Zainab Ahmed, Azure Application Insights overview, <https://www.alletec.com/blog/azure-application-insights-overview/>
- [51] Chris Seferlis, What is Application Insights? , <https://blog.pragmaticworks.com/what-is-application-insights> , 27 August 2018

- [52] Subhankar Sarkar, Azure DevOps – Manage your application lifecycle in cloud, <https://subhankarsarkar.com/azure-devops-manage-your-application-lifecycle-in-cloud/>
- [53] ProductPlan, How Agile product managers can build better products, <http://assets.productplan.com/content/Ship-It-How-Agile-Product-Managers-Can-Build-Better-Products-by-ProductPlan.pdf>
- [54] Ivan Porta, Azure Devops work items explained, <https://faun.pub/azure-devops-work-items-explained-10c4721c0880> , 29 September 2020
- [55] Luis Gonçalves, What’s epic, user story and task in scrum work hierarchy, <https://adaptmethodology.com/epic-user-story-task/> , 23 August 2022
- [56] Eplexity, Leveraging app patterns for AWS success, <https://eplexity.com/wp-content/uploads/2020/05/DevOps-Patterns-on-AWS-eBook.pdf>
- [57] Bilarasa, Serveless Computing in Azure Kubernetes Service Mit Keda, <https://bilarasa.com/serverless-computing-in-azure-kubernetes-service-mit-keda/> , 1 September 2022
- [58] Prudhvi Keshav, Kubernetes Basics, <https://prudhvikeshav.hashnode.dev/kubernetes-basics> , 11 May 2022
- [59] IBM Cloud, Kubernetes vs. Docker: Why Not Both?, <https://www.ibm.com/cloud/blog/kubernetes-vs-docker> , 13 June 2022
- [60] Sayeda Haifa Perveez, What is Git: Features, Command and Workflow in Git, <https://www.simplilearn.com/tutorials/git-tutorial/what-is-git> , 12 Jul 2022
- [61] Mushmad Mannambeth, DevOps: Git for Beginners! , <https://www.simplilearn.com/tutorials/git-tutorial/what-is-git> , 28 Jul 2021
- [62] Konrad Gadzinowski, Trunk-based Development vs. Git Flow, <https://www.toptal.com/software/trunk-based-development-git-flow>
- [63] Mantosh Singh, Top 10 best git hosting solutions and services in 2021, [https://docs.gitlab.com/ee/topics/gitlab\\_flow.html](https://docs.gitlab.com/ee/topics/gitlab_flow.html) , 1 September 2021
- [64] Narayan K, Top 10 Website to Host Git Repository Online, <https://www.scmgalaxy.com/tutorials/top-10-website-to-host-git-repository-online-2/> , 7 August 2021
- [65] Kentaro Wakayama, NoOps: What does the future hold for DevOps Engineers?, <https://www.scmgalaxy.com/tutorials/top-10-website-to-host-git-repository-online-2/> , 11 July 2021