AZURE PIPELINES BEST PRACTICES

PETRO KOLOSOV

Abstract. This document contains generic recommendations to follow to implement proper and reliable DevOps processes using Azure, Azure Pipelines and Azure DevOps

Contents

1. Introduction

In order to implement reliable and consistent DevOps within a company using Azure, Azure pipelines and Azure DevOps, it is worth to follow set of predefined best practices as follows. We begin from CI/CD pipelines improvements

- **Security first.** Each software product state should meet security precautions like avoiding to store secrets hardcoded, avoiding to use vulnerable packages and dependencies etc. Few tools help us to follow this rule, for example `SonarCloud` static code analyzer, `Mend bolt` security scan, `Snyk` dependencies scan. So above tools must be utilized during CI/CD process, where is possible.
- **Store pipeline secrets securely.** It is a best practice to store secrets, files and certificates accessed by CI/CD pipeline inside `Azure KeyVault` or `Hashicorp Vault`. It is true that for developers it is much simpler to store secrets inside Azure DevOps library, however security first. Secrets must be stored inside vaults, if possible.

Date: January 29, 2023.

Key words and phrases. Software engineering, DevOps, Azure DevOps, Azure pipelines, CI/CD .

- **Use YAML pipelines instead Designer.** Switch from graphical UI pipelines to YAML where is possible. Few benefits of YAML: bulk edit, templates reuse over multiple projects, pipeline as code stored in version control so that build definition is straightforward.
- **Don't repeat yourself.** This recommendation is closely related to previous one, and proposes to use generic `YAML build templates` where is possible, it moves us from repeating same YAML build definition over and over again for every new project. Instead, we can simply reuse generic build definition passing different parameters, for example project name, build configuration or build platform.
- **Cache build assets.** Performance is a key factor for reliable DevOps process. Sometimes it takes around half an hour to finish deployment process. It is not convenient. `Caching build assets` like NPM or NUGET packages helps to decrease pipeline execution time, so that release or development processes will be faster.
- **Parallel everything, if possible.** Yet again, performance is a key factor for reliable DevOps process and sometimes CI/CD process takes long time. What we cen do instead is to update build definition so that it uses `Parallel Jobs`. For instance, we are always able to run build, integration tests and code quality checks in parallel. It might decrease CI/CD pipeline execution time in more than twice.
- **Optional CI/CD jobs run.** Sometimes we need to test or debug a single pipeline job or stage, so it is necessary to provide a set of `Runtime parameters` that allow to skip particular jobs, when pipeline starts manually.
- **Automate everything, if possible.** Automate everything like: Azure blob storage seed files, Databases initial state, infrastructure provisioning.

Email address: kolosovp94@gmail.com

URL: https://kolosovpetro.github.io