# Setting up a Continuous Integration pipeline in Azure

# Table of Contents

[Setting up a Continuous Integration pipeline in Azure 1](#_Toc463526136)

[Table of Contents 2](#_Toc463526137)

[Getting started with Azure 2](#_Toc463526138)

[Creating an Azure account 2](#_Toc463526139)

[Setting up a resource group 2](#_Toc463526140)

[Adding members to the resource group 2](#_Toc463526141)

[Git 3](#_Toc463526142)

[Gitlab 3](#_Toc463526143)

[SonarQube 3](#_Toc463526144)

[Jenkins 3](#_Toc463526145)

[Install Jenkins virtual machine 3](#_Toc463526146)

[CloudBees Enterprise BYOL 3](#_Toc463526147)

[Bitnami 3](#_Toc463526148)

[Link to Git 3](#_Toc463526149)

[Link to SonarQube 3](#_Toc463526150)

# Introduction

This document describes how software development teams can use continuous integration tools on Microsoft Azure. The information can be applied to various software development projects, although Node.js was used as the focus and example project of this document. The details on how to set up a Node.js project and how to implement continuous integration in your software development team are out of the scope for this document. After reading this document, you should be able to set up a continuous integration pipeline in Jenkins, link your source code from Git to this pipeline, and review the quality of your code using SonarQube.

# Continuous integration

Continues integration is a development practice where developers can integrate their code in a shared repository. At the time the code is pushed to the repository it will be automatically tested. Because code integration takes place several times a day it is easier to increase the quality of the code. The goal of continues integration is to provide a set of tools that give rapid feedback to find out where things went wrong so that you can spend more time on building features.

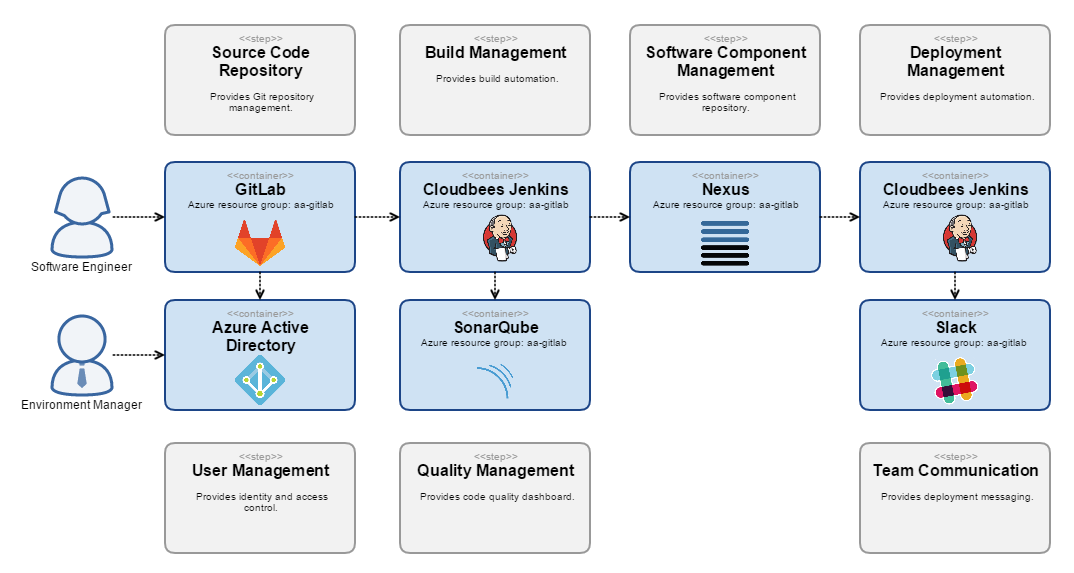
Principles:

* Maintain a single source repository
* Automate the build
* Make the build self testing
* Everyone can see what is happening
* Push the code frequently

How it works:

* Developers pull the latest version of the code from the repository.
* When developers are done the code can be pushed to the repository.
* Continues Integration server monitors the repository and checks out changes when they occur.
* The server builds the system and runs unit and integration tests.
* The server releases deployable artifacts for testing.
* The server reports if the test failed or succeeded.

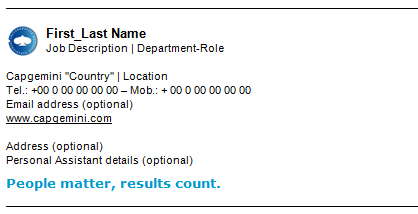
The following picture shows how the continues integration process works



# Getting started with Azure

## Creating an Azure account

No other information or image than the one shown here can be added in the email signature zone – i.e. between the two horizontal lines:



## Setting up a resource group

The Collaborative Business ExperienceTM (CBE) Label is automatically inserted in the html code of the signature template (hosted on Group server). This fix potential issues for PDAs (BlackBerry look & feel), PDF server, attachment notifications, size impact, etc.

## Adding members to the resource group

# Git

Git is a commonly used source control management tool. It allows teams to continuously merge their source code and provide all developers with the newest code at all times. This document assumes that you know how to commit, pull, push, and merge code to a git repository.

Any git repository can be linked to the continuous integration pipeline. Therefore, you can use your existing git repository. If you don’t have a git repository yet, or would like to instantiate one on Azure, the remainder of this section is of interest for you.

## Install Gitlab on Azure

Azure provides a git repository out of the box: Gitlab.

* First, you click the green plus button to add a new service.
* Then, search for “gitlab” to find the various Gitlab instances you can install.

The official Gitlab published two versions: Gitlab Enterprise Edition and GitLab Community Edition. The Enterprise Edition offers more features, but is not free to use. For more details see the [Gitlab website](https://about.gitlab.com/features/).

* Select the version that suits your project best.
* Click the “create” button.

**Basics**

* Enter the details for your virtual machine:
  + Name: provide a logical name for the virtual machine, e.g. “projectname-gitlab”
  + VM disk type: SSD is fine for now.
  + User name: provide a user name to log into the virtual machine.
  + Authentication type: select password for simple authentication. This password is linked to the user name.
  + Password: enter a password.
  + Subscription: the name of your subscription is automatically added.
  + Resource group: use existing user group that you created for Azure.
  + Location: select the location where you want the server to be located that is aligned with the legislations of your company.
* Click the “OK” button.

**Size**

* The next page shows options for the hardware that is used to host your virtual machine. Select the option that suits your usage requirements and budget best.

**Settings**

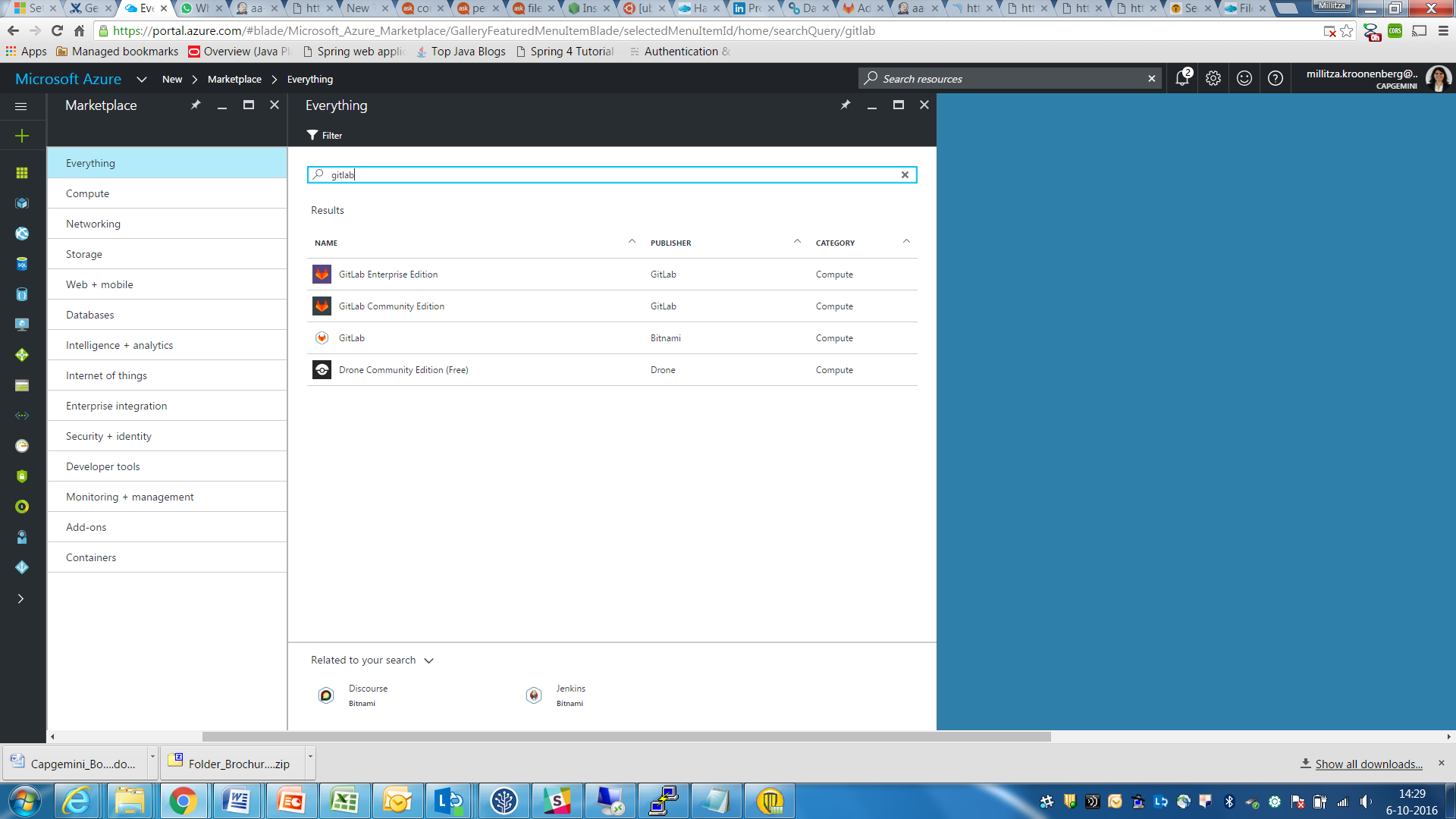
* All settings of the requested Gitlab virtual machine are summarized. There is no need to make any adjustments. You could connect the new virtual machine to existing services if you already set up virtual networks or storage accounts.
* Click the “OK” button.

**Summary**

* A full summary of the GitLab virtual machine is shown.
* Click the “OK” button if the summary is correct.

**Buy**

* The hourly rates for the requested services are shown.
* Click “Purchase” to start the creation of the virtual machine and related services.



# SonarQube

Code quality needs to be monitored to make sure the code is easy to understand and maintain. SonarQube provides code analysis services for multiple programming languages, including Java and JavaScript. All problems with the quality of the code are adequately summarized in a dashboard. This provides your team insight in the development of code quality and issues that need to be resolved.

SonarQube uses an SQL database to store the results of every analysis of code quality.

>>>Add information on SonarQube server and scanner <<<

## Create a Windows virtual machine on Azure

## Install SonarQube

## Install other dependencies (Java)

# Jenkins

## Install Jenkins virtual machine

## CloudBees Enterprise BYOL

## Bitnami

## Link to Git

## Link to SonarQube

|  |  |
| --- | --- |
| 1. **Download** the file called "Capgemini\_eMail\_Signature.zip" from the Visual Identity Guidelines site:   <http://visualidentity.capgemini.com/library/email_signature/>   1. **Save** it on your desktop. | 1.jpg |
| 1. **Unzip the file** (if needed):   Capgemini\_Email\_Signature.zip  eMail_Sig_Zip.jpg  After launching the WinZip tool, select all files and choose "**Extract**". | emsig.jpg |
| 1. **Select correct destination folder**   **Office 2007:**  Save the files under: C:\Users\username\AppData\Roaming\Microsoft\ Signatures  **The folder may be hidden. If so, please contact your local IT department.** | 2.jpg |